


# MX

## developer's journal

volume 2 issue 4 [www.mxdj.com](http://www.mxdj.com)



THE LEADING MAGAZINE FOR MACROMEDIA MX DEVELOPERS & DESIGNERS

ARE YOU A 

# JIGSAW

## AFICIONADO?

 **DREAMWEAVER**  
CSS

 **FLASH**  
Flash Meets Authorware

 **FREEHAND**  
Bulletproof Printing

 **COLDFUSION**  
FlashFusion

 **DIRECTOR**  
MOA City



\$9.99US \$9.99CAN



MX developer's journal page 11 2004 2.4

# 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

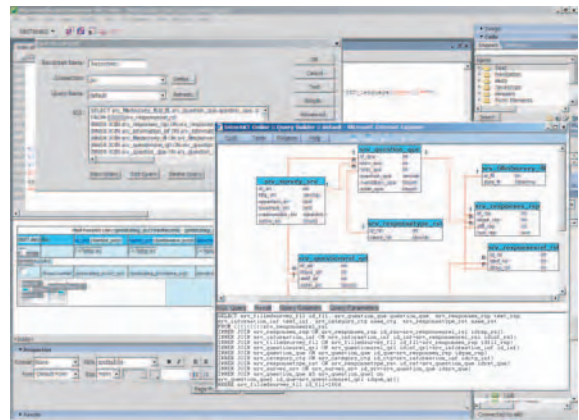
### Build unified add/modify forms

- 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



# 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.

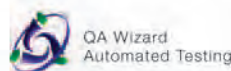
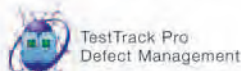
Scalable and reliable cross-platform, client/server solution supports Windows, Linux, Solaris, and Mac OS X.

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](http://www.seapine.com)

or call 1-888-683-6456





## CSS

*Finding your feet*  
by dave mcfarland



## Flash Meets Authorware

*Tricks for developers, from  
basic to complex*  
by derek stottlemyer



## Are You a Jigsaw Aficionado?

*Put it together and then  
make it move*  
by joyce evans



### on the cover

**y**ou're a jigsaw puzzle fanatic and want to create and share your own work. With Fireworks, you can create a puzzle with as little as two shapes, then set it dancing with Dreamweaver.



# 30

## It's All in the Object

*Reusable power is  
always a plus*  
by charles e. brown

# 7

## Many Uses for New Macromedia RoboDemo

*Enhancing the user  
experience*  
by jake sibley



# 10

## Why Central Matters

*It's not just about the  
browser anymore*  
by jesse r. warden



**Bulletproof Printing**  
*No one knows your job as well as you do*  
 by ron rockwell



**FlashFusion**  
*Integrating Flash MX 2004 and ColdFusion MX 6.1 with Web services*  
 by curtis p. hermann



**MOA City**  
*It's all in the architecture*  
 by tab julius

40



54



68



48

**Freaks and Geeks Unite**  
*Getting dynamic requires designers and coders to work together*  
 by tom green

66

**Integration**  
**New Media's Impresario Xtra for Director**  
*New use for an old friend*  
 reviewed by alec east

52

xile  
 Cartoon  
 by louis f. cuffari

74

vanguard  
 Line Art  
 by ron rockwell

# 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](http://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

[WWW.CFDYNAMICS.COM](http://WWW.CFDYNAMICS.COM)

866 - CFDYNAMICS

866-233-962-6427

**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](mailto: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

**Newsstand Distribution Consultant**  
Gregory Associates/W.R.D.S.  
732 607 9941 [BJGAssociates@cs.com](mailto:BJGAssociates@cs.com)

**List Rental Information**  
Kevin Collopy: 845 731-2684,  
[kevin.collopy@edithroman.com](mailto:kevin.collopy@edithroman.com),  
Frank Cipolla: 845 731-3832,  
[frank.cipolla@epostdirect.com](mailto:frank.cipolla@epostdirect.com)

**Promotional Reprints**  
Kristen Kuhnle, 201 802-3025  
[carrieg@sys-con.com](mailto:carrieg@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

# Many Uses for New Macromedia RoboDemo

*Enhancing the user experience*  
by jake sibley

Last December, Macromedia acquired eHelp Corporation, a San Diego-based software company offering an impressive set of tools that complements Macromedia products in very powerful ways. For MX developers, one of the most exciting additions is RoboDemo, an innovative tool with a surprising variety of uses.

Simply put, RoboDemo lets you easily create interactive demonstrations and software simulations in the Macromedia Flash format. You can use RoboDemo to record a series of actions on your computer screen as a compact Flash movie (SWF), then easily publish the movie online or send it in an e-mail.

As you can imagine, being able to quickly save and share anything you do on your screen creates tremendous potential for improving communication with both your customers and colleagues. With RoboDemo, you can demonstrate an on-screen activity to anyone, regardless of whether it is a customer visiting your Web site or a colleague down the hall.

RoboDemo is a powerful tool for enhancing user experiences you're already creating with Dreamweaver MX 2004, Flash MX 2004, and other Macromedia MX products.

If you're an MX developer working with marketing or sales to showcase your company's software or Web site, RoboDemo can be a powerful vehicle for conveying your product message. By recording a simulation of your software or Web site, you can express its value to potential customers quickly and effectively. It's always easier to show a product than to explain it, and since RoboDemo supports interactivity, you can even create a simulation that actively engages your potential customers. There is no better way to help your customers "get it."

E-learning developers already using Macromedia Flash, Dreamweaver, Breeze, Authorware, or any combination of these products will welcome RoboDemo as a powerful addition to their toolboxes.

RoboDemo simulations can include sophisticated interactivity, such as click boxes, buttons, text-entry boxes, and multiple-choice quizzes. RoboDemo includes branching and scoring capabilities that developers can use within stand-alone RoboDemo simulations or integrate into a larger e-learning course.

Since RoboDemo simulations are SCORM 1.2-certified and AICC-compliant, e-learning developers can effortlessly integrate their RoboDemo projects into Learning Management Systems (LMS), Macromedia Authorware, or other e-learning applications.

Regardless of the type of application you create, your users can benefit from a brief, animated tutorial that introduces them to your application and helps them get started. Use RoboDemo to create a tutorial that calls their attention to the major features of your application and simulates common tasks.

If your application is fairly sophisticated, you can use RoboHelp, RoboDemo's sister software, to create a comprehensive Help system that explains your application's features in complete detail. You can further enhance your Help system with individual RoboDemo tutorials.

Developers who provide technical support can use RoboDemo to create a knowledge base of tutorials for common topics or FAQs. Instead of having to read and follow difficult instructions, users simply view your demonstration and complete the task in question. And since the tutorials are compact SWF files, you can easily e-mail or publish them online to answer questions before an issue escalates to phone support.

## Internal Uses

By communicating visually instead of wasting time explaining things in words



or e-mail, RoboDemo helps you cut through the confusion that hinders many internal processes.

The less glamorous side of Flash development is the repetitive work that can burn up a lot of development time. Many developers wish they could skip the routine stuff and focus on more-creative pursuits. You can use RoboDemo as a companion to Flash to automate many of these tedious tasks, such as taking screen shots, creating mouse tweens, and animating text captions. With the RoboDemo FLA Module (sold separately), you can import your RoboDemo simulation into Flash and save it as an editable Flash file (FLA) that you can further enhance or integrate into a larger project.

Fixing bugs is a common task among developers and QA personnel, but these groups often waste time just trying to describe or replicate the bugs before they can fix them. Inaccuracies and misunderstandings throughout the process inevitably lead to unnecessary delays and frustration. With RoboDemo, you can capture bug behavior instantly, accurately, and permanently. Once a bug is discovered, you simply start recording and repeat your actions to recreate it. RoboDemo saves the steps you took and the resulting error in a Flash movie that

you can easily e-mail from within RoboDemo to appropriate colleagues.

We all know it's important for developers to understand the "big picture" of their applications, but who has time to learn? Hiring a new developer to help you reach your deadline can exacerbate this problem. How do you get the new guy up to speed on the app so he can get to work as soon as possible?

The answer is RoboDemo. Create a simple movie of your application that demonstrates the basic features, functionality, and purpose. You can make the movie in minutes, it only takes minutes to watch, and it will show your teammate how his or her work will integrate into the overall application.

It's a classic catch-22: you don't want to spend time building a feature that you won't include in an application, but you can't share your idea without spending the time to build it. RoboDemo breaks this cycle because you can use it to quickly mock up a feature. You can record the existing application interface and easily overlay small images in a series of frames to quickly simulate your idea – without coding a single line.

As you learn more about RoboDemo, imagine how you can apply it in your own work. Many customers have already let us know their innovative uses for this tool – uses they hadn't even anticipated. These comments are inspiring development on the next RoboDemo version release, so please continue to share them with us in the RoboDemo Community at [www.helpcommunity.ehelp.com/robodemo](http://www.helpcommunity.ehelp.com/robodemo). If you haven't tried RoboDemo yet, visit [www.macromedia.com/software/robodemo](http://www.macromedia.com/software/robodemo) and see what it can do for you. ☺

*Jake Sibley is a professional writer with experience in technical communications and marketing. As a communications specialist at Macromedia, Jake uses RoboHelp and RoboDemo on a regular basis. [jsibley@macromedia.com](mailto:jsibley@macromedia.com)*



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

**ADVERTISING**  
**Senior Vice President, Sales & Marketing**  
 Carmen Gonzalez, 201 802-3021  
[carmen@sys-con.com](mailto:carmen@sys-con.com)  
**Vice President, Sales & Marketing**  
 Miles Silverman, 201 802-3029  
[miles@sys-con.com](mailto:miles@sys-con.com)  
**Advertising Sales Director**  
 Robyn Forma, 201 802-3022  
[robyn@sys-con.com](mailto:robyn@sys-con.com)  
**Director, Sales & Marketing**  
 Megan Mussa, 201 802-3023  
[megan@sys-con.com](mailto:megan@sys-con.com)  
**Advertising Sales Managers**  
 Alisa Catalano, 201 802-3024  
[alisa@sys-con.com](mailto:alisa@sys-con.com)  
 Carrie Gebert, 201 802-3026  
[carrieg@sys-con.com](mailto:carrieg@sys-con.com)  
**Associate Sales Managers**  
 Kristin Kuhnle, 201 802-3025  
[kristin@sys-con.com](mailto:kristin@sys-con.com)  
 Beth Jones, 201 802-3028  
[beth@sys-con.com](mailto:beth@sys-con.com)

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

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

**ACCOUNTING**  
**Accounts Receivable**  
 Charlotte Lopez, 201 802-3062  
[charlotte@sys-con.com](mailto:charlotte@sys-con.com)  
**Financial Analyst**  
 Joan LaRose, 201 802-3081  
[joan@sys-con.com](mailto:joan@sys-con.com)  
**Accounts Payable**  
 Betty White, 201 802-3002  
[betty@sys-con.com](mailto:betty@sys-con.com)

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

**CUSTOMER RELATIONS**  
**Circulation Service Coordinators**  
 Shelia Dickerson, 201 802-3082  
[shelia@sys-con.com](mailto:shelia@sys-con.com)  
 Edna Earle Russell, 201 802-3081  
[edna@sys-con.com](mailto:edna@sys-con.com)  
 Linda Lipton, 201 802-3012  
[linda@sys-con.com](mailto:linda@sys-con.com)

**JDJ Store Manager**  
 Brundila Staropoli, 201 802-3000  
[bruni@sys-con.com](mailto:bruni@sys-con.com)



NORM MEYROWITZ  
PRESIDENT OF PRODUCTS



WEB DEVELOPERS ARE USING OUR MX PRODUCTS  
IN WAYS WE NEVER DREAMED. JUST IMAGINE  
WHAT THEY'LL DO WITH THE NEW MX 2004.

I've been endlessly amazed by the things our customers have done using our MX generation of products. And with all the new features in Studio MX 2004, it's going to be even easier and faster for them to realize their visions.

Dreamweaver MX 2004 helps you get that picture in your head turned into a web site faster than ever. That should be welcome news to the millions of web professionals who use Dreamweaver to create sites and applications. We've added things like CSS support, target browser check and improved code hinting to help you get through projects in far less time. And with Fireworks MX 2004 you can optimize web graphics up to 85% faster.

The new Flash MX Professional 2004 takes our industry standard tool for building rich content and applications to a whole new level. It really helps stretch your abilities—no matter where you fall on the designer-developer continuum. For development, we've added things like data-aware components and an extensibility layer so you can add your own features. For design, we've added high-quality, long-format video that's really impressive.

These are just a few of the new features. The products are available individually, but they really work well together in Studio MX 2004. Don't take my word for it. Download a free trial and read more at our web site.

Let me know what you dream up. norm\_01@macromedia.com

Copyright © 2003 Macromedia, Inc. and its licensors. All rights reserved. Macromedia, the Macromedia logo, Dreamweaver, Flash 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. \*Other valid for English, French and German products only.



Introducing  
Macromedia MX 2004.

[www.macromedia.com/go/2004](http://www.macromedia.com/go/2004)



# 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](http://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](http://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](http://www.nidus-corp.com) and [www.brainstormer.org](http://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](http://www.robertdiamond.com)



# Why Central Matters

*It's not just about the browser anymore*

by jesse r. warden

Central is important to developers for a number of reasons.

First, Central provides an application framework with which you can sell the applications you make. You just have to add a few purchasing details to an XML file and edit a Web form.

Additionally, the user is presented with information about how to purchase. Considering Macromedia's current deals with companies such as Intel and AOL, you can be sure that your applications will get exposure.

The evolving Flash Player will do better on the desktop than within the browser. There are fewer restrictions and a lot more room to add features and internal APIs developers may want or need. The politics of ubiquity rule the overarching plan of the Flash Player's development in a Web environment; you don't have those restrictions with Central.

Central makes deployment scenarios much easier. With Central's built-in App Finder and Free Finder applications, users are more likely to find your applications. Additionally, you know exactly what technology your end user has; currently Flash Player 6.0.65.0. There are no diversions from these specs, the only difference is OS. It's nice to have such assurances in development, as any developer knows.

But to really understand what Central means to us, we have to look at the history of where Central came from, including the problems Flash application developers have had in the past, as well as where we're going.

For an application developer, the browser is pretty lame. If you look at the history, starting with when the Web hit, a lot of application developers turned into Web application developers. Whether it was for back-end ASP/PHP/JSP coding, Java applications (and all the flavors), or copious amounts of JavaScript/DHTML, application programmers had a new playing field with many avenues to explore and a lot of uncharted territory.

In the multimedia realm, the CD-ROM

revolution made us aware of what was truly possible using all of the media disciplines (design, audio, video, programming) together to create unique and powerful interactive media. The push for the Web wasn't as easy, mainly because of bandwidth.

Even so, there was still a lot of interactive, offline work that application developers could do utilizing a lot of artistic media. Whether the range was a simple, interactive brochure for a kiosk, or a full-blown interactive application that may even have had Internet connectivity, Director was one of the few authoring programs that allowed you to merge the best that media could offer. However, as companies desired easier exposure to their customers (but with all the functionality) the plug-in wars got pretty harsh. Basic rights were violated, and IT tightened the noose of security – and for good reason. This made a lot of technologies inaccessible to a lot of people.

Enter Flash – this plug-in is everywhere. It is a cross-platform, lightweight media tool with which many people can view and experience whatever the creators make. You can create animations as well as applications on the same plug-in. Although Director has a great plug-in, even as a developer, I've had a lot of trouble getting it to work on anything even remotely associated with a firewall. Call it user error, but I haven't been able to get it to work easily – who's at fault doesn't matter. Flash, on the other hand, is very easy to install, and in many cases is already installed. This ease of proliferation has enabled a vast amount of the connected world to view Flash content.

Flash has been capable of connecting to outside data sources via GET/POST operations since Flash 4 and via XML and XMLSocket since Flash 5. With Flash 6 came Flash Remoting as well as the Flash Communication Server, built-in Central installation functionality, Web service code modules packaged with Flash MX 2004, and dynamic sound and video, all with the same real-time pixel-to-vector



# 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](http://www.activePDF.com)

engine that can scale your content, and all running in a secure sandbox. From a multimedia standpoint, whether for creating rich media ads or Web applications, Flash is the way to go.

But Flash can't go much further in its current environment. Considering all the factors working against innovation in the IE browser market, Microsoft's attitudes toward browsers and browser applications in general, and the challenges of integrating with a new barrage of browsers, the Flash Player has many challenges to overcome. Challenges like allowing Flash to easily respond to "next" and "back" navigation, in all versions of IE that are realistic (and who determines this?), as well as Mozilla and friends, and the \*nix flavors.

It's really about attitudes. Microsoft dropped development of IE for many reasons, but look where they're headed. Their prominent emphasis in Longhorn, and its development platform Avalon, is to create Web-connected applications that can be occasionally connected. This applies to laptops and devices such as PDAs and phones. There was a time when the push for win32 and Web applications as two product offerings was key to success and survival in technology. Studies show that a large portion (80% in some cases, but I never trust statistics) of Internet users utilize that connection without a browser. Applications that are connected have a lot more freedom outside the browser, not just in functionality, but in room to grow and adapt. Microsoft's enhancing their OS and allowing applications to access that rich GUI, as well as the classes they used to create it, opens a lot of doors for developers wishing to get into the rich GUI market. Flash, as its current IDE stands, is somewhat challenging to existing developers. The flair that Flash can offer, as well as its integration into disparate systems, makes it ideal for putting a nice face on applications – the issue lies in getting comfortable. The advent of Flex from Macromedia will definitely open the doors for enterprise application developers. Interestingly, I've read reports that a lot of smaller-time application developers feel they too would like the workflow of Flex. From how the initial offerings work, the Microsoft XML format and Macromedia's looked exactly the same in my eyes; one simply used ActionScript where code needed to be written, while the other used C#. Even the syntaxes were amazingly similar.

Macromedia, too, realizes that if they want to offer more benefits to developers,

they cannot unnecessarily bloat the Flash Player with features comparable to Microsoft's current Smart Client. From a developer standpoint, one of the great successes of the Flash Player was its proliferation to a lot of platforms and devices because of its small file size and lightweight implementation. Additionally, it's getting harder and harder to support each and every browser intricacy and still have the player operate and function the same on each platform without spending unrealistic development time or unnecessarily bloating the Flash Player to deal with these various discrepancies.

This is where Central comes in. The OS is our playing field now, instead of the browser. And by being freed of the chains of functionality and file-size restraints while retaining a secure sandbox, it has room to grow where we want it to. In contrast, because of the mess created on the Web by the lack of standards and the lack of applications following those standards, people have learned and have started following standards, which doesn't leave a lot of growing room. Going against the grain to push the envelope of design and media is now Okay only if you validate – lame.

My subjectivity is born of the fact that, during college, I chose the Director application route over Web design, merely because there weren't a lot of rules I had to follow. As long as we didn't crash your computer, the playing field for the team of designers I worked with was wide open. This allowed for a great number of custom-built solutions and great flexibility and creativity. Flash initially offered this on the Web, but as more functionality is needed from a programming standpoint, it's hard to deliver much without stepping on some file-size or nonubiquitous toes. The Flash Player on the Web has reached its breaking point in terms of what we can add without significant tradeoffs. Flash, already maturing into a really great development platform, saw the rise of some third-party applications to extend Flash where it was lacking in the desktop environment. Screenweaver, SWF Studio, etc., all offer ways to easily give Flash functionality that the player was lacking, such as reading the local hard drive and saving a text file, as well as additional APIs, some custom created by developers. As most of my work was in the application arena as opposed to Web, these programs were, and still are, invaluable.

This isn't saying Web applications are bad or are going away. Rich GUI applications that

people can use from anywhere with a Web browser and an Internet connection is a powerful concept. However, for people like me, who don't like dealing with browsers and all the fun little problems that Web developers and designers deal with day in and day out, a set platform is required. If you're a .NET developer, you get the .NET Framework; they also have one for devices. This is extremely cool for .NET developers, but what do Flash developers get? Originally, the same player with minor security restrictions freed. If you wanted functionality similar to what desktop apps had, you had to use a third-party application or Director. Now, we have Central. Just like some .NET apps, it has one-click installation. It also has built-in UI controls and a plethora of built-in APIs not offered in the Flash Web Player.

It's important to focus on where we're going, not where we are. Central, in its initial incarnation, is strictly for developers to get comfortable developing in it, as well as to provide Macromedia with the feedback it needs to better morph Central in a direction that is somewhat congruent with our Flash application development needs. We Flash developers now have a platform that has more potential for growth and the ability to adapt to our needs more easily and quickly. Not only that, but users can be upgraded more easily along with the applications that we deploy to them. Think of Flash with hardware acceleration and easier access to OS-integrated features. These are all possibilities depending on community need.

The really good stuff is yet to come. Once Macromedia starts marketing Central to the user base, packages it to places of great exposure (crossing fingers), and implements the AOL IM API (as well as related packages such as ICQ) along with some of the Breeze Live APIs for pods, Central will be a seductive platform to not only develop to, but also to sell applications on. ☺

*Jesse R. Warden, MXDJ's Flash editor, is a senior Flash developer at Surgical Information Systems, an operating room software company, where he currently uses Flash MX, Flash Remoting, .NET, and Oracle to create next-generation rich Internet applications for the OR. He contributed four chapters to the Flash Communication Server MX Bible and has written articles for various publications, including one for Macromedia for a DRK. [jessewarden@syst-con.com](mailto:jessewarden@syst-con.com)*



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](http://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



Unless you've been living under a rock - at least a rock without high-speed Internet access - you've no doubt heard of Cascading Style Sheets, or CSS. Over the next few months we'll be featuring articles on how to use Dreamweaver MX 2004 and Cascading Style Sheets to make your sites look great and work better. This month we'll cover the basics - to bring novices up to speed and to introduce these concepts to other MX Studio users who may have been too busy with ActionScript, Lingo, or CFML over the past few years to keep up with this rapidly evolving Web standard.



## What Is CSS?

Cascading Style Sheets – CSS – provides formatting control over HTML. With CSS, you can easily add borders to graphics, apply sophisticated typographic control to text, and create streamlined layouts without the use of tables. CSS is really a replacement for the anemic design controls offered by HTML such as the <font> and <table> tags and their associated properties. CSS lets HTML do what it does best – logically structure documents – while providing a rich formatting language to create visually sophisticated designs.

A stylesheet is a collection of styles – called rules – that define the presentation of elements on a page. For example, you could create a style that would make all <h1> tags appear in purple using the

pected places. Some of the dialog boxes you formerly used to add HTML-style formatting now use CSS. For example, as in MX, MX 2004's Page Properties dialog box (Modify ≠ Page Properties) lets you set the background color of the page. However, instead of setting the <body> tag's bgcolor attribute as in earlier versions, MX 2004 creates a new CSS style that defines the background color of the page. The end result is the same – a colored background – it's just that MX 2004 uses CSS to specify these page properties.

The page properties dialog lets you set a wide range of properties that have a global effect on a page. In addition to the properties available in MX – page margins, background color, background image, and link colors – MX 2004 lets you

like a paragraph or heading, Dreamweaver adds an HTML attribute called "class" to the paragraph – this property lets a Web browser know that the style affects the entire block. For example, you added a dark-blue color to the first paragraph of text on a page. Dreamweaver creates a new style, say style1. The HTML would look like this: <p class="style1">; all text in the paragraph is dark blue; and the style's name appears in the Property Inspector.

- If you selected just a portion of a paragraph, say a few words, and then applied some character formatting, Dreamweaver behaves slightly differently. It still creates a new style, for example, style2, and that style still appears in the Property Inspector. But

"In MX 2004, even complicated CSS designs often look nearly identical to the display of most standards-compliant browsers"

Verdana font, and underlined with a green dotted line. Or a style could add a 2-pixel red border to an image, and align it to the right edge of the page. In fact, each rule can contain many different formatting properties such as font size, background color, border type, and position.

## CSS in Dreamweaver MX 2004

Dreamweaver MX 2004 sports many new CSS features, not the least of which is its ability to render CSS-based designs with far greater accuracy than the previous version of the program. CSS designs looked like a garbled mess in Dreamweaver MX's Design view. In MX 2004, even complicated CSS designs often look nearly identical to the display of most standards-compliant browsers. In addition to MX 2004's display of CSS, how you create, use, and edit styles has also changed.

If you're a former Dreamweaver MX user, you'll find CSS popping up in unex-

pected places. Some of the dialog boxes you formerly used to add HTML-style formatting now use CSS. For example, as in MX, MX 2004's Page Properties dialog box (Modify ≠ Page Properties) lets you set the background color of the page. However, instead of setting the <body> tag's bgcolor attribute as in earlier versions, MX 2004 creates a new CSS style that defines the background color of the page. The end result is the same – a colored background – it's just that MX 2004 uses CSS to specify these page properties.

The Property Inspector has also undergone a major overhaul. By default, if you select text on a page and then set the text size, font face, and font color using the Property Inspector Dreamweaver creates a new CSS style that is applied to the selected text. The new style's name appears in the Property Inspector's Style menu with an unimaginative name like style1, style2, style 3, and so on.

*Tip: You can rename these style names by selecting Rename... from the Property Inspector's Style menu (see Image II).*

Depending on how you selected the text, before you applied formatting with the Property Inspector, one of two other things happens:

- If you selected a blocklevel element

it also injects an HTML tag called a span. The span tag might look like this: <span class="style2">formatted text here </span>. This tag will wrap around the text you selected, and the style is applied only to the text inside the span.

A good way to determine which method Dreamweaver used is to look at the tag selector at the bottom left of the document window. If you see something like <p.style4>, this means style4 is applied to the paragraph. If you see <span.style6>, Dreamweaver applied the style to just a portion of the paragraph, headline, or block-level element.

The cool thing about this technique is that you don't have to go through all of these formatting steps to apply the same look to other text on the page. Simply select the text you want to format and choose the name of the style you want to use – for example, style1 – from the Style menu.

Removing formatting you applied this



way is easy: click anywhere inside the formatted text and select None from the Property Inspector's Style menu. This removes the CSS formatting from the text.

However, if you decide to change the style of text you formatted using the Property Inspector some strange things can happen. Let's say you select a paragraph and apply a red color; select Arial or Helvetica, sans serif as the font; and change its size to 36 pixels. Dreamweaver creates a new style – style1. You decide you don't like the red color, so you immediately change it to a deep orange. Dreamweaver (as you'd expect) updates style1 with a deep orange color. In this scenario, everything is OK.

But, let's say you apply this style to another headline on the text. Now there are two headlines with the style1 style applied to them. However, if you select the second headline and change its color, Dreamweaver doesn't update style1. Instead, Dreamweaver creates another style containing just the new color and applies it (along with the first style) to the headline. To further confuse matters, the Property Inspector's Style menu displays "none", meaning no styles are applied to the headline, when in reality there are two styles – style1 and style2.

This may seem crazy, but it happens because Dreamweaver doesn't know

what you wanted. Did you intend to add color to just that one headline or change the color of the style. Either choice is possible – maybe you want this one headline to look mostly like the other, but stand out with a different color. Maybe not.

In any case, Dreamweaver's behavior may feel erratic and certainly won't help you edit the style. You have two choices if you want to update the style. First, make sure no other text on the page uses the style. You can then change the color, font face, and size using the Property Inspector and Dreamweaver will update the style. Once you're done – really done – then you can use that style freely on the page. A better choice, if you wish to edit the style itself, is to use Dreamweaver's CSS tools and not the Property Inspector.

There's another problem with using the Property Inspector and Page Properties dialog to format elements on a page. Dreamweaver stores the CSS style information in the <head> region of the page. While this is fine for just one Web page, it's a problem if you want to share styles among all the pages of your site. A better option is an external stylesheet. This is a separate file, with a .css extension, that just contains CSS rules. You can link any number of Web pages to this file. In addition to providing download savings (the external .css

file is downloaded once and then cached in the user's computer), external style sheets also ease site updating. If you want to change how a certain style looks throughout your site, you need to change only a single file.

*Tip: If you do use the Property Inspector or Page Properties dialog to add styles to a page, you can always export those styles into an external style sheet by File > Export > CSS Styles.*

## Creating Styles in Dreamweaver

Instead of relying on the Property Inspector or Page Properties dialog to create styles for your page, you'll have more control if you explicitly create your styles using Dreamweaver's CSS tools. The fastest way to create a new style is to right-click (Ctrl-click on Mac) on any empty area of a document, and from the contextual menu select CSS Styles > New to open the New CSS Style dialog box (see Image III).

*Tip: You can also access this dialog box by choosing Text > CSS Styles > New or clicking the New Style button on the CSS Styles panel.*

When you create a style you need to pick a selector type, supply a selector name, and tell Dreamweaver where to store the style information. A selector dictates what a browser applies the style to. For example, you could create a style that automatically formats all <h1> tags in blue, or a style that you manually apply to specific page elements – say you want the first paragraph of a page to be in slightly larger type than other paragraphs on a page.

Dreamweaver classifies selectors into three types: classes, tags, and advanced selectors. A class is most like the Word-processor notion of a style: you give the style a name of your choosing, like companyName, and apply the style manually to text or other page elements. Classes give you flexibility since you specifically identify which elements you want the style to apply to. For example, maybe you want some paragraphs on a page to look a particular way – create a style and apply it to just those paragraphs.

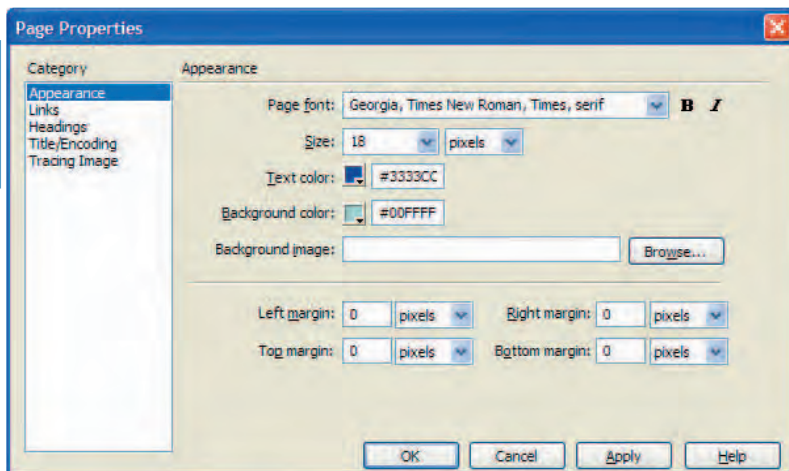


image I

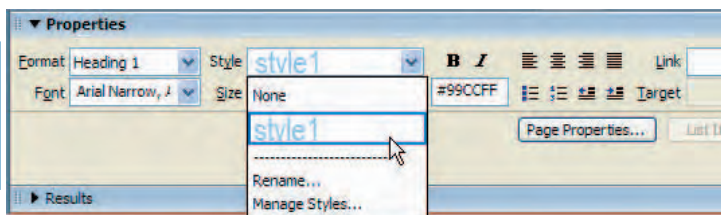
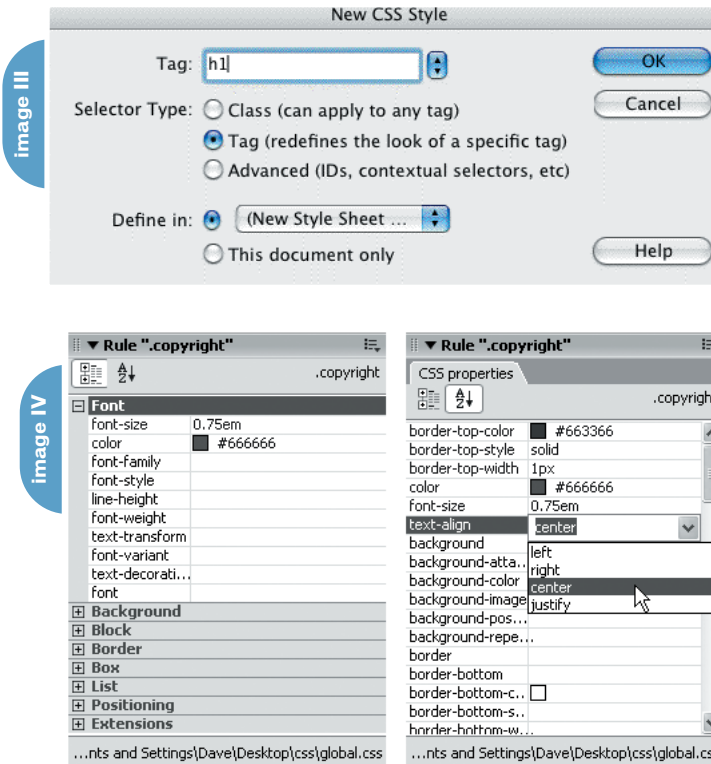


image II



*Note:* The name of a class style always begins with a period, like this: `.companyName`. However, if you forget to add the period when naming a class style, Dreamweaver is thoughtful enough to add it for you.

Tag styles, on the other hand, are indiscriminate. They define how a particular tag will look anywhere it appears on the page. For example, you could create a tag style that would replace the default bullet of every unordered list (the `<ul>` tag) on a page with a GIF image you created. Tag styles simplify formatting; you don't have to apply a tag style after it's created; the Web browser automatically does that whenever it encounters the specific tag.

Finally, what Dreamweaver calls "advanced selectors" are really a grab bag for all of the additional selector types CSS supports – and there are a lot. These include pseudo-classes (to let you format the different states of a link, for example, active, visited, hover), IDs (for identifying unique page elements such as the header, sidebar, or page footer), and descendant selectors to name a few. In future articles, we'll cover advanced selectors in greater depth.

After picking a selector type, you supply a name (for classes), tag (for tag styles), or selector (following the syntax

of CSS selectors), and finally tell Dreamweaver whether to store the style information in the current page – "In this document only" – or in an external style sheet. You can create a new external style sheet by selecting New Style Sheet from the menu, or, add it to an already attached style sheet. You'll frequently create a main external style sheet for your site, where you store global styles that affect all of the pages of your site. In addition, you may create additional external stylesheets for particular purposes (like formatting forms) or particular sections of your site that may have special design needs.

Clicking OK in the New CSS Style dialog either opens a save dialog box (if you're creating a new external style sheet) or the CSS Style Definition window. Here you have access to 67 different CSS properties for controlling everything from text properties – like font face, color, and size – to defining pixel-precise positioning for page elements like banners and sidebars. For a good listing of CSS properties – what they are and how they work – visit [www.w3schools.com/css/css\\_reference.asp](http://www.w3schools.com/css/css_reference.asp).

## Applying Styles

Depending on the type of style you created you may not need to do anything

to apply your newly created style. For example, tag styles are automatic; whenever the tag appears its style is applied. Class styles, on the other hand, must be applied manually. If you're a Dreamweaver MX user, you probably use the CSS Style panel to do this. In MX 2004, that same functionality is moved to the Property Inspector. The Style menu lists all of the classes available to a particular page.

You can apply class styles to any selection in the document window, whether it's a word, an image, or an entire paragraph. For example, suppose your company's name appears in a paragraph of text on a Web page that includes a class style named `company`. To format that text using the class style, select the name in the document window and use the Property Inspector's Style menu to select the class name. Similarly, to format larger selections, such as an entire paragraph, you'd select the paragraph and select the class's name from the Property Inspector style menu.

*Note:* If you choose a class name from the Property Inspector when nothing is selected (for example, when you click in the middle of a paragraph), Dreamweaver will apply the style to the nearest enclosing tag.

When you apply a class style (`.content`, for example) to a tag, Dreamweaver adds a special class property to the page's code, like this:

```
<p class="company">
```

On the other hand, if you apply a class to a selection that isn't a tag – a single word that you've double-clicked, for example – Dreamweaver wraps the selection within a `<span>` tag like this:

```
<span class="company">The National Exasperater</span>
```

This tag, in other words, applies a style to a span of text that can't be identified by a single tag.

Removing a class style is just as easy: select the styled text, and choose None from the Property Inspector's Style menu.

## Editing Styles

To change the properties of a style, you can always return to the CSS Style

Definition window by selecting the style in the CSS Styles Panel and clicking the Edit Style Sheet button (the icon with the pencil). However, MX 2004 introduces a more streamlined approach to editing styles: the Rule Inspector.

To use the Rule Inspector, first make sure the Tag Inspector is open. Choose Window > Tag Inspector or press the F9 key. Next, select the style you wish to edit in the CSS Styles Panel – this turns the Tag Inspector into the Rule Inspector (see Image IV).

The Tag Inspector has two different views: a category view, which groups the different CSS properties into the same seven categories used in the Style Definition Window (the left image in Image IV); and a list view, which provides an alphabetical listing of all CSS properties (the right image in Image IV).

So you can quickly determine properties already set for the particular style, Dreamweaver moves previously defined properties to the top of the list, highlighting the property names in blue and listing the property settings to the right. In Image IV, for example, the copyright class style is selected. In the Category view,

two font properties float to the top of the Font category: font-size and color. In the List view, you can see that the style actually has six different CSS properties set – border-top-color to text-align. This is a great way to quickly grasp what properties you've already defined for a style.

You set the value of a particular property in the space to the right of the property name. If that property already has a value, you can change it or delete the information there to essentially delete that property from the style. Fortunately, most of the time you don't have to type in the value: Dreamweaver provides the ubiquitous color box for any property that requires a color; properties that have a limited list of possible values include a pull-down menu of options; and, finally, some properties require a path to a file (such as when adding a background image to a style). In this case, Dreamweaver provides the familiar "browse for file" folder icon.

Other properties require knowledge of CSS and must be entered manually (in the correct format, of course). That's what makes this a more advanced option for experienced CSS gurus. Even not-so-

experienced users can find this window helpful, however. First, it's the best way to get a bird's eye view of a style's properties. The list view is especially helpful in this regard, since all of the defined properties are listed at the top of the window. Second, for really basic editing such as changing the colors used in a style, or assigning a different font for a style, the Rule Inspector is as fast as it gets.

### More to Come

If you're new to CSS, or still finding your way, this brief introduction should help you get your bearings in Dreamweaver MX 2004. In our next article, we'll explore some of the behind-the-scenes issues you should be aware of when setting up your style sheets, such as how to effectively use document type definitions, what validation is really about, and rules every style sheet should contain. ☺

*Dave McFarland is the Dreamweaver editor of MX Developer's Journal and the author of Dreamweaver MX 2004: The Missing Manual (O'Reilly/Pogue Press).  
davemcfarland@sys-con.com*

**Introducing the new FuseTalk.**

Collaboration

Discussion Forums

Flexible Security, 508 Compliance, Offline Capabilities, Reporting, Easy Integration and much more...

**1-866-477-7542**

2003 CFDJ Readers' Choice Award Winner for Best eBusiness Software & Best Web Application

[www.fusetalk.com](http://www.fusetalk.com)

Discussion forum solutions that make web-based collaboration risk-free and easy.

# A FLASH MEETS AUTHOR



In this article I show you some of the tricks I've learned over the years and give you some ideas for how to better combine Flash and Authorware in your work. I start with the basics and move to the more complex, and I cover some optimization techniques for speeding up Flash playback. You can find sample FLA and AW files at [www.sys-con.com/mx/sourceec.cfm](http://www.sys-con.com/mx/sourceec.cfm).

# ORWARE

by derek stottlemyer





## XTRA or ActiveX

There are two ways to embed Flash movies in Authorware, the Flash Asset XTRA and the Flash ActiveX Control. The XTRA is shipped with Authorware and does not require anything on the user's computer, while ActiveX is part of the Flash Player on Windows (see Table 1).

For XTRA, Authorware version 6.0 supports Flash 5, while Authorware 6.5 and 7.0 support Flash MX. Flash MX 2004 is not currently supported as an XTRA and would require use of ActiveX.

Regarding ActiveX, nearly everyone on Windows has a version of Flash ActiveX, but it is not guaranteed. I personally use XTRA because it will work on Macs and PCs, but many people use ActiveX or both ActiveX and XTRA. The samples in this article use XTRA.

## A Few Notes on Code

A few quick notes if you aren't used to scripting in both ActionScript and Authorware:

- In Authorware you set a variable with `":=": myVariable :=5`
- In Authorware you check equality with `"=:if myVariable=5 then...`
- In Flash you set a variable with `"=: myVariable =5`
- In Flash you check equality with `"==": if (myVariable==5) {...}`
- Arrays are "1" based in Authorware and "0" based in Flash.

Consider the following array:

```
myArray[ "2", "4", "6" ]
```

In Authorware the first value in an array is referenced by 1; myArray[1] would equal "2". In Flash it is referenced by 0; myArray[0] would equal "2" while myArray[1] would equal "4".

## Inserting Flash into Authorware

We first need to insert an existing

Flash SWF movie into Authorware. You load a Flash movie into Authorware through the menu: Insert>Media>Flash, which will open the Flash Asset Properties dialog, allowing you to select the file you wish to load. Use the Browse button to find the .swf you want to import (see Image I).

The Media and Playback options are pretty straightforward, but here are two items worth mentioning:

1. If you link to a file, make sure the link is relative; otherwise it may break when you package it.
2. Notice the Image and Sound checkboxes. These checkboxes decide whether your Flash movie has visual or audio elements. Though I imagine these are checked 99% of the time, it's good to know they're here since they can have such a dramatic impact on your Flash movies. I once spent an hour trying to figure out why the sound I'd added to my Flash movie wasn't playing before I realized I had the Sound checkbox unchecked – I had told Authorware not to play any sounds!

## Sending Messages from Flash to Authorware

Often Flash and Authorware don't need to communicate at all – developers just drop in the .swf and let it play. But that would make for a really short article, so let's take a look at sending information back and forth between the two programs and responding to that information. You may want to have Flash tell Authorware that a button has been clicked or that a specific frame has been reached.

Flash talks to Authorware using the getURL("myMessage") function. You can add getURL() to buttons or in keyframes to let Authorware know an event has occurred or a specific frame has been reached. GetURL() sends a message, and you have to set Authorware to capture the message; you do this using an Event

interaction. Drag an interaction icon to the Authorware flowline and add a calc icon to it. Then set the response type to Event, as shown in Image II.

Once you have set this response type to Event, the interaction will switch from a button to an "E". Click the "E" to change the properties of the Event interaction. You need to set this interaction to listen for getURL commands from Flash movies.

In the Sender section is a list of all the icons that send events that Authorware recognizes. In this sample I only have two Flash movies: "10\_10\_10" and "10\_20\_10". Double-click the icon (or icons) that interest you. When a "Sender" is selected, the Event field will display a few choices. We are looking for the getURL event, since that is what Flash is using to communicate (see Image III).

Notice the "x" next to each name that has been double-clicked. You can double-click again to deselect one or more items. In Image III I have also double-clicked the getURL Event – it also has an "x".

Note: You have to select an event for each sender. It may look like getURL is selected for both Icon 10\_10\_10 and Icon 10\_20\_10, but it is possible that getURL is only selected for Icon 10\_20\_10. Take your time to get this right, since this is a crucial part of communication.

We still aren't quite finished; we have to capture the actual message sent from Flash and act upon it. This is done with only one line of code.

In a calc icon in this Event interaction you need the following code:

```
result:= EventLastMatched[#urlString]
```

It may look complicated, but the code simply asks for the string sent from the Flash movie. The "result" will be whatever is sent from the getURL() function in Flash. GetURL("myMessage") in Flash would make "result" in Authorware equal "myMessage".

Sending messages to Authorware makes it possible to use Flash to create menus or user interactions that can be directly acted upon in Authorware. This allows Flash to be a tool in your courseware, not merely a media type.

## Authorware to Flash

Communication often needs to go in both directions. If Flash can tell

table 1

# X T R A A c t i v e X

Platform s	Mac / PC	PC only
Requirements	None	Flash Player needs to be installed
Transparency	Allowed	Not available
Performance	Lower than Active X	Better than Xtra
Functionality	Limited commands	More commands available

Authorware when a button is clicked, why not have Authorware tell Flash which buttons to display? Or tell Flash when to play an animation or what data is needed in a chart?

Communicating from Authorware to Flash is simpler than communicating from Flash to Authorware. Authorware uses three main function types – CallSprite(), getSpriteProperty(), and setSpriteProperty(). Each function is relatively straightforward, but you have a lot more options than you have when you send messages from Flash to Authorware. Authorware uses CallSprite(@"Flash Icon", #setVariable, "myVariable", "myValue") to send messages to Flash.

In Image IV, "myCount", a variable, which has to exist in Flash already, would be set to "5".

Note: Even though 5 is a number and "myCount" is most likely a numeric value, the parameter passed has to be a string. Therefore, either wrap it in quotes or use the String() function in Authorware. If it isn't sent as a string, the value of the variable will not be changed in Flash.

This Authorware sample has two icons, a Flash movie called "myFlash" followed by a calc icon that sets a variable in the "myFlash" movie called "myCount" to 5. The corresponding Flash movie file is one frame with a variable called "myCount" (initially set to "1") and a text box that displays "myCount". The text box properties look like Image V.

I use the Var property since it automatically updates when "myCount" changes. Before getting to "The Hard Part" I want to wrap up with some other common functions used to communicate from Authorware to Flash.

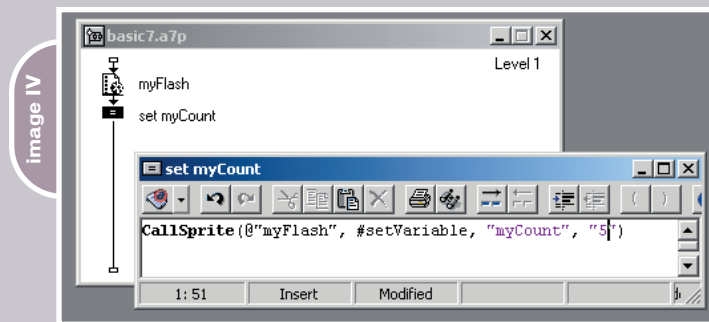
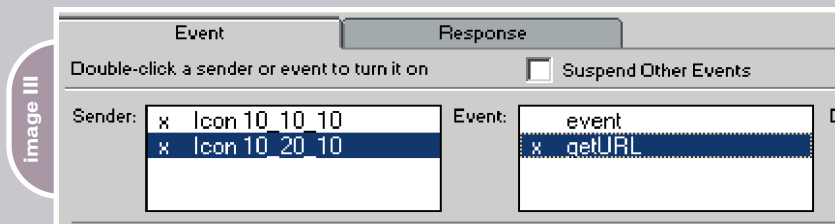
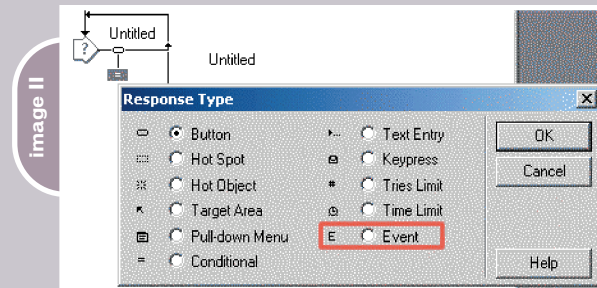
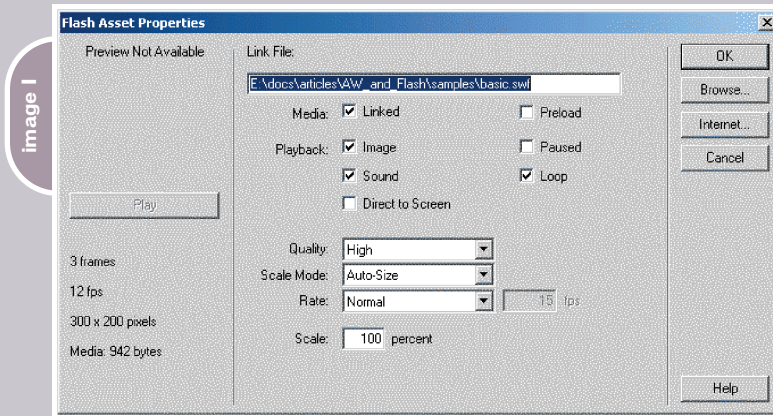
The corresponding function to #setVariable is #getVariable.

```
result:=CallSprite(@"myFlash",
#getVariable, "myCount")
```

You can use this to verify that your #setVariable worked or to check a value if Flash is also changing a variable's value.

Finally we have playback functions, which control the Flash movie timeline (see Code I).

It may take a while to get used to these functions, but eventually they become second nature.



## The Rewards

Aside from nice vector animations, Flash offers additional solutions for tricky problems in Authorware.

- **Components:** Download the Flash UI components sets 1 and 2 and the Flash Charting Components from [www.macromedia.com/cfusion/exchange/index.cfm](http://www.macromedia.com/cfusion/exchange/index.cfm). These components are usually easy to configure and use. I'm not a big fan of the WinControls in Authorware, so Flash's Tree Menu and input text fields offer great alternatives that are also cross-platform.

- **Embedded fonts and text features:** Flash has a great deal of text controls, and I love that you can call asfunctions or hyperlinks from text. When you combine this with embedded fonts it is a great way to handle tricky text issues. Authorware's inability to embed fonts is a hassle, especially when you have to deal with different font encodings for different languages. Flash can allow for HTML, dynamic hot text, and dynamic text styles – all of which are difficult for Authorware.
- **XML:** I like the way Flash handles XML better than the way Authorware does,



and Authorware can't write XML nodes easily. I tend to embed Flash files when I'm working with XML.

4. **Animated buttons:** I've made a few courses with animated buttons that move to show progress. This can look very nice, but don't overdo it.
5. **Pan and zoom:** Panning isn't too hard in Authorware, but zooming is. This is one of Flash's strong points. From Authorware, look into the #viewScale, #viewH, and #viewV functions. For an example of this, check out the "Flash.a7p" show-me file on the Authorware CD.

### The Hard Part

In my last example I used the Var property of a textField instead of the more current format of myTextFieldName.text=myCount. I wanted to whet your appetite a little before I got to some of the difficulties you will encounter when integrating Flash and Authorware.

I used Var because it's easy. There's nothing wrong with easy, but it's often more limited than other ways of doing things. The Var property updates automatically when its corresponding variable updates. However, when you set the text field in ActionScript code, it only updates when that code is run.

You can also use dot syntax with #setVariable, so CallSprite("@myFlash", #setVariable, " myTextFieldName.text", "5") would also work. But what happens if you want to do more than just display this variable in your Flash movie? If you need to modify the variable and respond to it, then you need to do it in ActionScript code. If you need to do it in code, then you need to ensure that all the necessary information is already in Flash before that code is executed.

Many beginners have Flash loop so that the frame containing the needed code is hit often. This works, but it's inefficient and will use a lot of processing power. It also isn't very accurate if you need to act upon more than one simple piece of information.

The second option is to use stop() functions in Flash and then use #setVariable followed by #play to hit the frame with your code in it after the variable has been set.

This also works, but leads to awkward Flash files that are confusing to modify and update. With one variable, it isn't bad. But can you imagine a 100-frame animation with variables set at different times?

This is the hardest part of using Flash in Authorware: responding to events and changes in a timely manner. It takes planning to create integrated projects that work the way you want without either using up processes by continuously looping or creating convoluted Flash files.

### Tips and Tricks

Okay, here's the part you've probably been waiting for – practical solutions.

#### Code for Flash Icons in Authorware

I almost always attach the code shown in Image VI to every Flash Icon on the flowline.

"Movable@IconID" should be pretty easy to understand: it just means that a user can't drag the Flash movie around the screen. You may want to allow Movable at times, but usually I want my Flash movie to stay where I put it.

The second line makes the variable "FlashID" a reference to your Flash icon, allowing you to modify the icon without referring to it by name.

```
CallSprite (@FlashID...)
Instead of
CallSprite ("myFlash"...)
```

*Note:* With "FlashID" you don't use quotes. Use @FlashID, not @"myFlash".

Why is this helpful? A typical Authorware piece of mine may have a dozen or more Flash movies, and this code allows me to accomplish several things:

- **Reuse code where applicable:** I usually use code to pass variables or navigate through a Flash movie file. This allows me to change parameters without hav-

ing to decide which .swf I need to target.

- **Code is easier to read:** "@FlashID" tells me I'm working with the current Flash movie. If you use icon names, then it's not often clear what you are changing. (@@"01\_02\_030\_Staff.swf")?
- **Global controls in Authorware:** This is the biggie. If I always know that FlashID is the most current Flash movie, then I can add a pause/play button that will always work!

Image VII shows some code that I've used for a global pause button. It should be placed in a perpetual button at the top of your flowline.

For the pause/play button see Code II.

It's that easy. You could also add other code here as well. You may want to tell the Flash piece it is inactive, or perhaps toggle the Direct to Screen property so that it doesn't take up as many resources.

#### Code for Flash Movie Completed

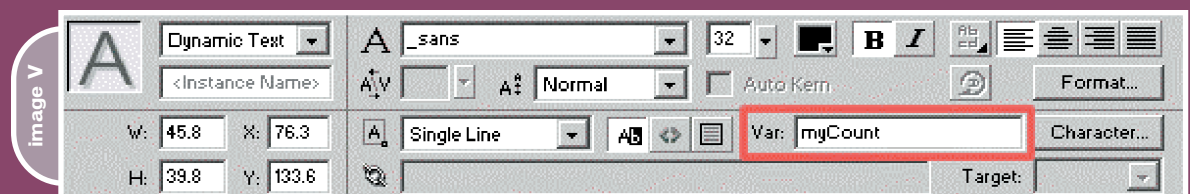
I have created a number of courses that don't allow users to proceed in a section until they have completed all activities on each page. It's therefore necessary to know when a Flash file is finished so that the Next or Continue button can be activated.

It is sometimes possible to use the following code in the active if field of your Continue button:

```
getSpriteProperty (@FlashID, #Frame) =
getSpriteProperty (@FlashID,
#frameCount)
```

This won't work, however, if the last frame has either submovie clips or actions the users need to complete. This is why I tend to add a line of code in each Flash piece letting Authorware know when I consider a movie to be complete.

While occasionally you can get away with checking if the current frame is the last frame, it is easier to simply add a little code in Flash to tell Authorware when it is finished.







# 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](http://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)





## Controlling Flash Files with Multiple Segments

It is often easier to have one Flash file with multiple scenes instead of maintaining several Flash files with similar content. While you want all the content in the one Flash movie, you'll most likely want to control which segment is played, and when, from Authorware.

There are easy and hard ways to do this, with the easy way having more limitations. The easy way is to use #gotoFrame at the appropriate times to get to the part of the Flash movie that you want. Here are a few pointers:

- The Flash file will ignore stop() commands in the frame it's sent to. So if you use CallSprite(@FlashID, #gotoFrame, 5) you'll need a stop() function in frame 6 or higher or the Flash movie will simply continue playing. Another possibility would be to also use CallSprite(@FlashID, #stop) from Authorware.
- #gotoFrame will not reload the current frame. I've worked around this by navigating twice – once to frame one and then a second time to the frame I want. If your #gotoFrame commands aren't working, make sure you are jumping to a distinct new frame.
- Using Frame labels in Flash is a lot safer than using Frame numbers.

A more complex way of controlling Flash playback is to create a control frame in Flash. All navigation takes place from this frame based on a variable that is sent from Authorware. So while Authorware decides when and where to navigate, Flash actually handles the navigation internally.

Authorware would use #setVariable indicating the desired scene and then would use #gotoFrame to the control frame. Since the control frame always uses goto(), it will never be the current frame for more than an instant.

## Advanced Data Sharing

You will most likely reach a point where you want to send more than a simple string or number to your Flash movie. You have to send everything as strings, but here are a few methods that I have used to get complex data types to Flash.

### Passing Arrays to Flash

Linear arrays aren't too hard. Say you have an array "myArray=["fish", 7]" in Authorware:

1. Convert the Array to a string and remove the brackets.

```
temp:=Strip("[]",String(myArray))
```

2. Send this value to the Flash Movie.

```
CallSprite(@myFlash, #setVariable, "PreArray", temp)
```

3. Navigate to a frame containing the code in step 4.

```
CallSprite(@myFlash, #gotoFrame, n)
```

4. Add the following code in Flash in frame 'n'.

```
myArray=PreArray.split(",")
```

5. myArray is now an array in Flash that mirrors myArray in Authorware.

*Note:* The two things to watch out for are commas in your Array and European deployment. In Europe, a semicolon is used to separate lists instead of a comma. If either of these is the case, you will need to loop through your array, adding each index to a string with a different, perhaps multicharacter, separator.

A colleague suggested the following method for sending arrays. In this example you would need an empty array in your Flash Movie called "\_array" (see Code III).

You must ensure that "\_array" is empty, however, as this method will not clear out pre-existing indexes if the original array has a greater length than the new array.

Multilinear arrays take more effort. I have had to create a pseudo-subroutine for those. I've never had to pass a property list (an associative array in Flash), but I imagine this could be done with a subroutine, as well.

### A Fake Listener in Flash

It is much easier to let Flash handle its own navigation than to have Authorware tell it which frames to go to. This is because edits to the Flash movie often add or remove frames, which can break all of your #gotoFrame functions if the frame numbers you jump to change.

Therefore, I built upon the control frame idea discussed before and created a control Movie Clip for my Flash movies. A simple control Movie Clip has two frames and continuously loops to check for new information from Authorware. Frame 1 is shown in Code IV; Frame 2 is shown in Code V.

This is just a sample; Frame 2 could also be a case statement, and a real example would probably have more navigation options. On the Authorware side you only need to send one variable and everything else is taken care of by Flash.

```
CallSprite(@FlashID, #setVariable, "authorwareEvent", "pause")
or
CallSprite(@FlashID, #setVariable, "authorwareEvent", "scene3")
```

If you are fairly comfortable coding in Flash, an even better solution would be to use the Watch() function introduced in Flash MX (see Code VI). Code VI can be

**"PAN AND ZOOM IS JUST NOT SOMETHING I WOULD CONSIDER IN AUTHORWARE ALONE, BUT WITH FLASH IT'S FAIRLY EASY"**

# 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](http://www.webcoretech.com)



placed on the main timeline and does not require any looping.

### A Pseudo Subroutine

Probably the most complex integration of Flash and Authorware that I've attempted is to set up a tree menu in Flash based upon a directory structure in Authorware. Let's say I have a course with frameworks for modules, subframeworks for lessons, and sometimes subframeworks for pages. I can't say I won't have deeper frameworks as well, so I can't limit my code to three layers deep.

This is too complex to describe here, but it can be done by looping a decision icon in Authorware that sends parent and children information for each Framework to Flash and then waits until Flash lets Authorware know that it has added the nodes to the tree Menu.

- Put your Flash movie on a Layer above 0 and which has no other display items.
- Minimize on-screen events if possible, in Flash and Authorware.
- Ensure that Mode is Opaque (2) and Direct to Screen (3) should be checked when possible.

Let me talk about Mode and Direct to Screen for a moment. Direct to Screen will tell Authorware to display the Flash movie above everything else and is significantly faster than placing the Flash movie on a layer. However, you can't have a transparent Flash movie when you check Direct to Screen – the transparency simply doesn't take effect.

You'll want to use Direct to Screen unless there is something else that you need to display above your Flash movie or if you need the Flash movie to be transparent. It may be necessary to toggle Direct to Screen on for complex scenes and off for standard use (SetSpriteProperty(@FlashID, #directToStage, true), SetSpriteProperty(@FlashID, #directToStage, false)) if you need that level of control.

Setting Mode to Transparent is a wonderful feature, but it will slow things down, especially since you can't set the Flash movie to Direct to Screen. You only want to use transparency when you need to. There are modes other than Opaque and Transparent, but they don't seem to have any affect on Flash Movies, so stick to Opaque or Transparent.

If you use Transparent, then also be sure to set Authorware's GlobalTempo to the same frame rate as your Flash Movie (Using 'GlobalTempo:=n') in an Authorware Calc.)

If you still need to increase playback performance, you have additional options in the Options button from the Properties panel.

### Additional Options:

- You can set Quality to Low, which will turn off anti-aliasing.
- Experiment with Rate. I'm a little fuzzy on this, but I believe "Normal" skips frames in the Flash movie if it is lagging behind. Both Lock-Step and Fixed allow a little more control for frame playback.
  - Lock-Step will match frames per

second to Authorware's global tempo (which can be changed in Authorware with this code 'GlobalTempo:=n'). I believe this will force Flash to refresh at this rate. Flash Asset Help says this is the best performance of the three options.
 

- Fixed allows you to set a playback rate to a specific fps rate.
- Experiment to find what works best.

- Avoid unnecessary looping in your Flash Movie.

There are other excellent tips in the Flash Asset documents [path to Authorware on your computer]\xtras\FlashAsset\Help\FlashAsset.html. If you're still having trouble, post a question to the Authorware Listserv <http://listserv.cc.kuleuven.ac.be/archives/aware.html>.

This is an excellent resource where new users and experts exchange ideas and assistance, with Flash performance being a frequent topic.

### Conclusion

I hope that this article has given you some new ideas as to how to blend Flash into your Authorware work and provided enough code snippets to get you started. I started as an Authorware developer, so my knowledge of what Flash could do was limited and learning what was possible has been a challenge over the years. Pan and Zoom is just not something I would consider in Authorware alone, but with Flash it's fairly easy. Consider whether Pan and Zoom would help in your current courseware.

If you are writing courseware that involves scheduling or history, consider whether the calendar component would make things easier. Would graphing components add a nice visual element?

I hope that you are able to use these samples. Look for these and other source files on my Web site as well ([www.guitar-learning.com/awflash](http://www.guitar-learning.com/awflash)). Happy Coding! ☺

*Derek Stottlemeyer has produced a number of guitar- and music-related software titles using Authorware, Flash, and Central. He also develops training and Web applications for the automotive, health, and financial industries. derek@guitar-learning.com*

image VI

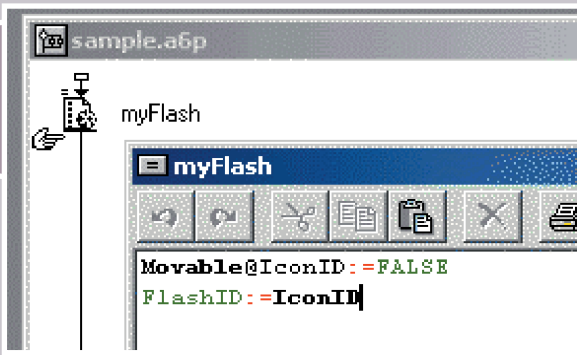
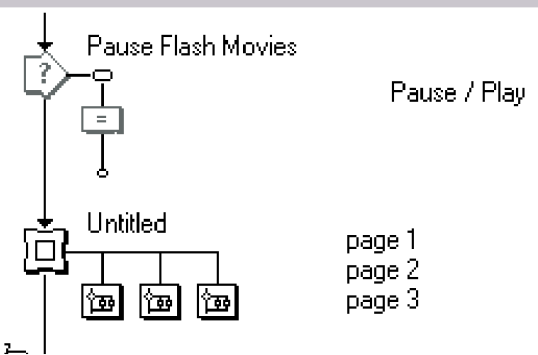


image VII



### Optimization Techniques for Flash Playback

If you've ever dropped a large animated Flash movie into Authorware, you've noticed that it can slow things down a lot! Here are some tips for getting the best playback possible. We'll mostly be looking at the display properties.

You should always:

- Keep your Flash files as small (in pixels) as possible.

code I

```

CallSprite(@"myFlash ", #goToFrame, 10) - (Notice that
10 here is a number!)

If GetSpriteProperty(@"myFlash", #playing) = False
then
    CallSprite(@"myFlash ", #play)
end if

If GetSpriteProperty(@"myFlash ", #playing) = True
then
    CallSprite(@"myFlash ", #stop)
end if

```

code II

```

--Check to see if the Flash Movie is onscreen. If
DisplayWidth > 0 then it is being displayed.

If DisplayWidth@FlashID> 0 then
    If getSpriteProperty(@FlashID, #playing) then--Check
if movie is playing
        CallSprite(@FlashID, #stop)-- if so, stop it
    Else
        CallSprite(@FlashID, #play) -- otherwise restart it.
    End if
End if

```

code III

```

myList := ["a", "b", "c", 6, 9090, #mySymbol, "Hello
World From Authorware"]
repeat with i := 1 to ListCount(myList)
    --_array[0] in Flash is _array.0 in the Flash
Asset Xtra
    CallSprite(@flash_id, #setVariable,
    "_level0._array."^String(i-1), String(myList[i]))
end repeat

```

code IV

```

//Look for new information
eventFromAuthorware=_parent.authorwareEvent;
//Since this is a Movie Clip within a Movie Clip, you
have to use _parent to get the variable
'authorwareEvent' which is on the main timeline.

```

code V

```

//Check to see if we need to respond to an event
if (eventFromAuthorware=="pause"){
    _parent.stop();
    _parent.subMovie1.stop();
    _parent.subMovie2.stop();
    //this is the easiest way I've found to pause and
play complex Flash Movies
}else if (eventFromAuthorware=="play"){
    _parent.play();
    _parent.subMovie1.play();
    _parent.subMovie2.play();
}else if (eventFromAuthorware=="scene3"){
    _parent.gotoAndPlay("scene3")
}

```

code VI

```

eventFromAuthorware="";
_parent.authorwareEvent="";
//remember to clear both Variables to make sure you
don't repeat actions.

```

```

var eventFromAuthorware;
this.watch("eventFromAuthorware",
function (prop, oldval, newval) {
    if (newval == "pause"){
        this.stop();
        this.subMovie1.stop();
        this.subMovie2.stop();
    } else if (newval == "play"){
        this.play();
        this.subMovie1.play();
        this.subMovie2.play();
    } else if (newval == "scene3"){
        this.gotoAndPlay("scene3");
    }
    newval = undefined;
});

```

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](http://www.serverside.net)



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



(888) 682.2544  
[hosting@serverside.net](mailto:hosting@serverside.net)  
[www.serverside.net](http://www.serverside.net)

**"We don't have a lot of time to write down documentation...but with Camtasia Studio, we can document our software with video tutorials."**

—Fred Shepardson, PhD, Mathematician, Management Consultant

Full-motion video tutorials of any application.



TechSmith®  
**CAMTASIA STUDIO™**  
 SHOW THE WORLD

# It's All in the Object

*Reusable power is always a plus*

by charles e. brown

Approximately 10 years ago a new word started to appear in programming circles: objects. The theory was that little bits of code, doing a specialized job, could be prewritten and plugged into existing code as needed.

Several years later a programming language called C++ was introduced. This gave the programmer an entire library of prewritten code called class files. Each class file had one specific task assigned to it and could be plugged into any project easily as needed. This saved programmers incredible amounts of programming and debugging. Today, the word object is used by nearly every program, including Fireworks.

In this article, we will examine what objects mean in Fireworks and how they can save you time and work. We will use very simple examples to illustrate the concepts. However, you will easily be able to apply these same concepts to

more complex scenarios. For purposes of understanding, we are also going to use some non-Macromedia terminology.

Begin by drawing a simple circle on your canvas. You can fill it in with any color you want. Once finished, select Modify > Convert to Symbol (or press the F8 key). You can give it any name you want. For example, call it myCircle and select the option to make it into a Graphic.

Whether you realize it or not, you just created an object. Take a look in the Library panel by selecting it in Panels (see Image I).

Think of this as a template that can be used to create other objects. The Library panel is handy in that it shows us what the object looks like and, in the bottom window, shows us the name and the type of object it is.

For example purposes, delete the original circle on the canvas.

Let's assume our design needs four circles, one in each corner. All we need to do is drag the circle, using either the graphic or the name, to each of the four corners of our canvas (see Image II).

Believe it or not, you just had your first lesson in object-oriented design. Using the template in the library, you created four new objects on the canvas. In object-oriented programming (or OOP) parlance, we would say that we created instances of the object. Each one is a duplicate of the original.

This has already saved us a lot of work because we did not have to redraw the circle four separate times. But what happens if we need all of the circles to be a different color?

We can now double-click on one of the circles. This will take us to a symbol-editing canvas (you can also access the edit area by double-clicking on the sym-

bol in the Library panel). Change the color to a different color and select the Done button, located at the top of the canvas. All the objects change to whatever color you selected.

Here lies the second benefit of objects: you can change the object's attributes (color, shape, etc.) in one place and all of the objects will automatically conform to it.

What happens, however, if we need to set one of the objects to a different attribute? For instance, let's say we want it to be a different color.

Select the object and then select Modify > Symbol > Break Apart. The symbol you broke off is no longer attached to the symbol in the library and can have any of its attributes changed without affecting the other instances.

Unfortunately, this also keeps the symbol from being changed from the library if you need to do so.

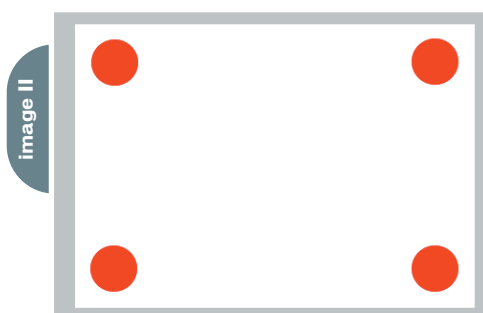
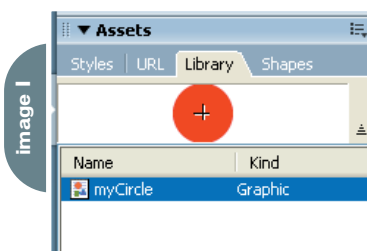
## Preparing Your Objects for JavaScript

We often use our objects within a programming environment. In most Web design, the language of choice is JavaScript. The interesting part about JavaScript is that it is a variation of the Java programming language. Java is one of the most notable object oriented languages available.

Many times we need to reference the objects with JavaScript. To do that, each instance must have a unique name. This is where I am going to depart from Fireworks terminology a bit.

Click on the symbol and, in the Property inspector, name the object in the Symbol field. Once again, in object-oriented parlance, we call these names *object references*.

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



JavaScript, like most languages today, is case sensitive. Because of that, your object references should conform to the following naming conventions.

- No spaces – if you need to make something look like a space, use the underscore.
- All lower case, except for mid-word capitalization – for example, addressBook.
- Begin with a letter, not a number.
- Only use alphanumeric characters with the exception of the underscore.

Use names that make it easy to identify the object clearly.

## Using Your Objects Elsewhere

Up to this point you've learned how to create an object and see how you can save us a lot of work by being able to instantiate it on the canvas as many times as necessary. In addition, we can change attributes universally.

What happens if we want to use our objects, or symbols, on different canvases?

Fireworks allows you to export and import your objects between different projects.

Using the Library panel, open the Options menu and select Export Symbols. You will be taken to a box that shows you all of the symbols in the present project (see Image III).

Here you can select the symbols you want to export. Once you've done that, select Export. After you select the folder you want to save your objects into it and select OK; they will be saved to an external PNG file.

If you now want to import your objects into another project, open the Library panel in the new project and select Import Symbols...

Once you maneuver to the folder that contains your symbol, select OK. You will be taken to a box that looks nearly identical to the one you saw when you exported. (Note: The figures shown here were created with Fireworks MX 2004. The screens in Fireworks MX look a little different but they have identical functions.) Select Import.

Your symbols should now be in the Library panel, as they were in the previous project.

Advertiser	URL	Phone	Page
ActivePDF	www.activePDF.com	11	
CFDynamics	www.cfdynamics.com	6	
CFUN	www.cfconf.org/cfun-04/	Cover III	
EdgeWebHosting	www.edgewebhosting.net	13	
Electric Rain	www.erain.com/mxdev.asp	73	
FuseTalk	www.fusetalk.com	19	
HostMySite.com	www.hostmysite.com/mxdj	25	
Interakt	http://www.interaktonline.com	Cover II	
Macromedia	www.macromedia.com/go/2004	9	
Macromedia	www.macromedia.com/into	Cover IV	
Seapine Software	www.seapine.com	3	
ServerSide	www.serverside.net	29	
TechSmith	www.techsmith.com	29	
WebCore Technologies	www.webcoretech.com	27	

## Building Libraries

Many professional designers build entire libraries of prebuilt symbols, or objects, that they can call upon at a moment's notice. Fireworks helps us out even further by bundling libraries of animations, buttons, bullets, and themes. For example, if you select Edit > Libraries, you can see the various classifications. Image IV shows the Bullets library.

This is the same box you saw when you just imported. Once again, if you select the bullet object you want, and click Import, that symbol will end up in the Library panel.

If you had selected Edit > Libraries > Other..., you would have been taken to the object libraries you exported a few moments ago.

## Summary

Hopefully, you can now see the power of objects as a timesaving tool. We saw how to create, name, change, export, and import objects. We also saw how to access the objects that Fireworks provides for us.

It all comes down to one very simple word: reusability. We can create an object

once, and then use it anywhere. This leads to another phrase: rapid application development. But that's for another article. ☁

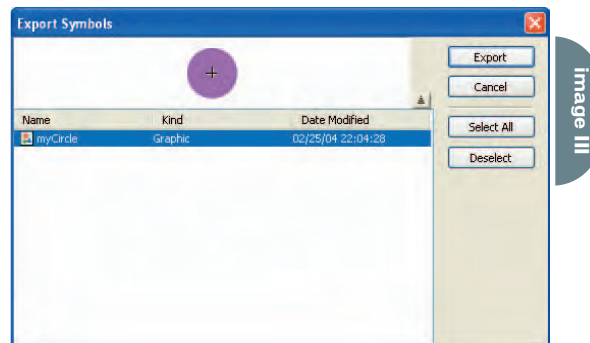


Image III

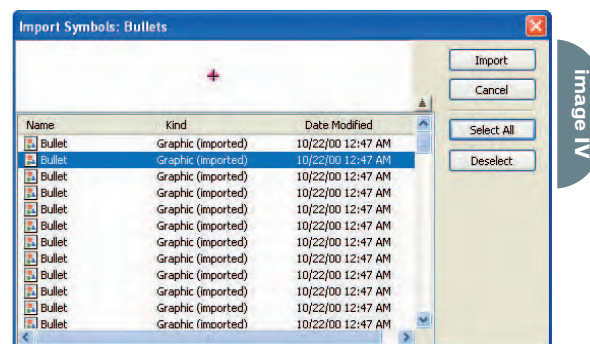


Image IV

# ARE YOU A **JIGSAW** AFICIONADO?

**YOU HAVE CHILDREN** who want to play with your programs. Or, you're a jigsaw puzzle aficionado and want to share your own work. With Fireworks, you can create a puzzle with as little as two shapes, using them to "cut" the rest of the pieces.

by joyce evans









Once the pieces are made you can get portions of an image into each piece – it really is simple! Once the puzzle is done you can bring it into Dreamweaver, where you'll add the ability for each piece to use draggable layers.

Puzzles are great fun and can be incorporated into many different kinds of sites. But keep in mind that the techniques you learn here apply to more than puzzles. For instance, the drag layer behavior can make any layer movable and can be used to drop items into a shopping cart. The techniques for making special shapes and getting images inside them works for any image you may want to place in a special shape. Image I is what the puzzle will look like when seen in a browser. Look at Image II for steps 1–4.

1. Start a new canvas and draw a 170x185 rectangle and a 65x45 ellipse.
2. ALT-drag out another copy of the ellipse. Position them on the rectangle and make each one a different color from the rectangle to make them easier to edit.
3. Use the Subselection tool to select the ellipses. Alter their shape slightly and position them with the Pointer tool.
4. SHIFT + Select all three objects and choose Modify > Combine Paths > Union.
5. Image II is the corner shape of the jigsaw puzzle. Now you need to make another puzzle shape with four "tabs" (see Image III). Use the same method as in the earlier steps to create a shape similar to this one.

## Top Row

This puzzle will have nine pieces in total. By the time you're finished you'll be able to make a puzzle with as many pieces as you want.

1. Position the corner piece in the top left corner of your canvas.
2. ALT-drag out a copy of the shape and then choose Modify > Transform > Flip Horizontal, placing this copy at the right side of your canvas.
3. To determine how wide the canvas needs to be, place the top corners of the shape with four tabs between the two corner pieces (see Image IV). Adjust the right side if necessary.
4. Choose Modify > Canvas > Fit Canvas.
5. Make sure both corner pieces are

aligned to the top of the canvas at y:0.

6. Move the center shape to the side for now and change its fill color (so you can better see what you're going to do next).
7. Draw a red rectangle to fill the space between the corner puzzle pieces (see Image V). Try lowering the opacity to see through to the tabs on the corner shapes while you're drawing. Make the rectangle 185 pixels high (the same as the base shape).
8. Put the opacity back up to 100% if you lowered it. Select the left corner shape, press SHIFT, and select the center shape.
9. Copy and paste the left-corner shape and hide the bottom copy in the Layers panel.
10. Select the right-corner shape, copy and paste it, and turn off the visibility of the bottom copy in the Layers panel.

You're making all these copies because once you use the corners as "cookie cutters" the original shape will be gone.

11. Move the red rectangle down in the Layers panel so that it sits below the gray shapes (see Image VI).
12. Select the left corner shape then SHIFT+ to select the red shape. Choose Modify > Combine Paths > Punch. As you can see, you made the shape in the side of the center puzzle piece that fits the corner shape (see Image VII).
13. Select the right-corner shape, SHIFT+select the red shape, and choose Modify > Combine Paths > Punch again.
14. Turn your corner pieces' visibility back on in the Layers panel. The top row of the puzzle is now done (see Image VIII).

## Center Row

1. Draw a rectangle that's 170x185 and make one more copy. Place one rectangle on each side of the canvas.
2. Copy each of these new rectangles and hide these copies in the Layers panel.
3. Move the center shape so that it sits above the side pieces (you may need to adjust the center piece's size).

Change the height and/or width so that it fills the center area.

4. Select the center piece and copy and paste it one time. Hide the view of the duplicate copy. Ensure that the center piece is at the top of the Layers panel (see Image IX).
5. SHIFT-select the top center piece, and each of the side pieces of the top row (see Image X). With all three pieces selected, choose Modify > Combine Paths > Punch.
6. Make a copy of both top left and right corner pieces and hide them. You're going to use the corners to punch out the tab in the top of the side pieces of the second row.
7. Turn the visibility of the four-tabbed center piece back on. Move the copies of the corner pieces and the center piece up into the Layers panel so that they're above the blue side pieces (see Image XI). (Now you can see why using different colors while building your puzzle is a good idea.)
8. SHIFT+select the left corner and the left side and choose Modify > Combine Paths > Punch. Repeat this for the right-hand side of the puzzle. Turn the visibility back on for the corner shapes on the top row.

## Bottom Row

1. Make a copy of each top corner shape and drag them down to the bottom of your canvas. Use Modify > Transform > Flip Vertical for both shapes and position them as shown in Image XII. If your canvas isn't large enough to fit all the shapes, choose Modify > Canvas > Canvas Size and make it larger.
2. Make two copies of each bottom corner shape and hide two of them. You should have three bottom corner shapes with only one visible.
3. Drag the bottom corner shapes above the blue sides so that you can see their gray tabs above the blue shapes (see Image XIII).
4. SHIFT+select one of the gray bottom corners and a left blue side shape and punch it as usual. Repeat for the other side of the puzzle.
5. Turn the visibility back on for each bottom corner piece.
6. Draw a new rectangle to fit into the remaining space. Make it bright yellow (see Image XIV).

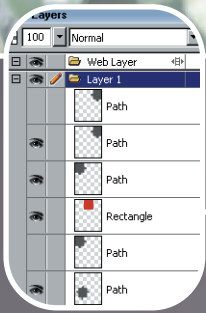


image I

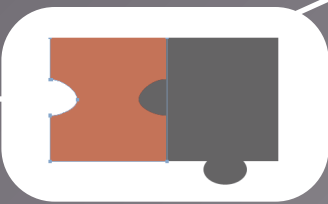


image VI

image VII



image VIII

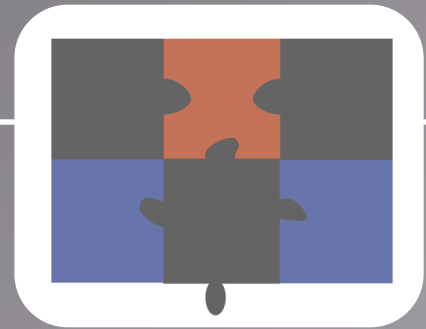


image IX

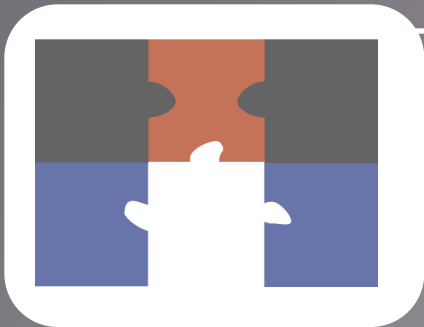


image X

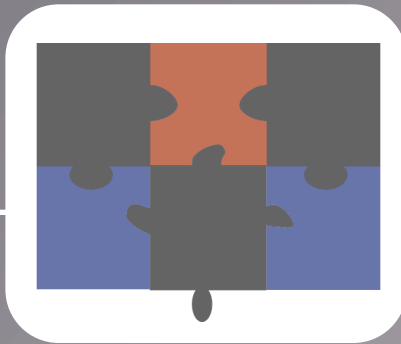


image XI

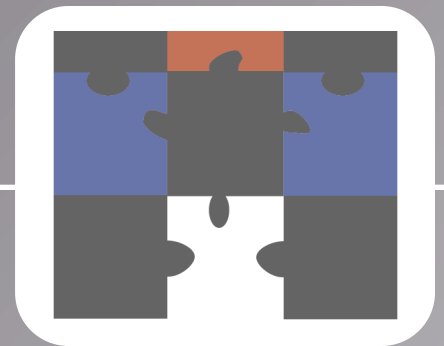


image XII

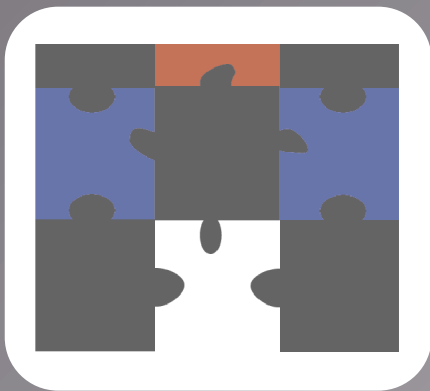


image XIII

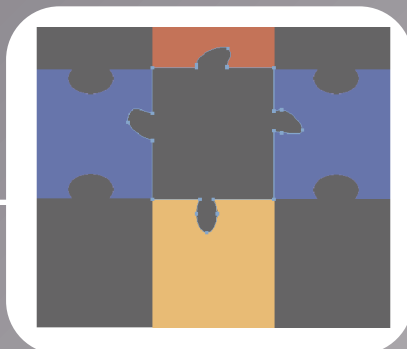


image XIV



image XV



image II

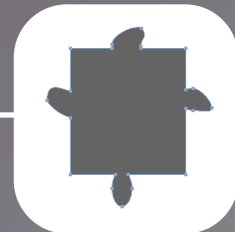


image III

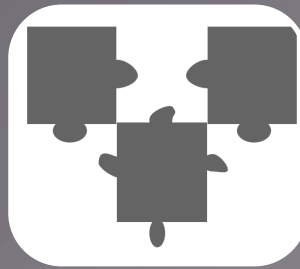


image IV

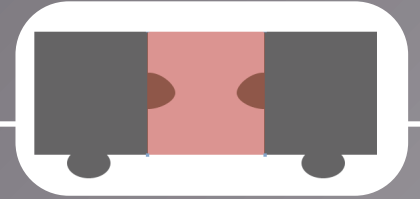


image V



7. Copy and paste the four-tabbed center piece and hide the copy. Move it up on the Layers panel so that it sits above the yellow shape.
8. SHIFT+select the center gray shape and the bottom yellow shape and Modify > Combine Paths > Punch.
9. Select the bottom corner shapes and drag them up the Layers panel so their tabs are visible above the yellow shape.
10. SHIFT+select a bottom corner and the yellow shape and punch (see Image XV). Repeat for the other side.
11. Choose Modify > Canvas > Fit Canvas. You should now have a total of nine shapes in the puzzle. If you still have extras in the Layers panel, delete them now.
12. Change the object names in the Layers panel. Working left to right and top down, name the top left corner shape p1, the top center shape p2, and so on.
13. Save the file as puzzle\_shapes.png and keep it open.

As you can see, it isn't difficult to make the puzzle pieces, it just takes a while. Remember to always make a copy of the piece that's acting as the "cookie cutter". Selecting a piece makes it easy to identify in the Layers panel so you can move the correct one above the shape.

### Adding a Picture

Let's add images to the puzzle pieces to prepare them for use in Dreamweaver. You'll add a drag layer behavior in Dreamweaver, which will allow the pieces to move.

#### Prepare the Structure

Import an image the size of your puzzle. Ideally you'd know the image size prior to making your puzzle pieces.

1. Back in your puzzle\_shapes.png document, add a new layer called Image

and drag it to the bottom of the Layers panel. Rename Layer 1 as Pieces.

2. Select the Image layer and import (CTRL+R) a photo (in my case parrot.png). Click at the top left of the document to place the image.

The current puzzle shape colors don't matter so don't bother changing them.

Each piece will be used as a mask. You now need to place each piece on its own frame.

3. Select the Pieces layer and choose Select > Select All. Open the Frames panel and from the Options pop-up menu, select Distribute to Frames. Click on a frame; you'll see that it contains one puzzle piece (see Image XVI). Each frame contains a piece except for Frame 1, which is the image (see Image XVII).
4. You want the photo to be visible in every frame so you'll need to set it to be shared by all layers. Double-click the Image layer name and check Share across frames (see Image XVIII). Click OK in the warning dialog that pops up.
5. Double-click the Image layer again and deselect the Share this Layer option. Another dialog opens. Basically, instead of sharing the layer the image will now be copied into each frame automatically. Click on All. What you just did was copy the image to every frame instead of simply sharing the same image. You'll need an actual copy in every frame.

*Note:* Another way to get the image placed precisely in all frames is to select it and choose Copy to Frames > All from the Frames panel options menu.

6. You can now select and delete Frame 1. The frames will renumber automatically. All you need now are the frames with the puzzle pieces in them.
7. Save your file. A copy will be saved as puzzle2.png in the download folder.

### Putting the Image into the Pieces

Now you need to get the part of the image below each piece into the puzzle piece. It sounds tough, but wait until you see how easy it is!

Image XIX corresponds to steps 1–3 below, but the cloverleaf is cropped out, indicating that it is masked.

1. Select Frame 1; select the puzzle piece and cut it (CTRL+X).
2. Select the parrot image and choose Edit > Paste As Mask. Your piece may look different than mine depending on how your pieces were distributed to the frames.
3. To make it look more like a puzzle piece select Bevel and Emboss > Inner Bevel from the Effects menu. Change the width to 3.

You'll repeat steps 1–3 for each puzzle piece, so to save on the repetition, let's make a command. It'll save you repeating the previous three steps for each piece.

4. Open the History panel. Select the last two operations (which should be Paste and Set Effects). Notice that the Cut operation is separated between black lines, indicating that it cannot be added to the command.
5. Select the Save As command from the History panel's Options menu and name this command puzzleMask.
6. Open the Frames panel and select Frame 2 (see Image XXII).
7. Cut the puzzle piece in Frame 2 and then select the image.
8. Choose puzzleMask from the bottom of the Commands menu.
9. Repeat steps 7 and 8 for the remaining frames.

### Background Image

You now have nine frames, each containing a puzzle piece. You'll need to add

---

**“It isn't difficult to make the puzzle pieces, it just takes a while”**

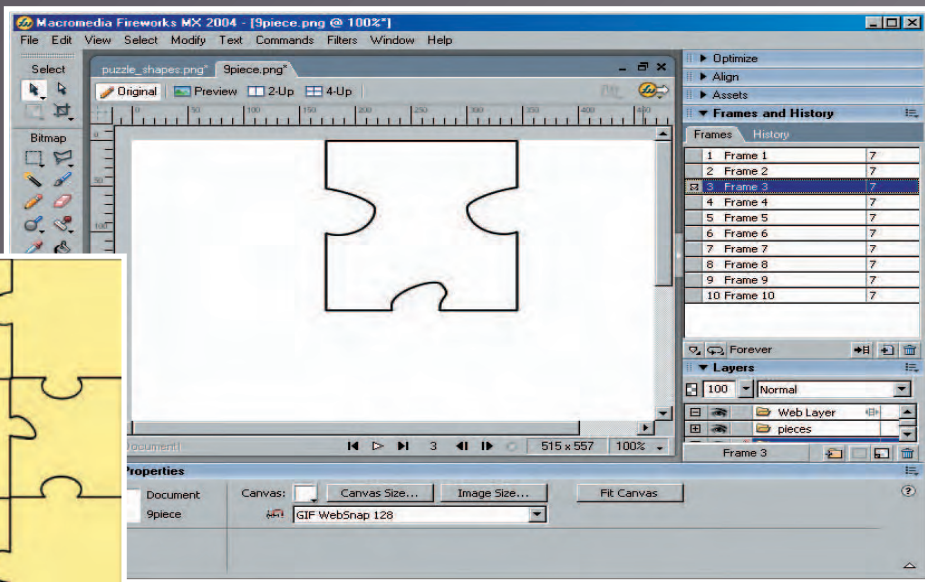


image XVI

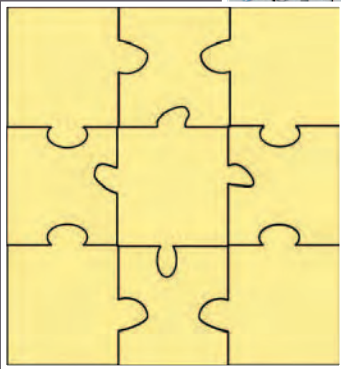


image XVII

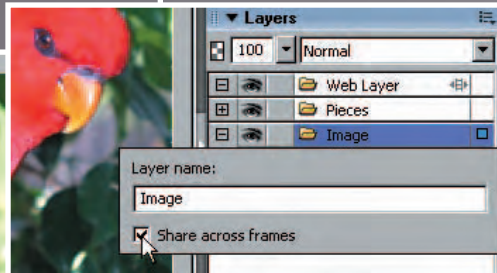


image XVIII

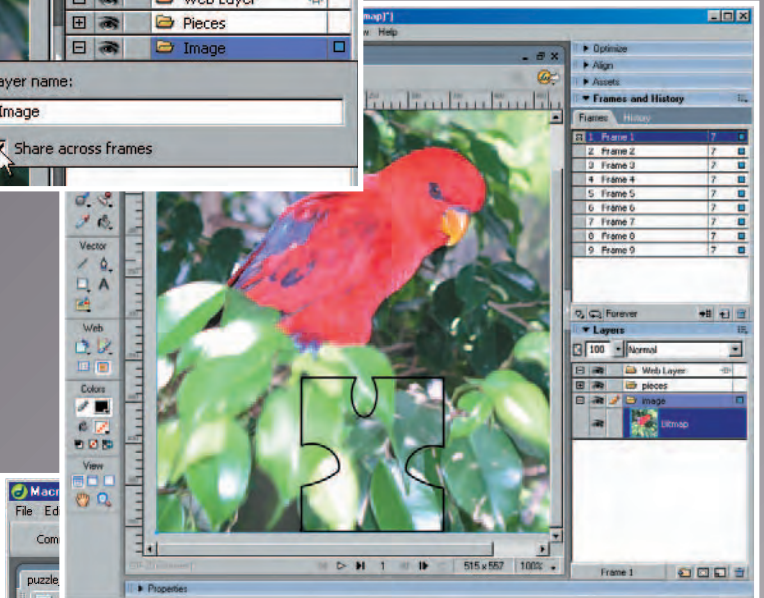


image XIX

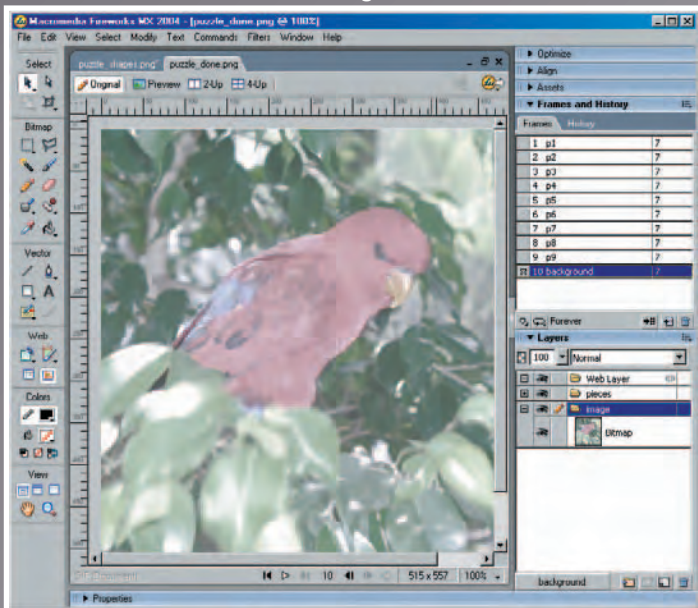


image XX

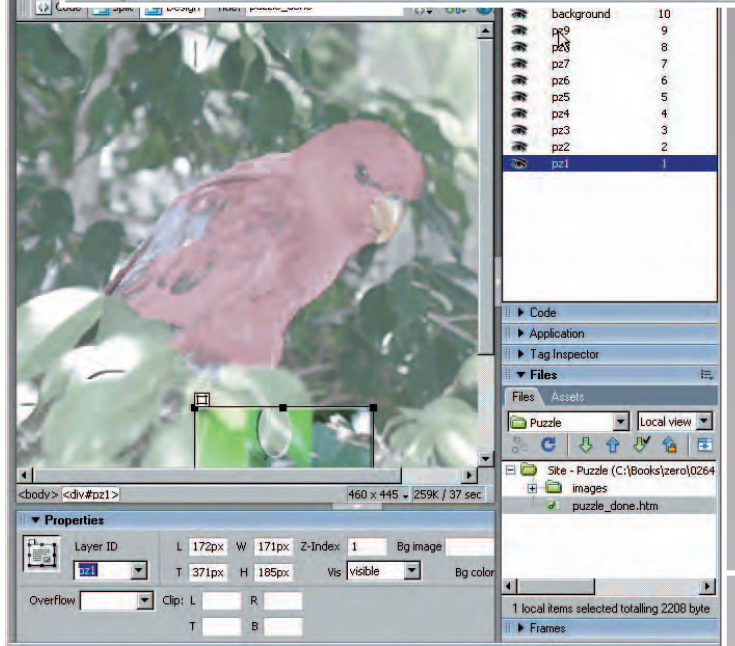


image XXI



the background image in again. The background will be the image the user uses as a guide for putting the puzzle together.

1. Select the last frame in the Frames panel and choose Duplicate Frame from the Options menu. Check After the current frame and click OK.
2. In frame 10 (the new duplicate) select the puzzle piece and choose Modify > Ungroup. Deselect by clicking off the document then select the puzzle piece and delete it. You'll be left with just the image.
3. Select the image and choose Adjust Color > Hue/Saturation from the Effects menu. Change the saturation and lightness as needed.
4. In the Effects area you may see the inner bevel effect on the image, select it and click the minus (-) sign.
5. Double-click Frame 1 and name it "p1". Repeat for all 9 frames except number 10 the last one. Name it "background" (see Image XX). Save your file.

### Exporting as CSS Layers

Let's export the frames as CSS layers. Fireworks will export the individual frames as individual CSS layers and do part of the work for you in order to make finishing the puzzle in Dreamweaver faster. Fireworks will make the HTML page automatically.

1. Choose Modify > Canvas > Canvas Color and make it transparent (see Image XXI).
2. In the Optimize panel, select GIF as the export file format and set the Colors to 128. Choose Index Transparency and set the Matte to none.
3. Choose File > Export. In the Export dialog box, navigate to your site folder. In the Save as type field, select CSS Layers from the drop-down menu and Fireworks Frames from the Source drop-down menu. Check Put images in subfolder so that Fireworks will automatically add a separate images folder. Also check Trim images and then click Save to export.

The trim image option will cut away the extra masked area of the entire image and leave you with only the puzzle piece.

### Powering the Puzzle in Dreamweaver

If you don't already have Dreamweaver installed, you can get a 30-day trial from Macromedia ([www.macromedia.com/downloads](http://www.macromedia.com/downloads)). You'll need to register if you haven't before – but it's free.

You'll be adding the Drag Layer behavior to make your puzzle pieces move. The Drag Layer behavior in Dreamweaver allows you to drag layers within an HTML page. There are a lot of possibilities built into the behavior. You can choose to constrain where the layer is dragged to, and by how much, and can even have it snap into place when the piece gets close to the area it belongs in.

1. Open Dreamweaver. Choose > Site > Manage Sites > New > Site. Name the site "Puzzle". Click the folder next to Local Root folder and navigate to your puzzle folder. Select it and open.
2. Open the Files panel (F8). Double click on the puzzle\_done.htm file to open it (this is the file that Fireworks generated). It contains all the puzzle pieces in it even though you can't see them.
3. Open the Design (panel groups on the right) panel and click on the Layers tab to open the Layers panel (see Image XXII). Select a puzzle piece and notice that it's in its proper position. Using the Layers panel is going to be the only practical way to select these layers.

Notice in the Layers panel that the name you gave to each frame in Fireworks has now become the layer name in Dreamweaver (see Image XXIII). A quirk of Fireworks is that it gave the layers the name of the image. If you select one of the layers and look in the Property inspector, notice that the Image name is the same as the layer name. This won't work because layers must have unique IDs.

4. Select the first numbered layer. In the Property inspector, delete the name and change it to "pz1" and press Enter/Return. Repeat this for all nine puzzle pieces giving them unique names.
5. Select the <body> tag in the Tag Selector. The Drag Layer behavior will be attached to the body tag using an onLoad event.

6. Click on the Tag Inspector panel group and open the Behaviors panel. Click the Add (+) button and select Drag Layer from the list (see Image XXIV).
7. The Drag Layer dialog box opens (see Image XXV). Select pz1 from the Layer drop-down menu. Notice that all the layers in the document are listed.
8. Leave Unconstrained checked. Click the Get Current Position button. These are the coordinates that will be used to determine whether the piece is dragged to the correct position.
9. Set the Snap if within field to 15 pixels. When the user gets to within 15 pixels, the piece will snap into place.
10. Close the dialog box and repeat steps 8–11 for every puzzle piece (but not the background).
11. Save your file. Now select a puzzle piece in the Layers panel and click and drag on the white square handle. Drag the piece to the right side. Mix them up and stack them. When you select a piece that is at the top of the document you won't see the white square handle you drag with (see Image XXVI). Use the keyboard arrow keys to lower it. Then you can drag it over to the side.
12. Save and preview. Test out your puzzle. Drag each piece into place to be sure you didn't forget to add the Drag Layer behavior to one of them (see Image XXVII).

And voila, you're done! ☺

*Joyce J. Evans is a training veteran with over 10 years of experience in educational teaching, tutorial development, and Web design. She has presented at conferences such as Macromedia MAX 2003 and TODCON. Joyce has authored books including Macromedia Studio MX 2004 Bible, Dreamweaver MX 2004 Complete Course, and others. Joyce is a Team Macromedia volunteer and her work is also featured in the Macromedia Design/Developer center and the Macromedia Edge newsletter. Her Web sites are [www.JoyceJEvans.com](http://www.JoyceJEvans.com) and [Idea Design \(www.je-ideadesign.com\)](http://Idea Design (www.je-ideadesign.com)). [Joyce@JoyceJEvans.com](mailto:Joyce@JoyceJEvans.com)*

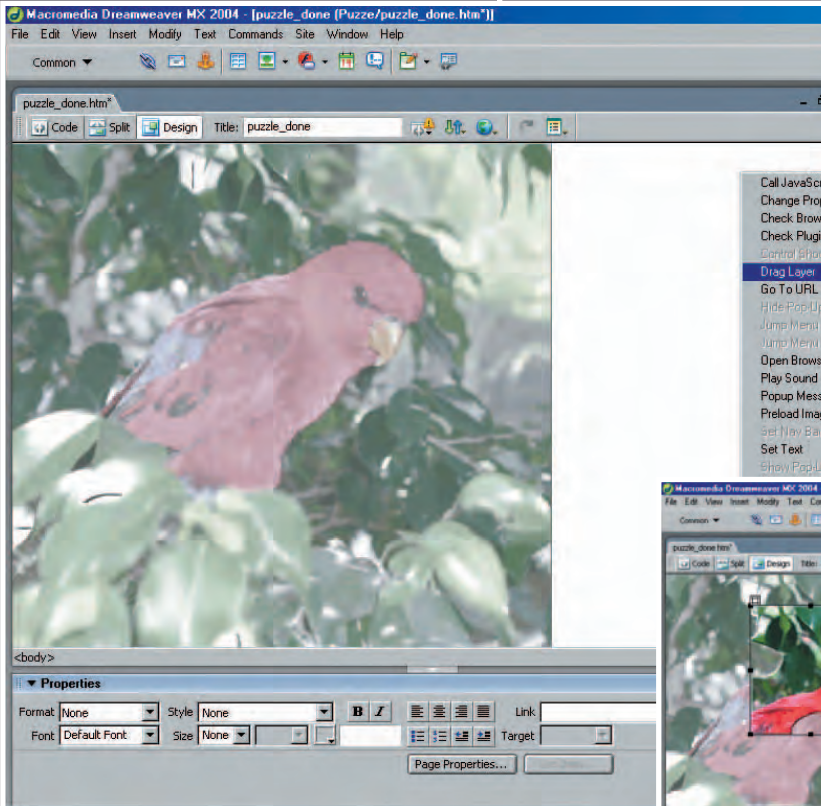


image XXIV

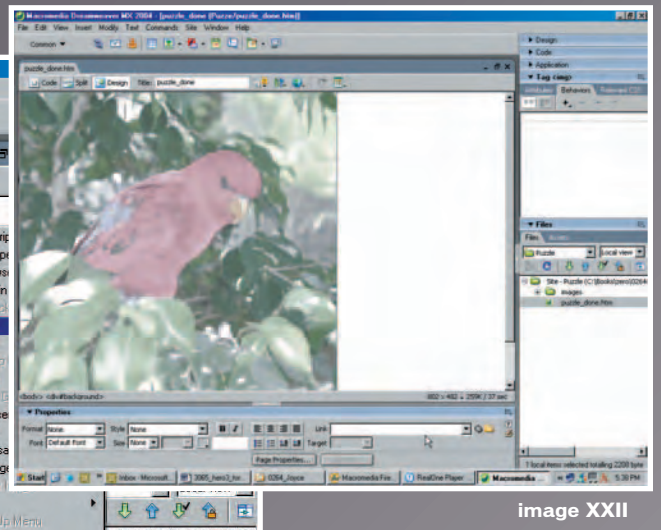


image XXII

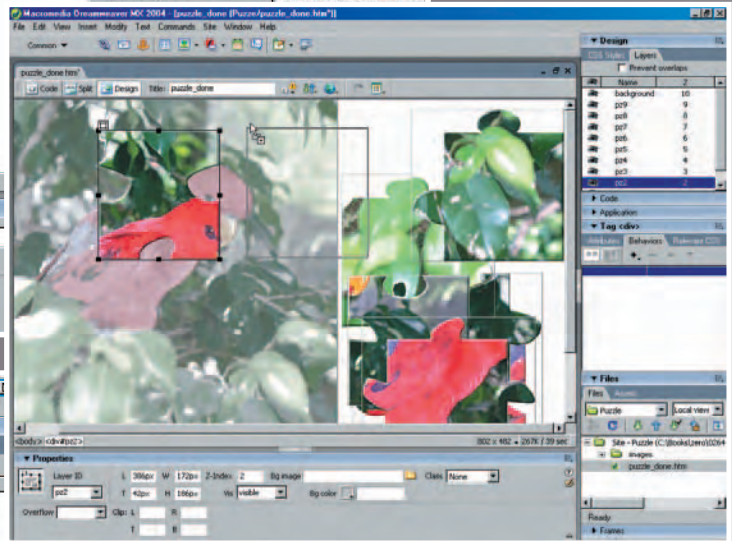


image XXVI

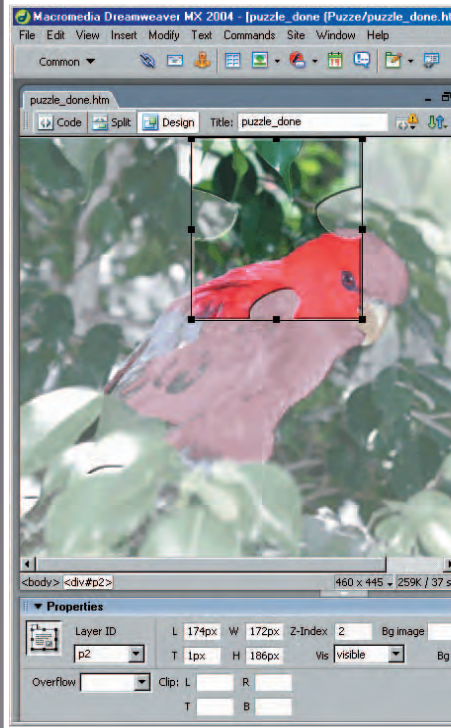


image XXIII



image XXVII

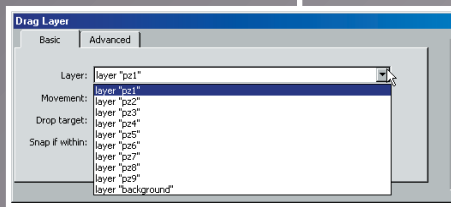
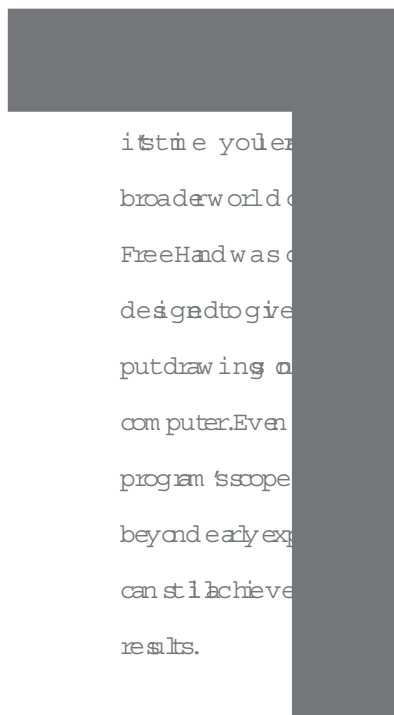


image XXV





# Bullseye of Printing



and MX  
s, then  
the  
broad world of  
FreeHand was  
designed to give a way to  
put drawing on a  
computer. Even  
program's scope  
beyond early exper  
can still achieve printing  
results.



by ron rockwell



## PostScript Printing

The most important thing to remember about FreeHand is that it requires a PostScript printer to achieve acceptable results. Yes, you can print to a desktop non-PostScript inkjet printer, but the output is sometimes less than what was expected.

PostScript is a page description language introduced by Adobe in 1985. Its application on the Apple LaserWriter brought Macintosh computers fully into the front-runner position in the growing desktop publishing field. A PostScript file interprets or describes text and images in a device-independent fashion. Device independence means that the file can be used on any PostScript printer without regard to printer resolution or other printer attributes. The same file will print to the capacity of your PostScript desktop printer or the service bureau's RIP (Raster Image Processor).

## Non-PostScript Printing

Many factors determine what is going to print and what won't print from a particular file on a given day to one printer or another. To get good results from imported images, set Preferences > Redraw > Better – but set slower display, and Onscreen Resolution to Full. For the most part, you'll lose certain effects or features that you may have fallen in love

with on screen.

Some of those items may be very important to your work. For instance, a placed EPS file (EPS stands for Encapsulated PostScript – that's your first clue) will print only the on-screen preview, not the actual file. That means a jaggy, totally unacceptable print. Then, add a bitmap effect – any bitmap effect – and you'll see more jaggies. However, text effects (Highlight, Inline, Strikethrough, Shadow, Underline, and Zoom) do print as you see them on the monitor, but will not output correctly to a high-end RIP at the service bureau. For that reason alone you should get in the habit of creating your own special effects on text if you plan on doing commercial printing.

Without special third-party software, you cannot print "printer marks" on most non-PostScript printers either. Printer marks are needed by professional printers to properly set ink levels, crop or trim the page, and provide information about the color of ink that is to be printed.

Separations is the term given to the result of creating negatives for each color that is going to be printed on the page. It's good practice – and required by most service bureaus or printers – to provide a complete set of separations with the finished job. Printing your own separations is necessary for you to proof the work. Non-PostScript printers usually don't allow you to print separations without extra software. Since we're after professional results, the rest of this discussion will assume the use of PostScript printers.

## Printer Resolution

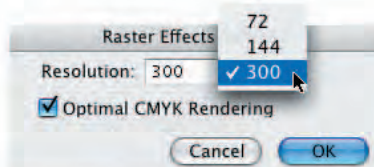
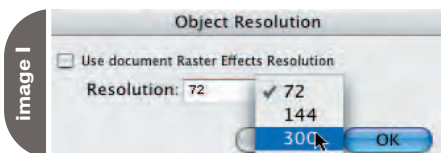
We get used to seeing our artwork on a monitor and too frequently forget to set the proper parameters for the document to be output correctly. FreeHand MX tossed a few new features into the

parameter soup to digest, starting with raster effects. For spice, add a pinch or two of printer resolution, raster effects resolution, and document raster settings.

Printer resolution has been around for years, and is set in the Document window of the Properties panel. The default is 300 dpi (dots per inch – the number of halftone dots per inch that are printed on the page), but you have several other choices in the drop-down menu. Printer resolution uses this information to generate the number of steps in auto-created objects such as blends and gradient fills. The higher the printer resolution setting, the more steps in the blend or fill. To see this in action, choose a low number, such as 300, and create a blend between two different colored objects. Note the number of steps in the blend. Then raise the resolution to 2540 and blend the same two shapes. You'll see many more steps in the blend. A high number will give you a smoother blend effect, but it also complicates the file. That equals nonproductive waiting. It's important to remember that you can change the number of steps in a blend at any time. Just enter a new number in the Number of Steps box in the Object panel while you have a blended shape selected.

When you apply any of the raster effects (shadows, blurs, glows, embossing) to an object, steps must be taken to ensure that you will achieve the desired result when you go to print or press. Select an object with a raster effect and open the Options menu in the Object panel. Midway down the list of options you'll find Raster Effects Resolution. Select this option and the Object Resolution dialog box opens (see Image I). The default setting is 72 dpi, which is fine for Web use, but not at all acceptable for print work. Select 72 dpi from the drop-down menu while you're working on the project and FreeHand will react more quickly. When you're ready to go to press, change the resolution to 300. Notice the "Use Document Raster Effects Resolution" option. Choosing this option overrides any setting that appears in the Resolution field. If you set this option to "on" in your default page, then all objects with raster effects will use the Document Raster Effects you set.

So, where do you set Document Raster Effects? Go to File > Document



Settings>Raster Effects Settings and a box very similar to the Raster Effects Resolution box opens (see Image II). If you're printing in CMYK (and only if all your colors are CMYK), click the "Optimal CMYK Rendering" box. From that point on, any objects with raster effects will have adequate resolution. You can override any effects you wish by dropping back to the object level described earlier. Again, your work will go faster with a 72-dpi setting (although the image may look a little coarse), but remember to switch to 300-dpi resolution as you go to press. Your service bureau will not make this adjustment for you – it's your responsibility.

### Why All the Fuss?

In short, if you don't deal with resolution while you're working on the job, you'll deal with extra printing costs to have your job printed again. Do a quick test on your own by creating several objects with various raster effects applied. Give them a resolution of 72 dpi. Clone them and raise the resolution to 144; then clone again and change the resolution to 300. At 100% size on screen, they look pretty much the same, but print the page and examine the results. It's really important to realize that all the raster effects are RGB, not CMYK. If you're using spot colors, those colors are converted to RGB for screen display. Again, they look great on screen, but when it comes to printing them, the conversion from RGB to CMYK is often less than pleasing, so proof your work often and critically. If it doesn't look right when it comes off of your desktop PostScript printer, it won't look any better coming off the press. For best results you will also want to "set on-screen image resolution" to "Full" when printing documents with Raster Effects.

### We're Caught in a Trap

You probably know that if you place one colored object on top of another in FreeHand the colors are not mixed or multiplied as they are with traditional paints. Instead, FreeHand "knocks out" the area of overlap and places pure colors in the two shapes. When two colors abut on-screen, you see no space between them because you're looking at adjacent pixels that are fixed on the mon-

itor. Getting those two colors to touch exactly on paper is quite another story. You usually end up with a sliver of white or a dark color that is the combination of the two colors.

To minimize slight misregistration, you must create traps. Check with your service bureau or printer before proceeding – some have special trapping software that does the job for us, and to their exact specifications. Traps are basically the minor overlaps of colors that eliminate possible white spaces where the colors are supposed to abut. To create a trap, you either choke or spread one color onto another by printing a tint of the lighter color on top of the darker color. In Image III you can see the results of trapping. You can also see tiny "o" shapes in the overlapping areas. If you go to FreeHand Preferences>Redraw and select Display Overprinting Objects, you'll acquire this on-screen verification that overprinting is going on. At times it can be visually annoying, but if you're doing any trapping it's absolutely essential that you see what you're doing. Be advised that basic text colored black is set to overprint. With this option checked, the text will be filled with "o" shapes if you convert the text to paths. The "o" shapes will not print.

Trapping can be done easily by using the Trap Xtra. Just select the inner object, Shift-select the outer object, and choose Xtras>Create>Trap (or click the Trap icon if you've placed it in your main toolbar). A dialog box opens (see Image IV) where you can input the size and type of trap you want. The size of the trap depends entirely on your printer's specifications, usually a quarter-point to half-point trap is sufficient for offset printing – screen printing needs a bit more – check with your printer before setting the traps. The Reverse Traps option will make the overprinting color dark instead of light.

To create a trap manually, simply select the lightest-colored object and click the Add Stroke icon in the Object panel. Make the stroke equal to double the width you want to use as the trap. Give the stroke a suitable (around 40%) tint of the light color, and set the stroke to overprint. The center portion of the light-colored object will correctly knock out of the darker color and the outside half of the stroke will overprint the dark

color, eliminating any white space caused by misregistration.

### Spot or CMYK Colors?

As you design a print project, certain decisions must be made concerning ink colors. FreeHand allows you to create colors in the Color Mixer panel in many color modes, including RGB and CMYK. If you will be printing the project, avoid creating colors in the RGB or HLS mode. Those color modes are for on-screen viewing and must be converted to CMYK in the printing process. Due to the limitations of the process, or CMYK, method of printing, color conversion may work well or not. Certain RGB colors will not give pleasing results, and most colors will show a slight or noticeable color shift. If a specific color is desired or required, then you should consider using a spot color. Spot colors are specific ink colors from ink manufacturers; you can find several libraries of them in the Swatches options menu. The ink color will be interpreted and displayed on your monitor in RGB, and will be marked with three dots (red, green, blue – RGB) in the Swatches list, indicating that it is a spot color. Don't be lulled into the beauty of any color you see on screen, as the calibration of your monitor along with the RGB interpretation may have no bearing on the true color. Instead, invest in, and learn to trust, a printed color swatch book. PANTONE makes a "solid to process" book that is invaluable for process printing. It contains hundreds of solid colors printed next to their CMYK breakdown so you can see what the color conversions will produce.

### Collect for Output

FreeHand makes a heroic attempt to help you produce a trouble-free print job with the Collect for Output command. When you select this feature from the File menu, you will be asked to save the document before continuing. A dialog box will open (see Image V), giving you the options of adding various types of information about the document in a simple text-only file that your commercial printer or service bureau personnel can open and read. Each category has a list of options; holding down the Option/Alt key as you click any of the options will either deselect the entire list, or select all



the options in the list. Unless you only want to see certain information for yourself – to get a list of fonts, for instance – leave all the options checked for the report that you give the service provider. Close the box by clicking Report.

A Save window will open, giving you the opportunity to place the report and all of the document's data in a particular location. This data includes all the fonts and placed images and EPS files you have used in the document. That can amount to a large number of files, so it's a good idea to create a new folder for everything. The Save As name refers to the document record itself. When you've found a place for the files and named the record, click Save. FreeHand will then col-

image III



image IV

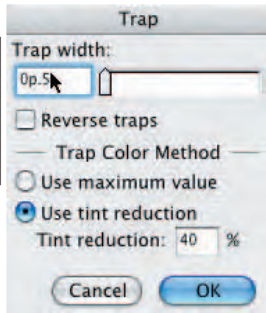


image V

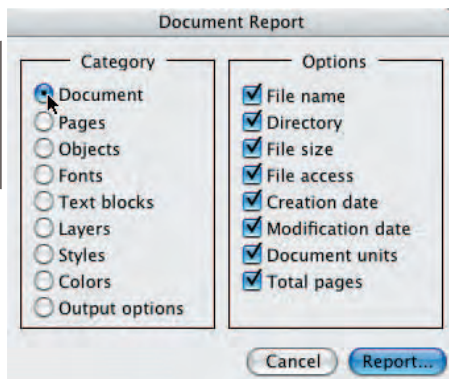
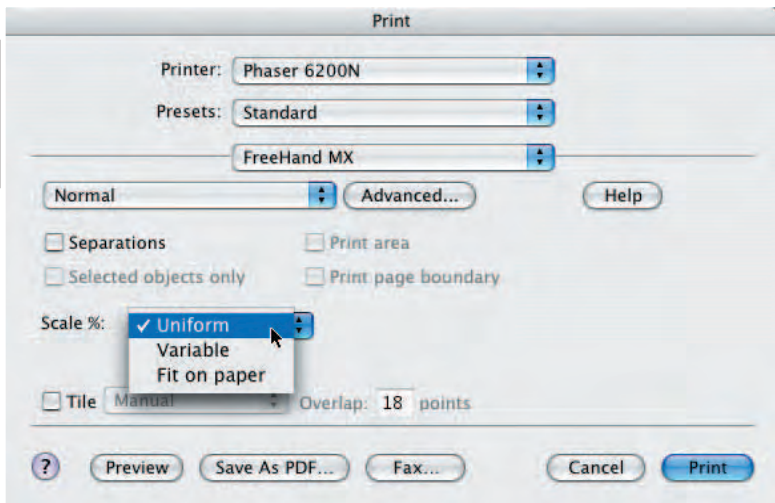


image VI



lect everything that pertains to the job for you. If placed files or fonts have been moved or there are other problems with the files, FreeHand will tell you that there were problems collecting the data. If that happens, look through the new folder to try to see what's missing. Usually it's a linked file that you've worked on and inadvertently moved. If you have several placed bitmap images or EPS files, open Edit>Links to see a list of all the files in your document. Compare that list to the output folder and then manually gather the missing files. Alternatively, you can relink the files and Collect for Output again.

### Test Prints and Separations

As you begin to wrap up your job, it's necessary to proof-print the job. Doing so will eliminate 99% of the problems your service bureau or commercial printer will encounter. It tells you how well you've done your job, and ultimately saves you money with do-overs. You need to print both a composite print and a set of separations.

Choose File>Print Setup, and select your printer (assuming

you've got a PostScript printer) and the page orientation. Then choose File>Print. You can print a single page or a contiguous page range by entering numbers in the fields. Then select FreeHand MX from the print options drop-down menu (see Image VI).

If your document is smaller than the paper that you are going to print on, then choose Uniform from the Scale % drop-down menu. If you are printing a letter-sized document onto a letter-sized sheet of paper, there will be no room for printer marks, so you will be forced to choose Fit On Paper and print a reduced version of the job. It's a proof, so don't lose too much sleep over it. Click the Advanced button to open the Print Setup window (see Image VII).

Print Setup is really the heart of the printing operation. The left side of the window is dominated by the Preview window. I keep mine set to Preview, but Keyline and X-Box are also available views. It's a little-used feature, but you can click and drag the preview image in its window. Only the parts of elements that show in the window will print – good for test-printing partial pages without using the Area Tool. Start by choosing the level of quality in the Print Setting menu – use Quality PS Level 2 or Normal for most jobs. Select the Use PPD option to search for the correct PPD (PostScript Printer Description) for your printer. In the Separations tab, you can choose to print a composite page or sep-

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

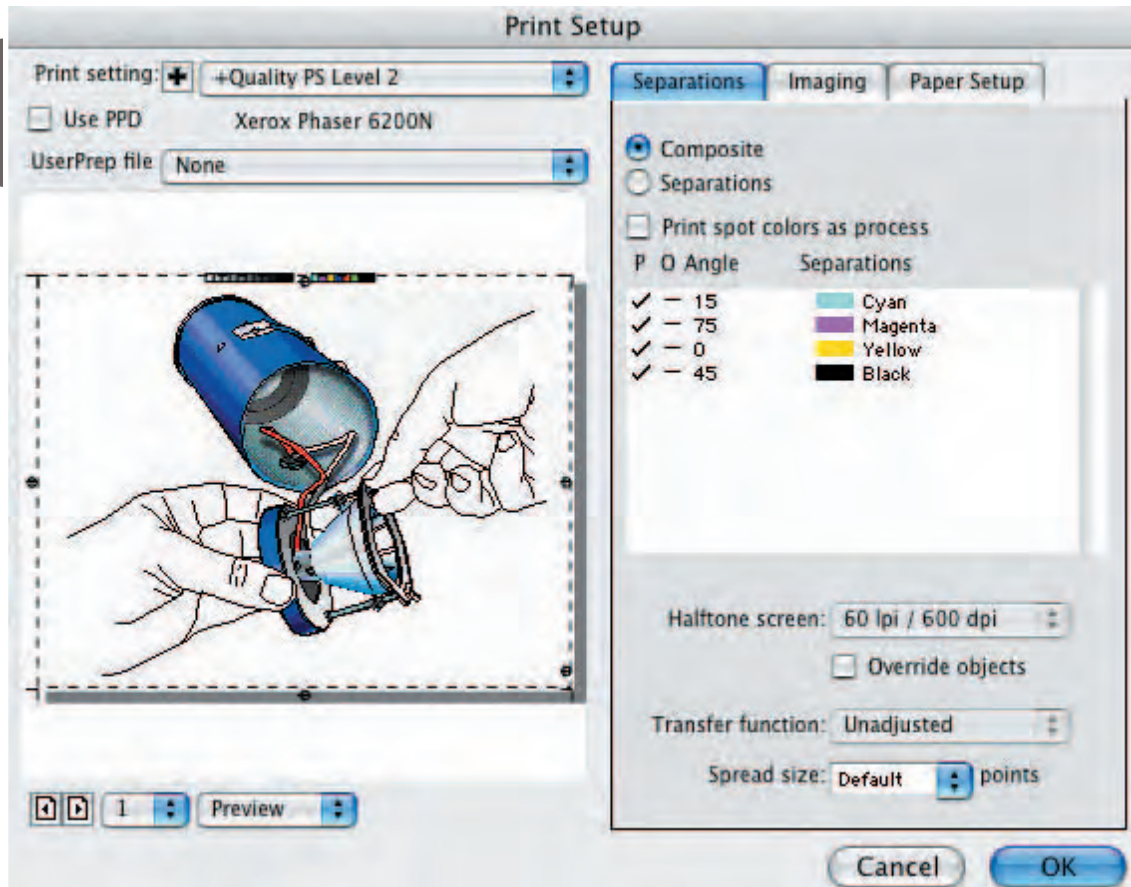


**COLD FUSION** Developer's  
Journal





Image VII



Illustrator, designer, author, and Team Macromedia member and MXDJ FreeHand editor 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 coauthored Studio MX Bible and the Digital Photography Bible. He has Web sites at [www.nidus-corp.com](http://www.nidus-corp.com) and [www.brainstormer.org](http://www.brainstormer.org). [guru@brainstormer.org](mailto:guru@brainstormer.org)

arations. By selecting the Print Spot Colors As Process option, you convert the document's spot colors to CMYK for the purpose of printing. Doing so will usually cause some sort of color shift, so pay attention to the composite print. The separations window has three columns on the left: P (a check mark will Print the color, click the check to prevent the printer from printing that color), O (for Overprint – this causes the color to overprint all other colors in the document – it's only advised in rare instances, be wary), and Angle (the angle of the rows

of dots that make up the screens in your image – C=15, M=75, Y=0, K=45 to prevent most moirés). The other field's defaults are usually adequate.

Click the Imaging tab (shown on the left side of Image VIII). Here you can choose to print labels and marks or not. FreeHand will place crop marks that match your page size if you check that option. If you've designed a business card in the middle of a letter-sized page, the crops will be at 8.5 x 11 inches, not 2 x 3.5 inches. If you want FreeHand to apply crop marks, make the document exactly the size you want it to print. Registration marks, separation names, and file name and date are all important information that you should use, and they're free – use them. To proof your separations (and save a lot of black toner), choose Emulsion Up, Positive Image. A film output is usually Emulsion Down, Negative Image, termed RRED (Right-Reading Emulsion Down). The right side of Image VII shows the Paper Setup tab. Make sure the page orientation matches your Page Setup orientation. Otherwise, this panel should reflect your document's settings. Click the OK button to close the window.

Click the Print button.

Depending on the complexity of your project, FreeHand will whirr for a few seconds before your separations come out of the printer. The default settings will give you a positive image of each color separation. Check each separation page against a composite print so you can see where overprinting and knockouts appear. Look for anything out of the ordinary. If something is wrong, this is the time to correct it – don't blindly hope the service bureau will fix your errors. No one knows your job as well as you do.

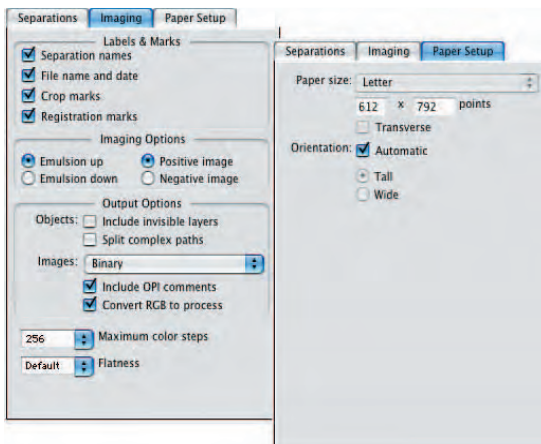
## Summary

FreeHand is a very powerful application; with it you can print just about anything from business cards to multi-page catalogs to packaging. Communication with your printer or service bureau will speed up your learning curve and profitability.

## Acknowledgments

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

Image VIII



# FREE\* CD! (\$198.00 VALUE!)

## *Secrets of the ColdFusion Masters*

Every *CFDJ* Article on One CD!



### — The Complete Works —

CD is edited by *CFDJ* Editor-in-Chief Robert Diamond and organized into 23 chapters containing more than 450 exclusive *CFDJ* articles!

All in an easy-to-navigate HTML format! **BONUS: Full source code included!**

**ORDER AT [WWW.SYS-CON.COM/FREECD](http://WWW.SYS-CON.COM/FREECD)**

*\*PLUS \$9.95 SHIPPING AND PROCESSING (U.S. ONLY)*

©COPYRIGHT 2004 SYS-CON MEDIA. WHILE SUPPLIES LAST. OFFER SUBJECT TO CHANGE WITHOUT NOTICE. ALL BRAND AND PRODUCT NAMES ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS OF THEIR RESPECTIVE COMPANIES.



*Only from the World's Leading i-Technology Publisher*



# Freaks and Geeks Unite

PART 1

*Getting dynamic requires designers and coders to work together*

by tom green

**T**his past summer I decided that my courseware site needed a complete overhaul. It was, in fact, a bit of an embarrassment. As a teacher, writer, and lecturer I had been sticking my courseware and lectures notes up on a site that was designed more for convenience than anything else. After quietly suffering through the jabs of my students and my colleagues, I decided the time had arrived to bring order to the chaos.

In August 2003, I sat down with a very talented Toronto graphic designer, Shawn Butchart, and asked him, as I succinctly put it, “to give my site an enema.” After a lot of the usual discussion we settled on a rather interesting design and I went to work exploring the joys and the frustrations of the world of CSS-Positioning, or CSS-p. When the redesign went live in November, it met with the approval of my students and peers and, as is normal, I thought I would live with the design for the next couple of years (see Image I). Then the wheels started to wobble.

One-third of the site was devoted to a series of tutorials and Studio MX-related articles I have written. They can be found at [www.tomontheweb2.ca/MXtras/Tips\\_techniques.htm](http://www.tomontheweb2.ca/MXtras/Tips_techniques.htm).

Image II shows a redesigned page – this area will grow, and I hadn’t incorporated this rather important fact into the design of the site. In November of 2003 there were about a dozen tutorials. Today, a few months later, the number is approaching 30 and will most likely crack the 50 mark by the end of this year. Rather than solve a problem, I had created a bigger one.

The problem had two aspects to it. The first was the entry page listing the tutorials. There was no real organization to it. I would prepare a tutorial for the students and tack its description to the bottom of the list. This is workable if you

have only a handful of tutorials. It falls apart when the number grows. Trying to find a specific tutorial, “Color Correction in Fireworks,” for example, requires the user to hunt through the page. I needed to rethink this page and make it more accessible to my visitors.

The second problem was the actual process of adding a tutorial. Though I was using a “template” the actual creation of the page was a time-consuming process. The text would have to be flowed into the page; the images would have to be added; the links, if there were any, would get tossed in; and the page would be saved to the directory. The listing page would then be opened, the description and link created, and then it would be uploaded to the server. It was inefficient and, if allowed to continue, would become an administrative and site management horror show.

The time had arrived for me, to borrow a phrase from my latest book, “to get dynamic or get dead.”

This two-part article will take you through how I did it. It most likely won’t be pretty because Web design based upon a dynamic paradigm is an inexact science. Planning data, data flow, database choice, middleware choice, and so on are all subjective, not objective, decisions. The whole field of dynamic Web site creation is relatively new. It’s only over the past few years that dynamic sites have moved into the mainstream of the industry, and the introduction of Studio MX 2004 and Macromedia’s embracing of “rich Internet applications” only served to accelerate acceptance of this discipline.

I didn’t try to do it by myself. My area of specialization – how Studio MX tools work together to provide a comprehensive workflow solution – is already quite large. Early in the game I made a con-

scious decision to understand how databases and ColdFusion work and interact with the tools in Studio MX. Still, rolling up my sleeves, digging into the code, and getting my arms “dirty” with electrons was something I chose not to do. Ours is an industry where teams of specialists bring their unique skills to bear on the task at hand. It is an industry where two polar opposites – geeks and freaks – have to learn to not only work intimately with each other, but also to speak each other’s unique language.

Designers, to coin a phrase from the ‘60s, “the freaks,” simply can’t be expected to be coders. They aren’t hardwired into logic. Coders, “the geeks,” live in a sequential and logical universe. This is not to belittle or stereotype anyone. It is simply describing how the two groups think and approach the problem-solving process.

If a freak wants to get a cup of coffee he or she will point in the general direction of the café and say, “I am going over there for a cup of coffee.” The geek will map out the route with a focus on the most efficient and logical method of arriving at the café. In the dynamic way of doing things, the most successful organizations are those in which the geeks and the freaks harmonize their unique skill sets to meet clients’ needs.

This brings me to James Cullin – James is a geek. I have known James for more than a decade and, next to my book’s coauthor Jordan Chilcott, he is one of the best coders I have encountered. More important, like Jordan, James understands the creative process intimately.

James and I have worked together for more than a decade teaching digital media technologies at our college’s school of media studies. Our two programs – interactive multimedia and



Internet management – complement each other, and three years ago we came to the conclusion that his Internet management students needed to know what I do and that my multimedia students needed to know what he does. This wasn't done for political or turf reasons. It was more pragmatic than that. Our two groups were graduating into an industry in which they would eventually have to work closely with each other, and it only made sense that they learn how to communicate and work together before they graduated.

### Getting Out of the Box

When I discovered I had boxed myself in with my new design, I started looking at how to get out of the box. Over Christmas, I came to the conclusion that the only solution was to make the tutorials section dynamic. When I returned to work in January of this year, I plunked myself down in a chair in James's office and said, "I have a problem and need your help to solve it."

As is so typical of James, he listened to my problem, asked a few very pointed

image 1



image 2

### Tips and techniques

Here are a series of extra techniques for you to explore and try out on your own. The key to all of these is to adapt them for your own purposes. Enjoy.

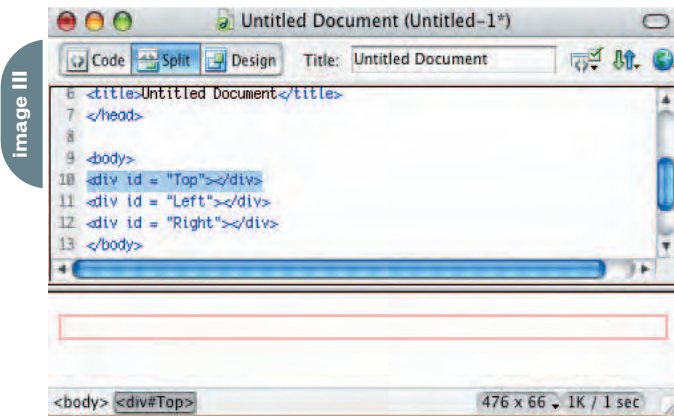
1. **Fireworks Basics:** Still unclear as to how to create stuff using Fireworks. Here is a set of tutorials - from Halifax developer and Fireworks expert, Kleantlis Economou- that should get you going.
2. **Panning Images:** See how to code an image that can then be panned.
3. **Zooming Images:** Learn how to write the code that zooms in on an image which can then be panned .
4. **Slide Show:** Create a slide show using a folder full of images....not the cast
5. **Keyframes in Flash MX:** A tutorial that covers off all of the basics of working with keyframes on the Flash timeline.
6. **Fading out Movie Clips in Flash MX:** A tutorial that shows how to have a movie clip fade out on the Flash timeline. A great transition effect.
7. **Vanish:** How to recreate that old TV effect of an image shriking into a single point of light.
8. **Shockwave Applet:** Shockwave isn't just for building whizzy stuff. Here's how to build a series of navigation buttons for a site.
9. **Hypertext links in Director:** Use hypertext to navigate through a Director movie that is on the web or CD.
10. **Sound tricks in Director:** Download the source files that show how to create 3D Sound, a volume slider, a piano and a Player Piano.
11. **Lingo Terminology:** What does "me" really mean? What's a "Function". These questions and more are answered.
12. **You Don't Know Squat!!:** The IMM version of "You Don't Know Jack".
13. **Color Correction in Fireworks MX:** Everything you ever wanted to know about colour correction in Fireworks MX. From levels to USM with a number of stops in between.
14. **Editable text and "enveloping" in FreeHand MX:** In this recent tutorial posted to the CommunityMX site , I demonstrate how text that has been "enveloped" in FreeHand MX can still be edited.
15. **Multipage Comprehensive Designs:** In this tuorial posted to the Community MX site, I explain how the multipage feature in Freehand MX and the slicing capabilities of FreeHand MX combine to cut production to almost minimal levels.
16. **Autoshapes in Fireworks MX 2004:** In this tutorial, written for the Community MX site, I explain the basics of the new Autoshapes feature of Fireworks MX 2004. Think "interactive clip art". Think of clocks filled with Blue Goo and more stars than Hollywood and you have an idea of these unique additions to the Fireworks MX tools.
17. **The Color Replacement Tool in FireworksMX 2004.** I explain , in a Community MX tutorial, how the new Color replace tool is quite an amazing addition to the line up. In this exercise you change the colour of a flower and add color to a greyscale tomato.



questions, whipped out a lined pad of paper, and said, "Let's get to work."

At this stage of the process, there is very little, if any, design work to be done. The key issue is how to incorporate the movement of dynamic data into a page composed solely of <div> tags.

If you look at the structure of the page in Dreamweaver MX 2004's design view, rolling over a <div> is indicated by a red outline, as shown in Image III,



around the perimeter of the <div> area. If you are used to creating pages with tables, a <div> replaces either the table or the cell of a table. This means, theoretically, a CSS-p page can actually be turned into a Dreamweaver template. The <div> tags can be placed in the Optional regions of the template and the content can flow into the editable regions of a Dreamweaver MX 2004 template that are commonly used in dynamic pages.

Though the theory is sound, I wanted to be absolutely sure I could do this. I went to Murray Summers, coauthor of the definitive book on the use of templates in Dreamweaver MX, for a second opinion. His response was rather interesting: "I like it when I look at the page's source and all I see is <body>. This tells me that I am either dealing with a rank amateur, or an experienced master!" As you can see, there is a razor-thin margin for error in this project.

When James pulled out his lined pad, he was assuming control of the first part of the process: planning the data.

## Planning the Data

This step involves a lot more than simply pointing to a text area on a page and saying, "Well, the text goes here." When planning the data you have to "think like a computer" and create the

database in a logical manner. This is the area where the geeks should take control. Databases are not created on the fly. They are carefully designed and planned to meet both immediate and future needs.

When planning the data consider these points:

- **Determine what is stored in the database:** To the freak it would be pictures and text. To the geek text isn't an

answer because, as James put it to me, "That's great, Tom. What kind of text are we dealing with here?" In the case of the project at hand the text could be two pieces of data – a headline and a brief description of the tutorial's contents on the entry

page. In the actual tutorial page the text is the tutorial's write-up, headline, image captions, and links to sites using the technique.

- **Determine what each piece of data looks like:** This is not a formatting question. For example, the tutorial description would be a "tinytext" field in a database. The main narrative, meanwhile, calls for the "longtext" data type.
- **Determine a model that meets your needs:** Again, this is not a design issue. The model is the construction of the database, which is a collection of the fields that hold the data and the tables that contain the fields. The tables in the database will most likely contain data that is related to other fields. James and I spent a lot of time examining the various fields and tables that we will need to create.
- **Determine the relationships between each table within the database:** Take the time to do this properly and you can actually consolidate your data and avoid repetition. For example, a tutorial's headline could be used in both the entry page and the tutorial page. This means one headline has two uses and, therefore, doesn't need to be reproduced. Freaks tend to call this process "getting organized" while geeks like

James insist on calling it "data normalization."

- **Account for the present and the future:** James intuitively understood the importance of this point. I had a vague idea and he started to focus my thinking around this one.

Situations change as data is added or removed, and failure to consider this point at the beginning of the process will hand you a maintenance nightmare. Tutorial sites have to be current. This makes them "living pages" as new tutorials are added and old ones are removed or updated to accommodate product changes or industry best practice.

For example, when Macromedia released FreeHand MX they included the use of the Live Effects found in Fireworks MX. A database planned to accommodate the future would be designed in such a way as to easily allow us to add this feature to a table that contains a list of filters and effects in FreeHand MX. If we didn't accommodate this kind of growth it would be difficult to graft FreeHand MX Live Effects onto a model that had them hardwired as fields in a table.

## Choosing the Database

Once James had an idea of the data, the discussion then swung over to the choice of database. This decision was easier to make than it first appears. Because James was involved in the process from the start, the choice of database was his to make. From the freak point of view this is a godsend. I have no problems discussing the nuances and fine details of software, but when it comes to databases, this is uncharted territory. As an experienced database developer James had a personal preference, which in this case was MySQL. The reasons for the choice, apart from the fact that it's free, were that it is fast and stable, and can handle a large amount of data.

By involving a geek like James right at the start of the process I also avoided the potential for error. The last thing you need is to plan around a database and then have to scrap the plan and any work that went into it. If the database developer is brought into the process after the fact and discovers the data plan is incorrect or you are trying to "shoehorn" one

technology into another, you'll eventually have to start all over again or resign yourself to less-than-optimal performance.

### The Data Dictionary

As James and I reviewed the plan his notepad was becoming filled with important points covered in our discussion. This wasn't because James wanted a written record, but because he was going to eventually have to construct a data dictionary. This document is the data model and is nothing more than a list of the tables and fields used to store that data.

A data dictionary can be created using a spreadsheet, an HTML table, or a vector drawing tool like FreeHand. It should contain the following headings:

- **Field:** This will be the field name in the database. (Geeks often refer to fields as columns.)
- **Type:** What type of data will be in the field? Text? Time and date? Numeric? (Geeks will sometimes use the phrase "datatype" in this case.)
- **Length:** How large will the field be? If it is a headline it doesn't need to have more than 100 characters. This area is generally flexible and can usually be changed at a later date if needed.

- **Description:** Which data is going into the field, or how the field functions in the table. This is not a technical requirement for database design as

such. But you would be foolish not to take the time to think your description through. It's helpful to think of the description in the same way you think

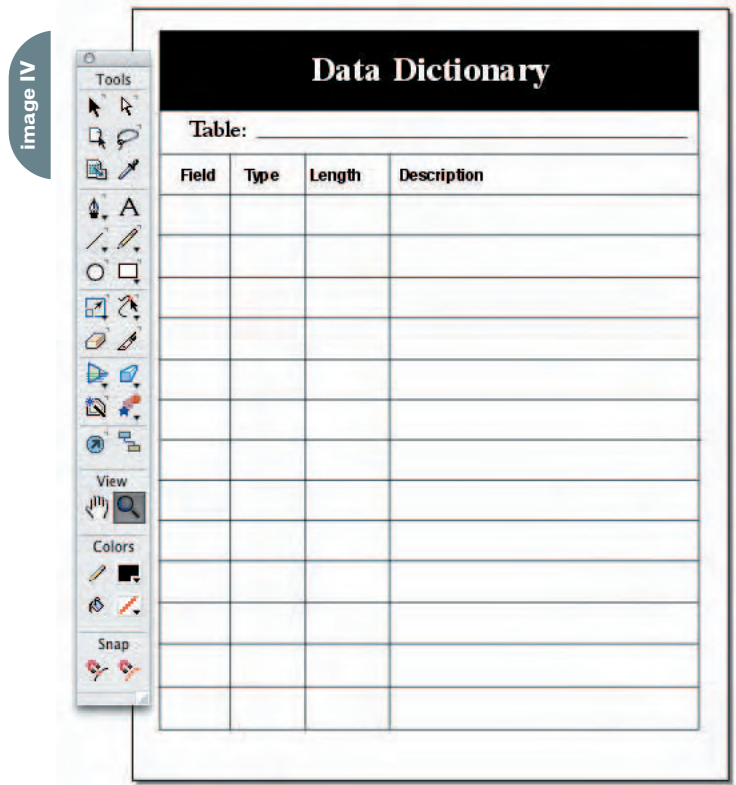
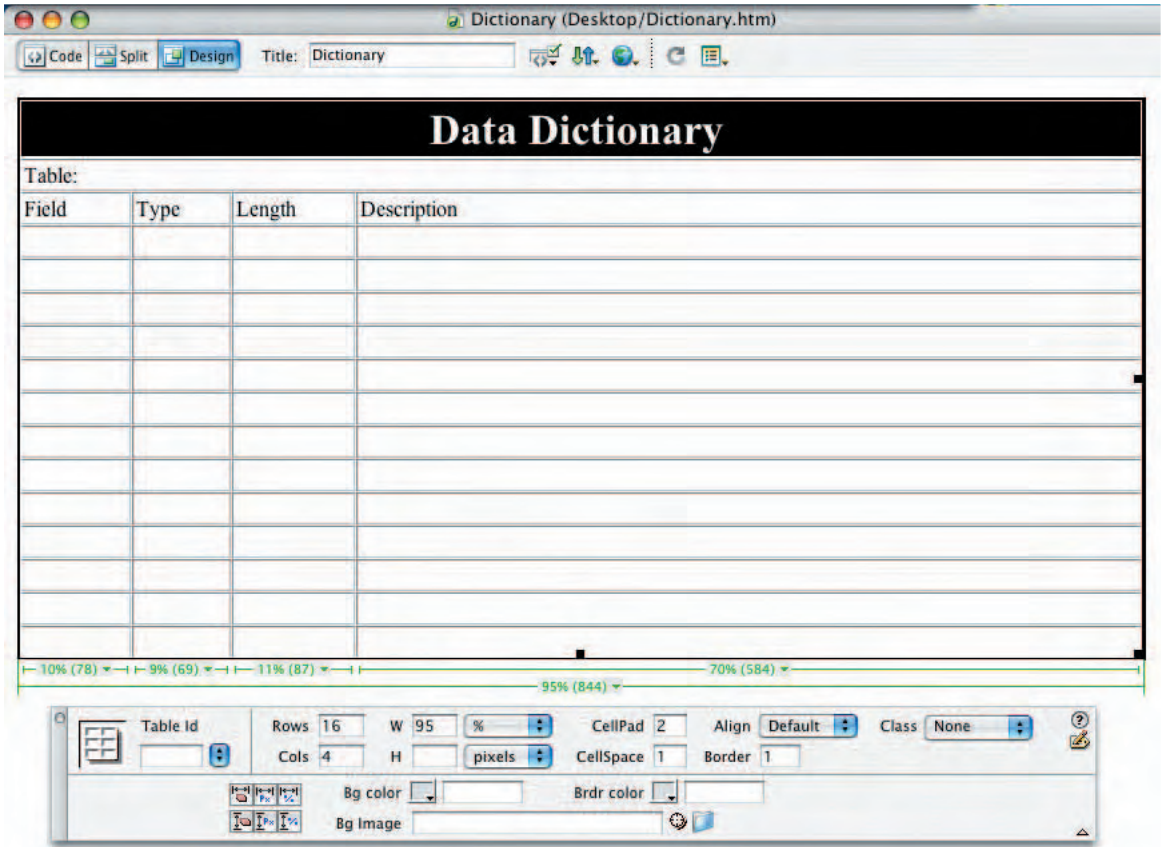


image V





about comment tags when creating a Web template. The better you document your work at the beginning, the easier it is to collaborate with others downstream.

This document can quickly be created using the text, shape, and line tools of FreeHand MX and then, using File>Export can be saved as a PDF document for use by members of the team. The other alternative, if you want to remain digital, is to use Dreamweaver MX 2004's Tables feature to create the data dictionary. This page can then be posted to the group's work site (see Images IV and V).

### The Back End

The next issue we dealt with was the "back end." This is how data gets entered in to the database using a Dreamweaver MX 2004 page. My need was succinct: "it has to be simple."

This has nothing to do with me and everything to do with growth and use. Like James, I was looking to the future. The plan is to eventually open this section to students and faculty. Thus the back end had to be intuitive and flexible. It had to be designed in a manner that allows content to be quickly added, changed, or removed without a steep learning curve. From a design point of view, this feature, because the visitor never sees it, has to stress functionality and usability over aesthetics.

### Data Types

The final piece of the data-planning puzzle was defining exactly which data types were needed. Based upon the cur-

rent design and the existing pages this was fairly easy to identify. For the entry page the data would consist of three fields. The first is a unique identifier, called a primary key, for each tutorial, which would be an integer. A text field to hold the name of the tutorial would also be required, as well as another text field for the tutorial's description. For the actual tutorial the data would consist of fields holding the names of the JPG images for the tutorial and fields for the captions and tutorial write-up.

On the freak side of the equation, it became pretty clear during James' questioning that the page listing the tutorials would have to change. It was not user friendly, and a more accessible design approach is needed.

For example, new tutorials are simply tossed in at the bottom of the listing area. This is not exactly a visitor-friendly approach.

First, I'm making the visitor's "job" terribly difficult by forcing him or her to scroll through the list looking for a particular tutorial. On top of that, the tutorials aren't identified by product. This means a student looking for my color correction tutorial would have to not only scroll through the list but also read each description. This requires an investment of time that I am sure not many visitors are prepared to make.

The solution to this usability problem involves categorizing the tutorials by MX product. The visitor would select the product – for example, Fireworks MX 2004 – from a list. That choice would then initiate a "call" to the MySQL database and build a list of all of the Fireworks MX 2004 tutorials that include this particular

categorization. In this manner, I can also look to the future. As the numbers grow they can easily be refined into subcategories (e.g., Fireworks>Color Correction>Using Levels), making the design even more accessible.

From a design point of view this decision opens up the page by reducing the amount of content presented. It also becomes more usable and functional, designed more to meet the visitor's need for information access than my need for expediency.

### Conclusion

As you have seen, converting a static page to a dynamic page involves a lot more than a designer pointing at an area of the screen and saying, "Text goes here; images go there." Involving a database and/or ColdFusion pro right at the start of the process will keep the project moving in a straight line toward success. These geeks, as I call them, are now an integral members of the Web development team and, rather than being regarded as adversaries, they are now collaborators.

Making a site dynamic is a complicated process, and where the designer (freak) sees an image, the geek sees something even more important. He or she sees data, which is the raw material for dynamic site development.

### Stay Tuned

With the planning process complete, it's time to go to work. James will handle the task of bringing order to my chaos, and I will learn that making a CSS-p page dynamic is harder than it seems. We'll document that process in Part 2. ☺

*Teacher, author, lecturer, chief cook and bottle washer, and Instructor at Humber College's School of Media Studies in Toronto, Tom Green is also the author of Building Web Sites with Macromedia Studio MX and Building Dynamic Web Sites with Macromedia Studio MX 2004. Both are published by New Riders. tgreen17@cogeco.ca*

*The course coordinator and "Lead Geek" for the Internet management and Interactive Multimedia programs through the School of Media Studies at Humber College in Toronto, James Cullin currently teaches courses in Internet technology and Web programming.*

## xile

written & illustrated by louis f. cuffari 5



**PREVIEWING MAY 11, 2004, AT NETWORLD+INTEROP, LAS VEGAS**

# **INFORMATION STORAGE+SECURITY JOURNAL!**



**FOR MORE INFORMATION VISIT  
WWW.ISSJOURNAL.COM**



*From the World's Leading i-Technology Publisher*

This is a very good time to be a Web application developer. Over the years we have moved from complex and proprietary methods of sharing data, to a more standardized and easy-to-implement method of exchanging simple or complex objects over the Web.

Though nothing is perfect, data transfer via Web services is much more standardized and has been made more simple than anything we have had in the past. Web services allow us to not only share data but to share complex data. With Web services you can send and receive objects that range from simple numbers to record sets.

# Integrating Flash MX 2004 and ColdFusion MX 6.1 with Web Services

by curtis p. hermann



# **FLASHFUSION**



This is accomplished by defining interfaces and by passing data back and forth in XML format. Both ends of the communication pipe know how to translate the XML to native objects, like a String or Array. Since the data is defined with XML, it is platform independent. As long as this data adheres to the shared standard, multiple applications can communicate and know nothing about the other's technology. This is a very powerful distributed computing world that has existed since the inception of the World Wide Web. You cannot only share data between different components of one application, but can make it available for others to grab, like the latest-breaking news from your Web site or stock quotes.

The standard used in the technologies discussed within this article is a protocol called Simple Object Access Protocol (SOAP). SOAP has taken the lead in messaging standards. Furthermore, Macromedia, along with other technology giants like IBM, Microsoft .NET, and others, has decided to adopt SOAP with the new set of data components and Web services classes.

### What About Flash Remoting?

What we will be looking at in this article are Data Components and WebServices classes, but there is another technology that Macromedia produces called Flash Remoting. The WebServices classes are much like Flash Remoting in that they act as the mechanism to transport data between applications without requiring that the developer knows how it is done. WebServices classes are based around SOAP whereas Flash Remoting is a proprietary technology. Those who already know about Flash Remoting

might be asking the question, "Are the ActionScript Web Services classes replacing Flash Remoting?" The answer is no. Macromedia has just been focusing on SOAP-based Web services to meet the needs of the community. Since they already had Flash Remoting up and running, internal development efforts have been concentrated on SOAP-based initiatives.

For those of you who don't know much about Flash Remoting, it is a technology that is used in the same way as SOAP Web services, but is proprietary to Flash MX+ applications. Flash Remoting is a Macromedia technology, whereas SOAP is an open standard. One advantage that Flash Remoting has over SOAP, is that it is sent in binary format so that it is much more efficient and "lighter weight." SOAP, on the other hand, is very verbose in its object descriptions and can be much slower than Flash Remoting when returning large results; this is because SOAP is based on passing XML strings between machines. A disadvantage of Flash Remoting is that it is not free – kind of. You either need to purchase it for your server (for "pure" Java or .NET), or be running on a ColdFusion MX+ or JRun 4+ platform. Also, at the time of this writing, Flash Remoting is still written in ActionScript 1.0 while the Web service classes and data components in Flash MX 2004 are written in ActionScript 2.0. So, if you are into developing strictly ActionScript 2.0 applications, and you use Flash Remoting, you will need to still use ActionScript 1.0 syntax to include the libraries into your application. However, Macromedia promises that a new version written in ActionScript 2.0 is coming soon. I look forward to this release. I myself have written many applications

using Flash Remoting and enjoy the technology very much and encourage others to explore and work with it as well.

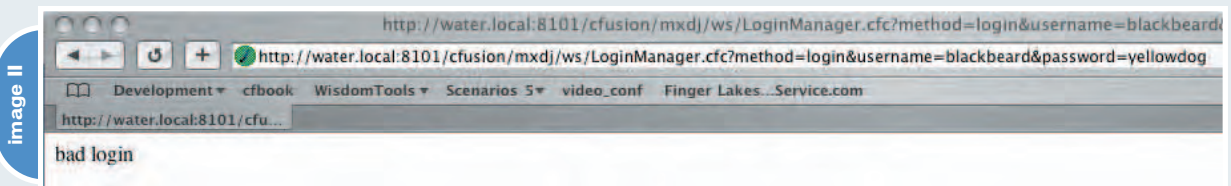
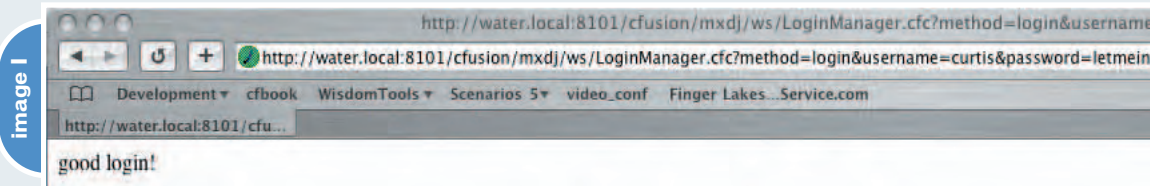
### Web Services, Flash MX 2004 and ColdFusion MX 6.1

The problem we are trying to solve with these technologies is how to push data between applications that don't speak the same language and use the Web as their vehicle. In versions of Flash prior to Flash MX, developers would use the LoadVars method, URL rewriting, and manually parsing XML files. These methods work, but are either too limiting in the data they pass or require too much knowledge about how the data is structured (they are not suited to the transfer of complex data). In the case of using Web services, Flash can send ColdFusion an object that was constructed in ActionScript (represented as XML), and when ColdFusion receives it, the ColdFusion Application Server translates the XML into a native variable and uses that object as a ColdFusion Struct.

The same happens in reverse. Flash knows nothing about ColdFusion CFML and ColdFusion knows nothing about Flash ActionScript, but they do share a common language – SOAP – and it's a beautiful thing. The same holds true for Java, PHP, Visual Basic, and other programming languages.

### What We Are Going to Do

To illustrate how this communication works, I have chosen a user login screen to be the example. This is something we all deal with, so I figured I would not have to explain the application purpose and we could just focus on the Web service communication. This is not how you





would do user authentication in the real world. For instance, the example does not do any user input validation and obviously you would want to build the application to authenticate from a dynamic data source rather than just a hard-coded example as you will see.

In our sample application ColdFusion is going to be the technology used to publish the Web service that supplies Flash with data from the server. Flash is going to make calls to the ColdFusion server to get that data, and will pass any data that the server requires for its methods. SOAP is going to be the vehicle for the data to ride on, between the two.

In a nutshell, the user will type in a username and password, then click the login button. At this point Flash will pass the two arguments to ColdFusion in the form of a SOAP request. ColdFusion will take that SOAP XML and translate the two arguments to a string type that ColdFusion can understand. ColdFusion will then evaluate the username and password and return a valid or invalid string value back to Flash in a SOAP Response. Flash then receives that XML data structure, converts it to a String object it can understand, and displays the result to the user. That's it!

The last thing to discuss before we go into the project, is that Flash MX 2004 interacts with Web services in two ways. First, it uses design time graphical components to set the properties in the IDE rather than in code. You can actually develop applications without typing a line of code. Second, it uses the Web service classes and does not rely on the graphical components. We will examine both in this article.

## The Web Service

By far, ColdFusion MX is the simplest way to create a Web service. ColdFusion MX does this through CFCs (ColdFusion Components). You can think of a CFC as a Web accessible class. Like classes in other languages such as ActionScript and Java, a CFC has methods and properties. CFCs are saved in a file with a .cfc extension and the file location and name determines the class and package names. Let's take a look at the basics of a CFC:

<CFCOMPONENT>

This is the root tag that will encapsu-

image III

```
<?xml version="1.0" encoding="UTF-8" ?>
<wsdl:definitions targetNamespace="http://ws.mxdj" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns1="http://rpc.xml.coldfusion" xmlns:impl="http://ws.mxdj" xmlns:intf="http://ws.mxdj" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/">
  <wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://rpc.xml.coldfusion">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <complexType base="soapenc:Base" name="CFCInvocationException" />
      <sequence />
      </complexType>
    </schema>
  </wsdl:types>
  <wsdl:message name="CFCInvocationException">
    <wsdl:part name="fault" type="tns1:CFCInvocationException" />
  </wsdl:message>
  <wsdl:message name="loginRequest">
    <wsdl:part name="username" type="xsd:string" />
    <wsdl:part name="password" type="xsd:string" />
  </wsdl:message>
  <wsdl:message name="loginResponse">
    <wsdl:part name="loginReturn" type="xsd:string" />
  </wsdl:message>
  <wsdl:portType name="LoginManager">
    <wsdl:operation name="login" parameterOrder="username password">
      <wsdl:input name="loginRequest" message="impl:loginRequest" />
      <wsdl:output name="loginResponse" message="impl:loginResponse" />
      <wsdl:fault name="CFCInvocationException" message="impl:CFCInvocationException" />
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="LoginManager.cfcSoapBinding" type="impl:LoginManager">
    <wsdl:soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  </wsdl:binding>
  <wsdl:operation name="login">
```

image IV

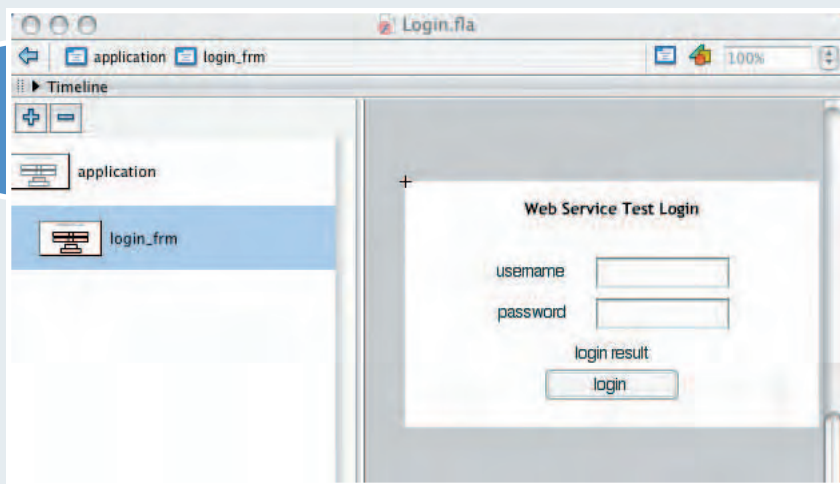


image V

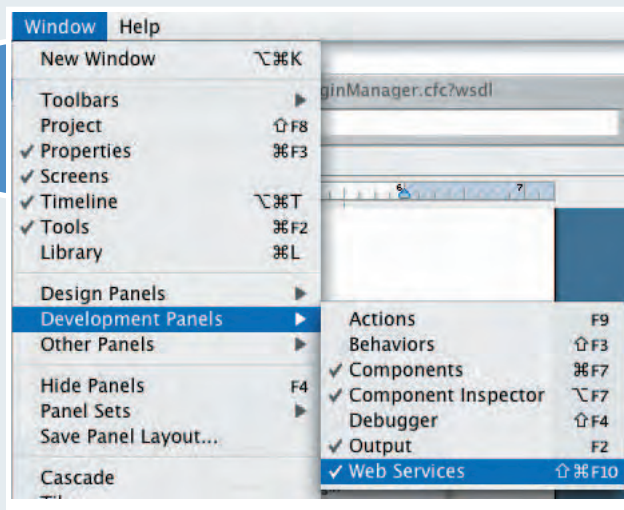
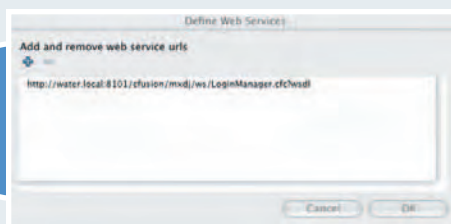


image VI





late all of the supporting code. There are no required attributes, but it can take two optional attributes HINT and DISPLAYNAME. These attributes are used to document the component, which tools like Dreamweaver MX 2004 take advantage of.

image VII

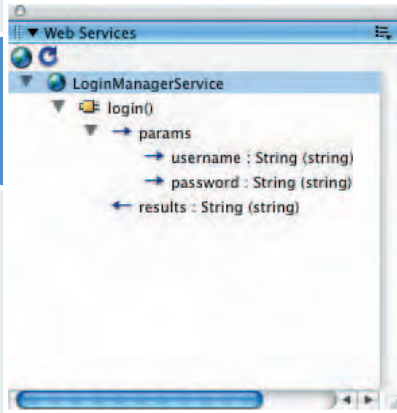


image VIII

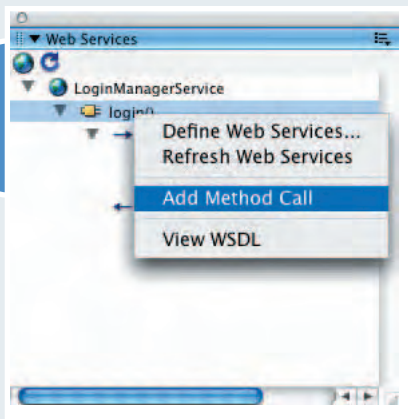


image IX

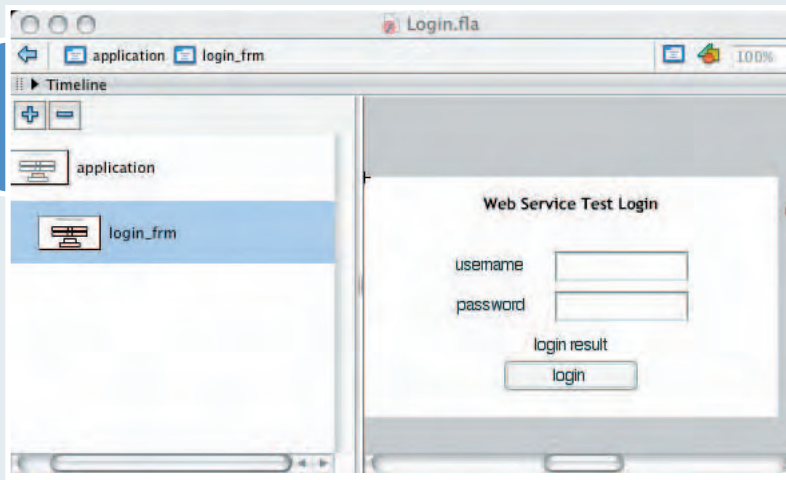
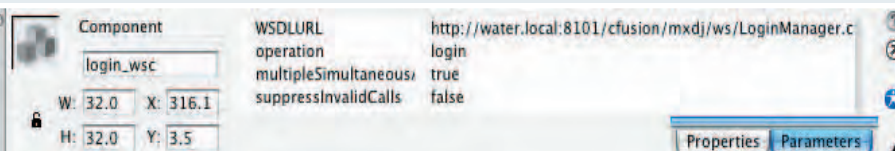


image X



#### <CFFUNCTION>

All of your methods will be written within the <CFFUNCTION> tag. The attribute NAME is the only required attribute, but when developing Web services you will always need to set the ACCESS attribute to "Remote"; otherwise ColdFusion MX server will not make it accessible to Flash as well as specifying the RETURNTYPE of the method. The RETURNTYPE attribute is an attribute that you will get accustomed to very quickly. The RETURNTYPE must be set when treating the CFC as a Webservice, even if the method does not return anything; then set the returnType attribute to void. This attribute sets what type of object is allowed to be returned, such as a struct, date, string, etc. The ROLES attribute is used to set a security level on who is allowed to access this method. The OUTPUT attribute determines whether or not the method can produce output, like the <CFOUTPUT> tag, but when developing for WebServices this should be set to "no". Once again, HINT and DISPLAYNAME are optional and should be used to help document your component at the method level.

#### <CFARGUMENT>

The <CFARGUMENT> tag is used within the <CFFUNCTION> tag to define the arguments taken by the method. <CFARGUMENT> requires that the NAME attribute be defined; this gives a readable identifier to the argument like "username" or

"id". The TYPE attribute is required for WebServices. The REQUIRED attribute takes a Boolean value to set whether or not this argument must be passed. But when developing CFCs for Web services, the attribute required="false" is ignored, all arguments are required, and the DEFAULT attribute is not used. As always, HINT and DISPLAYNAME are optional and should be used to help document your component at the argument level.

For more information on publishing CFCs as Web services check out: [http://livedocs.macromedia.com/coldfusion/6/Developing\\_ColdFusion\\_MX\\_Applications\\_with\\_CFML/webservices5.htm](http://livedocs.macromedia.com/coldfusion/6/Developing_ColdFusion_MX_Applications_with_CFML/webservices5.htm)

### LoginManager.cfc

In this article I am using a login page as an example of integrating ColdFusion MX and Flash MX 2004 through Web services. To make the login happen I need a component that will accept a username and password, then return a success if they are good, or a failure if they're not.

### Setting Up the Application Under ColdFusion MX

For this article I set up a directory structure under my ColdFusion MX web-root as follows: [coldfusion web root]/mxdj/ws/

Then I saved a file named LoginManager.cfc in the ws directory: [coldfusion web root]/mxdj/ws/LoginManager.cfc

Open this file in your favorite editor (I use Dreamweaver MX 2004) and place the following code.

First start with the component definition:

```
<cfcomponent>
</cfcomponent>
```

Second, define the method to login:

```
<cfcomponent>
<cffunction name="login"
access="remote" returnType="string">

</cffunction>
</cfcomponent>
```

*Note:* The access attribute is set to remote; if you do not set this attribute correctly the component will not be accessible to Flash.

# LOOK FOR YOUR **FREE...**

**IT solutions >GUIDE**

»Linux »Java »Web Services ».NET »XML »Wireless »Storage »Security

The Premier Resource for Today's Corporate & IT Decision Makers

VOL 1 ISSUE 1 SPRING 2004

WWW.SYS-CON.COM/IT

**TECHNOLOGIES YOU NEED NOW!**

How to Manage Your Ideas Using Today's i-Technologies

- » Delivering Software as Service
- » Leveraging Linux/Open Source
- » Moving to a Service-Oriented Architecture
- » Desktop Software: Migrating from Server to Client
- » Using Developer Tools to Drive Cost Out of Software
- » Application Integration
- » Storage & Security

**Reaching 135,000**  
Corporate and IT Decision Makers

\$5.99US \$7.99CAN 12>

0 09281 01121 7

Coming this **SPRING!**

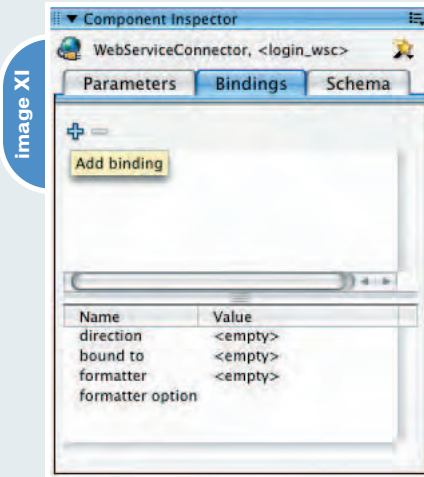


Image XI

Third, define the arguments needed to log in:

```
<cfcomponent>
<cffunction name="login"
access="remote" returnType="string">
  <cfargument name="username"
type="string" required="true">
  <cfargument name="password"
type="string" required="true">
</cffunction>
</cfcomponent>
```

Finally, write the code to verify the username and password, as well as return the result of that evaluation.

```
<cfcomponent>
<cffunction name="login"
access="remote" returnType="string">
  <cfargument name="username"
type="string" required="true">
  <cfargument name="password"
type="string" required="true">
  <cfif username eq "curtis" and pass-
word eq "letmein">
    <cfreturn "good login!">
  <cfelse>
    <cfreturn "bad login">
  </cfif>
</cffunction>
</cfcomponent>
```

Obviously, this is just a quick example of a login method that is not very realistic. A real login method would be connecting to a database and not using hard-coded values as this one does.

That's about all it takes to create a component. Go ahead and test it out by accessing the component in your Web browser with the following URL (replace the localhost:8500 with the URL and port to your ColdFusion MX server):

```
http://localhost:8500/mxdj/ws/LoginManager.cfc?method=login&username=curtis&password=letmein
```

You should see a result like the one in Image 1.

Now, produce a bad login by changing the arguments to an invalid username and password:

```
http://localhost:8500/mxdj/ws/LoginManager.cfc?method=login&username=blackbeard&password=yellowdog
```

You should see a result like the one in Image 2.

Now that we have our component working, let's turn it into a SOAP Web service that can be used by the new data components in Flash MX 2004. Well, there is not really anything to do, except change the way we access it. By adding ?wsdl to the end of the URL to access a component tells ColdFusion MX that we want our communication with that component to be SOAP based. To look at what our CFC looks like using SOAP type, see the following URL: <http://localhost:8500/mxdj/ws/LoginManager.cfc?wsdl>.

WSDL (Web Services Description Language) is an XML format for describing Web services. It defines what methods (or operations) are available, what parameters the method will take, and what the method returns.

The result should look like Image 3. Note: If your browser appears to be blank, view the document source. Some browsers do not display WSDL.

This is how WSDL describes the LoginManager CFC, so that when other SOAP-aware applications attempt to use this service, they know how the objects are defined.

## The Flash Front End

When using Flash MX 2004 data components, you can either set them up through the IDE by dragging and dropping data components onto the stage and binding them to the data graphically using the component inspector or by hand coding the connection. I will start with the first option, then show an example of coding everything by hand. Some familiarity with Flash is assumed.

## The Interface

1. Create a new Flash Form Application and save it as Login fla
2. Rename form1 as login\_frm
3. From the components panel under the UI Components node, use TextInput components to create two input fields named "username\_inpt" and "password\_inpt". For the second TextInput component, password\_inpt, set the password value to true in the Properties panel. Add a Button, call it "login\_btn" and change the label to "login". Then add a Label component,

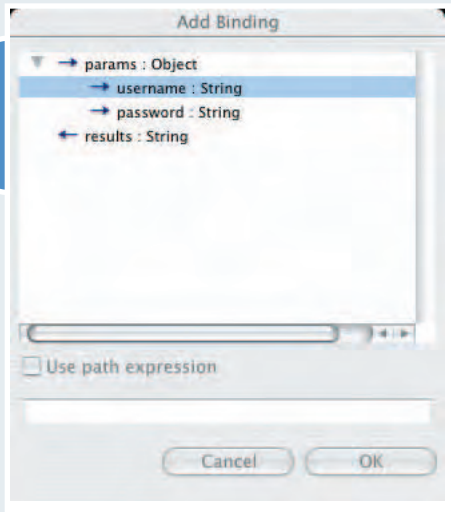


Image XII

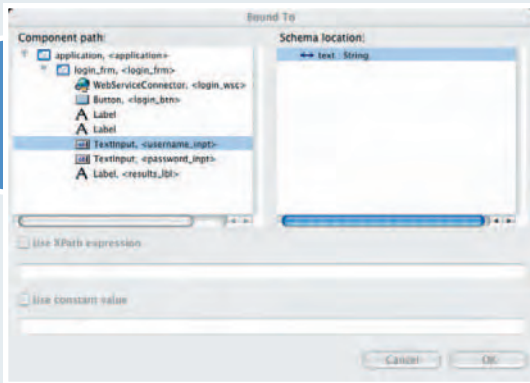


Image XIII

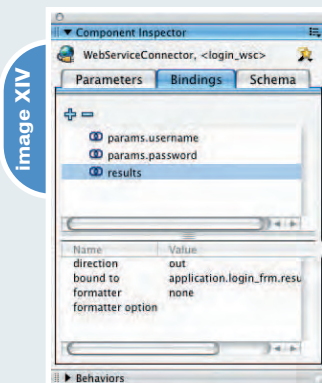


Image XIV

name it "results\_lbl", and change the label property to "login results". Finally, finish the interface by adding the appropriate labels and text fields to make login\_frm look like Image 4.

Now that the interface is in place we can focus on getting data into it.

### Web Services Panel

A great way to inspect available components is to view them through the Web Services Panel, which can be accessed via the menu in Flash MX 2004 under Window-Development Panels-Web Services (see Image 5).

To add a Web service, click the globe icon in the upper left hand corner of the panel. A Define Web Services dialog will open. We want to view the LoginManager service so click the plus icon to add a service, then enter the URL to the service and tack ?wsdl to the end (<http://localhost:8500/mxdj/ws/LoginManager.cfc?wsdl>; see Image 6).

Press the "OK" button. Now your Web services panel will display the newly added service. In this panel, services are displayed in a tree so that you can drill down to its method and that method's arguments and results. Expand each level of the LoginManager service so that you can see the Login method, its arguments, and results (see Image 7).

### The WebServiceComponent

To make calls to Web services we need to add a WebServiceConnector component to our application. This component can be found in the components panel under the Data Components node. There are two properties that need to be assigned to use this component: WSDLURL would be the URL to LoginManager.cfc and operation, which would be the method (login) you want to call. This component can be used by dragging the object from the Components panel to the stage and assigning those values, or you can right-click the method login in the Web Services panel and select Add Method Call from the popup menu (see Image 8). This will add the component to the stage and fill out the appropriate values needed.

Once the component is on the stage go ahead and move it off the viewable

area. It is not seen at runtime, so wherever you place it on the screen is just a matter of organizational preference (see Image 9).

Select the WebServiceConnector component and name it to login\_wsc. Also, verify that the WSDLURL and operation property is correct (see Image 10).

### Binding

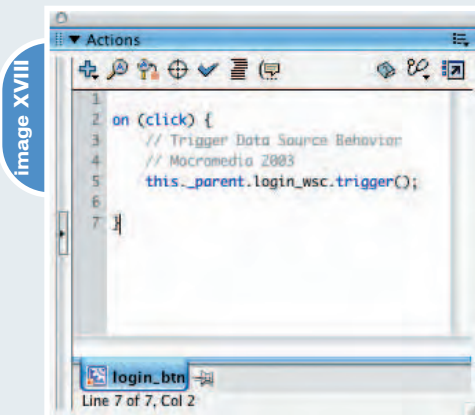
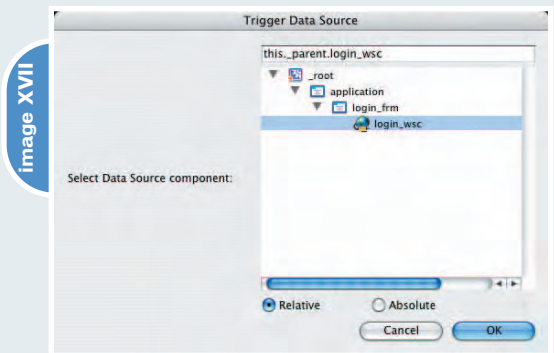
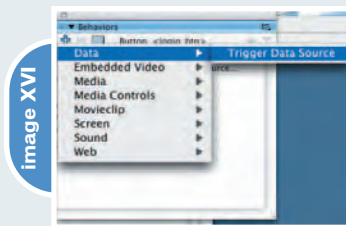
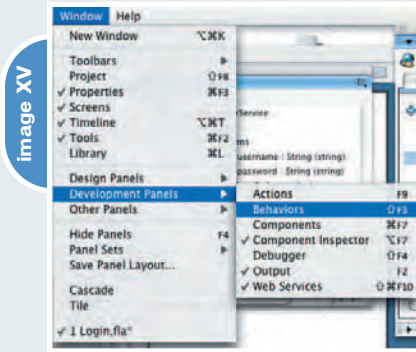
As you know already by writing the LoginManager CFC, the login method requires two arguments: username and password. It also returns a result string. How these values get set is through the username\_input and password\_input TextInput components. The result is displayed by the results\_lbl Label component. The quickest and easiest way to do this is to set the connection between these components and the login\_wsc WebServiceConnector through the bindings tab in the Component Inspector panel.

To bind the arguments to the TextInput components, select login\_wsc then click on the bindings tab in the Component Inspector tab. Click the plus icon to add a parameter binding (see Image 11).

An Add Bindings dialog will open displaying the possible arguments and results that can be bound (see Image 12).

Select the username argument and press the OK button. Back in the Component Inspector panel, under the Bindings tab, you will now see the username argument in the Bindings list. With the username binding selected, click in the bound to property field to bind the username argument to the username\_inpt TextInput component. Within the Component Path tree lays the login\_frm and all of its components. Select the <TextInput> username\_inpt node then click OK (see Image 13).

The component is now bound to the username argument. Notice that the direction property is equal to "in". When a user types in a username, that value will be bound to the username argument and passed to the login method. Repeat the same process to bind the password argument to the password\_inpt TextInput component. When passing values to the WebServiceConnector, the direction value will be "in", but when receiving values from a





WebServiceConnector the will be "out". You will now see how the "out" direction works with the next step.

At this point you are able to pass arguments to the WebServiceConnector, but are not able to receive the result. To do this, click the plus icon on the Bindings tab the same way you added the arguments. This time select the results node and click OK. In the Component Inspector, under the Bindings tab, you should now have three items that are bound: username, password, and results. To complete the binding of the results, click in the bound to property field, then drill down to the Label <results\_lbl> node in the Bound To dialog, finally clicking OK. Your Component Inspector panel should now look like Image 14.

Okay, now our interface is bound to data! All we have to do now is trigger the data source. To continue down the path of NO CODING, we will trigger the data source using the new Flash MX 2004 behaviors. To do this, select the login\_btn on the stage, then open the Behaviors panel, accessed via the main menu-Window-Development Panels-Behaviors (see Image 15).

To add a behavior, click the plus icon, then navigate the menu Data-Trigger Data Source (see Image 16).

The Trigger Data Source dialog will open. Drill down the tree and select login\_wsc, then click OK (see Image 17).

To see what has happened, select the login\_btn Button component and open the actions panel. As you can see in Image 18, code has been generated for you to trigger the login\_wsc WebServiceComponent.

So, now we are bound and triggered and ready to log in. Test your application and enter in a username and password. If you want to have a successful login enter "curtis" for the username and "letmein" for the password. If the username and password are correct, then the results\_lbl will display "good login!" (see Image 19). Otherwise, a bad login will result in "bad login".

This has been a simple example of how the WebServiceConnector component and a ColdFusion CFC can work together. Through the use of the Component Inspector, you can bind data to other components without one line of code. It's fun to throw together a quick application or prototype using these tools. However, you can soon find yourself limited by this way of application development and find yourself deep in code to solve the problem.

## Coding the Web Services Classes

Rather than using the Component Inspector or Properties panels, I am coding 99.9% of the time. Mostly, I use the Flash interface to lay out and organize my forms and components, then I pour ActionScript into the application in external class files. One of the problems of developing applications with design time tools is that when it comes to runtime,

they cannot be dynamically changed without the use of code. Often, developers need this flexibility. Also, while storing properties in design time components, you can have a real hard time tracking down problems as the application becomes very large. It is much easier to dig through code than through multiple levels of movie clips, selecting them to find their properties. With coding in external files, you have the power of search

and replace and diff tools, that compare the differences between versions of source code, not to mention versioning. As much as these design time tools open the world of application development to many people who do not have developer experience, they do have boundaries as to what they allow the developer to do. Ultimately, you need to perform logic and code reuse. To date I have not seen a complete solution that is entirely visual in nature. They serve a good purpose by allowing beginning developers to become wrapped around advanced technologies fast, so that they can understand what the code is doing when they get there.

So let's get started on creating a coded version of the Login application. The easiest and quickest way to get things going is to save a copy of the Login.fla as Login2.fla. Next, delete the login\_wsc WebServiceConnector component from the stage and delete the WebServiceConnector component from your library. Finally, select the login\_btn Button component on the stage and delete the click behavior from the Behaviors panel. Now we are ready to set up our application with code.

We are going to need a class for the login\_frm Form. Classes in ActionScript 2.0 are defined in their own file with the .as extension. Let's begin by creating a new class from within the Flash MX 2004 development environment. From the main menu select File-New. A New Document dialog will open. Make sure the General tab is selected, then choose the ActionScript File option from the list and click OK (see Image 20).

Save the new file as LoginForm.as in the same directory as the Login2.fla. Make sure to pay attention to how you name and reference files, ActionScript 2.0 is case sensitive no matter what platform you are developing on. Many articles and online tutorials are written on ActionScript 2.0, so I will not go into it here. But, if you want to read a good one, check out the Ramping up on ActionScript 2.0 and Flash MX 2004 in the January 2004 edition of *ColdFusion Developer's Journal* ([www.sys-con.com/coldfusion/](http://www.sys-con.com/coldfusion/)). Let's write our class.

First the class definition. Since we will be associating this class with login\_frm

image XX

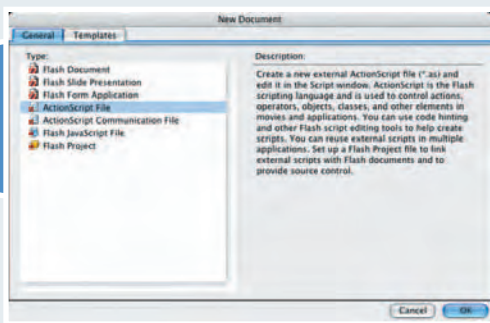
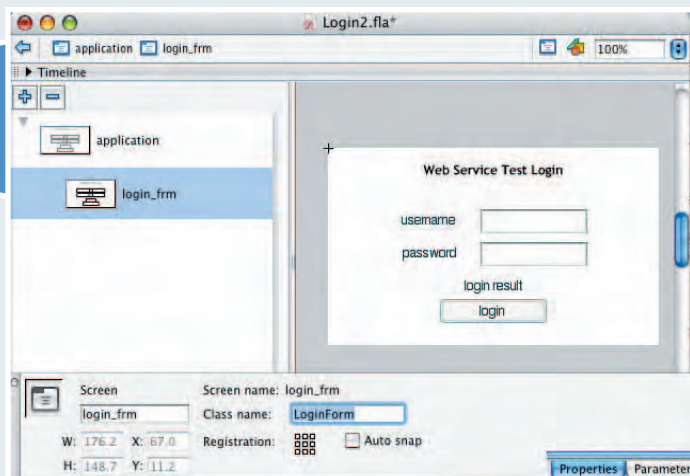


image XXI



Form, we will need to extend the Form class:

```
class LoginForm extends
mx.screens.Form {
    function LoginForm() {

    }
}
```

Save the LogForm.as file then go back to the design environment to associate the LoginForm class with login\_frm. To do this, select login\_frm in the form hierarchy pane and select the properties tab in the Properties panel. In the class name field, replace mx.screens.Form with LoginForm (see Image 21).

Now when an instance of login\_frm is created, so will an instance of LoginForm class. Before we go back to coding we need to get the Web services classes into our application. You can access these classes from the main menu by navigation Window-Other Panels-Common Libraries-Classes (see Image 22).

In the libraries panel you will see a library with the heading Classes.fla. Within Classes.fla there are DataBindingClasses, UtilsClasses, and WebServicesClasses. Now, we will deal with WebServicesClasses. Drag the WebServicesClasses icon to the stage so that it is in the Login2 library. Then delete the object from the stage.

Back in our LoginForm.as file, we need to import the services that we just added to our file.

```
import mx.services.*;

class LoginForm extends
mx.screens.Form {
    function LoginForm() {

    }
}
```

Then we need to define a variable to hold a reference to our login\_btn Button component and add a listener for the click event. Also, define references to the TextInput and Label components:

```
import mx.services.*;
import mx.controls.Button;
import mx.controls.Label;
import mx.controls.TextInput;
```

```
class LoginForm extends
mx.screens.Form {

    private var login_btn:Button;
    private var results_lbl:Label;
    private var username_inpt:TextInput;
    private var password_inpt:TextInput;

    function LoginForm() {

    }

    function onLoad() {
        var obj = this;
        var loginBtnListener = new Object();
        loginBtnListener.click =
function(evt) {

        }
        login_btn.addEventListener("click",
loginBtnListener);
    }
}
```

```
function onLoad() {
    var obj = this;
    var loginBtnListener = new Object();
    loginBtnListener.click =
function(evt) {

    }
    login_btn.addEventListener("click",
loginBtnListener);
}
}
```

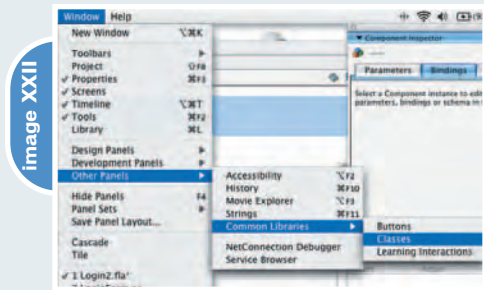
When the login\_btn Button is clicked then we should make a call to our Web service. So, within the click event for login\_btn we will set up a service and make a call.

```
loginBtnListener.click =
function(evt) {
    var loginService:WebService = new
WebService("http://water.local:8101/cf
usion/mxdj/ws/LoginManager.cfc?wsdl");
}
}
```

Next, make the call. Also, assign an object to that call to listen for the result and fault events.

```
var loginCall:PendingCall =
loginService.login(obj.username_inpt.t
ext, obj.password_inpt.text);

loginCall.onResult =
function(result) {
    obj.results_lbl.text = result;
}
loginCall.onFault = function(fault)
{
    obj.results_lbl.text = "An Error
Occurred"
    trace(fault.faultCode + " : " +
fault.faultString);
}
}
```



Finally, your completed LoginForm class should look like this:

```
import mx.services.*;
import mx.controls.Button;
import mx.controls.Label;
import mx.controls.TextInput;

class LoginForm extends
mx.screens.Form {

    private var login_btn:Button;
    private var results_lbl:Label;
    private var username_inpt:TextInput;
    private var password_inpt:TextInput;

    function LoginForm() {

    }

    function onLoad() {
        var obj = this;
        var loginBtnListener = new Object();
        loginBtnListener.click =
function(evt) {
            var loginService:WebService = new
WebService("http://water.local:8101/cf
usion/mxdj/ws/LoginManager.cfc?wsdl");
            var loginCall:PendingCall =
loginService.login(obj.username_inpt.t
ext, obj.password_inpt.text);
            // onResult is called when all goes
well
            loginCall.onResult =
function(result) {
                // display whether the user is
valid or not
                obj.results_lbl.text = result;
            }
            // an error occurred while trying
to make the Web Service call
            loginCall.onFault = function(fault)
            {
                trace(fault.faultCode + " : " +
fault.faultString);
            }
        }
    }
}
```



```
login_btn.addEventListener("click",
loginBtnListener);
}
}
```

## Debugging

No matter how good of a developer you are, bugs occur. When they do, you will need to find out what is going on. Flash MX 2004, has a built-in graphical debugger you can use with your application development. With Web services, there are a couple of objects that you can take advantage of to help you track down errors.

## PendingCall Class

If you want to see what is actually being sent and received you can use the PendingCall class to view the actual SOAP that is being produced and parsed. The PendingCall object we would use in this class would be the variable loginCall. To view the SOAP being sent, trace the loginCall.request property and to see the SOAP being returned, trace the loginCall.response property.

For example, in the loginCall.onResult event I trace the loginCall.response property. The following is what is in the output:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xml-
soap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body><ns1:loginResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="http://ws.mxdj">
```

```
<ns1:loginReturn
xsi:type="xsd:string">good
login!</ns1:loginReturn>
</ns1:loginResponse>
</soapenv:Body>
</soapenv:Envelope>
```

## SoapCall Class

The SoapCall class is created for each operation in the Web service. This object acts as a description of the call and allows you to customize that particular call delaying the conversion of SOAP objects to ActionScript. As far as debugging goes, you can loop through the SoapCall class and view the properties of the SoapCall and their values. Access the SoapCall by getting a reference to the PendingCall.myCall property.

Change the loginCall.onResult event handler to this:

```
loginCall.onResult = function(result)
{
var sCall:SOAPCall =
loginCall.myCall;
for(var i in sCall) {
trace("loginCall.myCall." + i + "
= " + sCall[i]);
}
obj.results_lbl.text = result;
}
```

## Log Class

The Log Class tracks each step that has taken place during the communication process. You might expect a file to be produced, like most logging features in software development. In fact, the Log class will send text just to the onLog event. The Log class has three logLevels:

- Log.BRIEF: The log records primary life-cycle event and error notifications.
- Log.VERBOSE: The log records all life-cycle event and error notifications.
- Log.DEBUG: The log records metrics and fine-grained events and errors.

In this example we will use the VERBOSE level.

By adding the following code we can track what is going on during our Web service method invocation:

Insert this code right above the loginService declaration:

```
var loginLog:Log = new Log(Log.VERBOSE);
loginLog.onLog = function(log_str) {
trace(log_str);
}
```

Change the loginService declaration by adding an additional parameter containing the loginLog:

```
var loginService:WebService = new
WebService("http://water.local:8101/cfusion/mxdj/ws/LoginManager.cfc?wsdl",
loginLog);
```

Now run the application and log in again. You will see output from the log object in the Output panel (see Image 23).

As you can see there are many ways to debug your application, so you won't be left out in the cold.

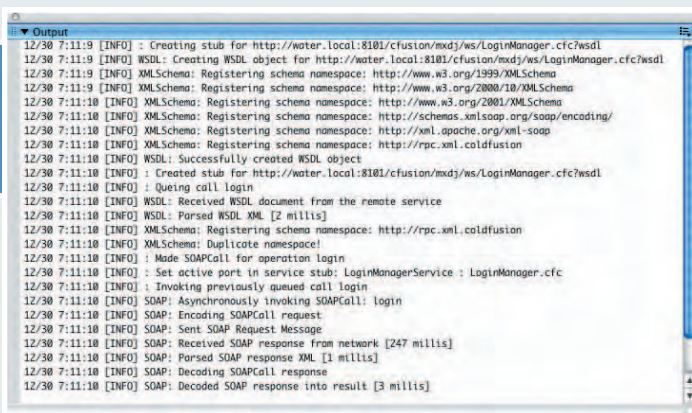
## Wrapping Up

We have covered a lot of information here, but this is only the very beginning. There are many features with the WebServices classes and Data Components that I have not come even close to touching on, but this will get developers started and exploring on their own. Web services are definitely here to stay, so as a professional in this field I advise you to study them and master their use.

Would you prefer to use the LoadVars technology of yesteryear? I think not. Coming from years of transferring data through URL rewriting, HTML form elements, and manually parsing XML documents, I am pretty set on not going back to that – and recent trends show that the industry agrees. Having the ability to work with objects, and not caring how they got to the server and came back to me, is a good thing. With integrating technologies such as ColdFusion MX and Flash MX 2004, data should be smooth and trouble free. ☺

Curtis P. Hermann is a Macromedia Certified Flash MX Developer. He owns and operates a small consulting firm, iindwell, inc. (www.iindwell.com). He also heads the Flash MX development and quality-assurance department for WisdomTools.com (www.wisdomtools.com). curtis@iindwell.com

Image XXIII





A new tool for MX professional developers and designers..



**ADVERTISE**

Contact: Robyn Formis  
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





# Integration New Media's Impressario Xtra for Director

*New use for an old friend*  
reviewed by alec east

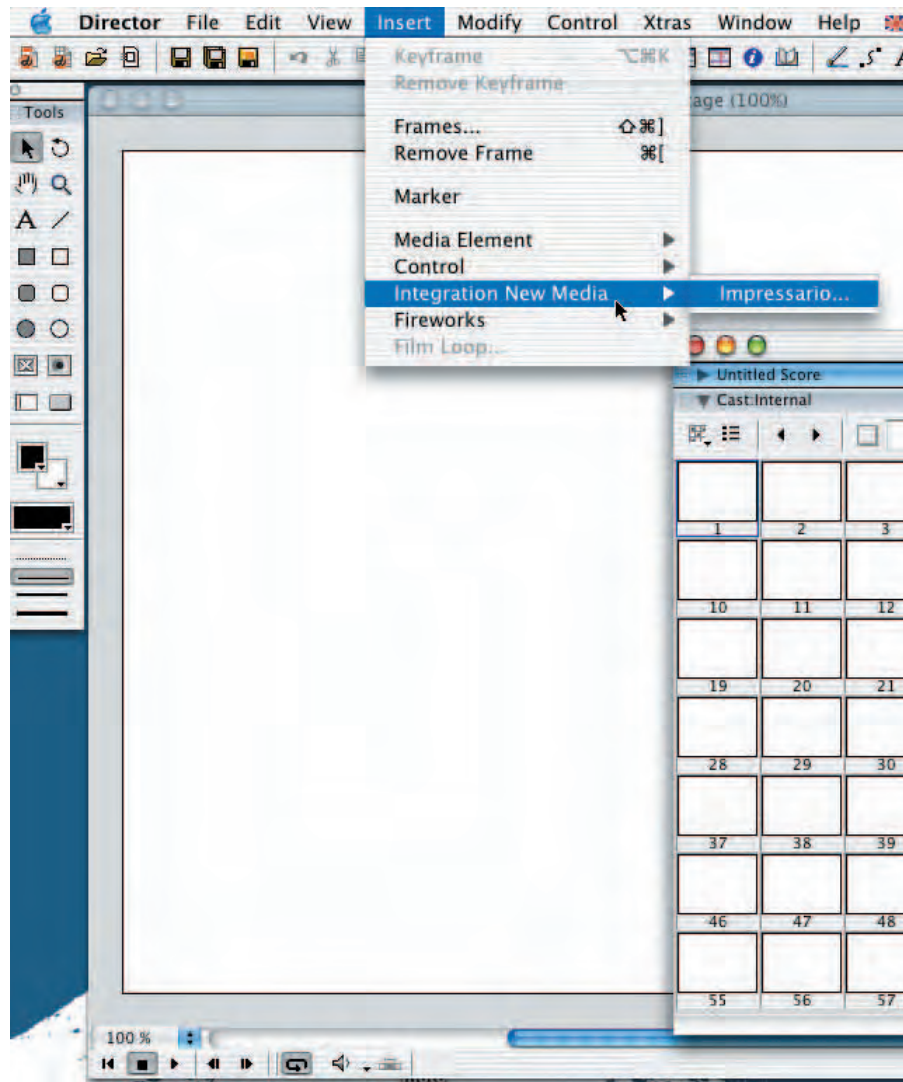
From the people that brought you the V12 Database Engine and PDF Xtra comes Impressario: an Xtra for the manipulation of PDF documents. OK, so it's true that PDF Xtra already provides navigation and zooming features to Director, but Impressario is PDF Xtra Pro. It enables you to use PDF documents in ways that, quite frankly, are really impressive.

For the purposes of this review I downloaded the trial version that requires you to have a serial number e-mailed to you. It comes as an installer that, puzzled me at first, as I know where my Xtras folder lives. But I soon discovered that the installer also adds a library of button assets and another of behaviors to assist your journey and, having had a quick peek at the code in the behaviors, I'm very glad they did.

The behaviors aren't the easiest for those unfamiliar with PDFs, but they don't need to be. Not only do you have most of the essential behaviors in a

library anyway, there are also a few examples on the site to help you get your head around it. Impressario Xtra is rich in accessibility features aimed at creating projects for the motor and visually impaired. It has the ability to extract text from a PDF document and throw it at the

Speech to Text Xtra, as well as the kind of functionality you would expect, such as zoom and page navigation. It offers a lot of other helpful features too, including support for PDF forms, embedded hyperlinks (but not many other embedded actions), the ability to open password-



## Info

### Company:

Integration New Media, Inc.  
1425 West René-Levesque Blvd.,  
Suite 906  
Montreal, Quebec  
H3G 1T7  
Canada  
1(800) 400-1772  
<http://www.IntegrationNewMedia.com/>

### Test Environment:

**Software:** Director MX 9.0

**Processor:** G5 1.8Ghz,

**Memory:** 1.5 Gig Ram,

**OS:** Mac OSX 10.2.8

### Price:

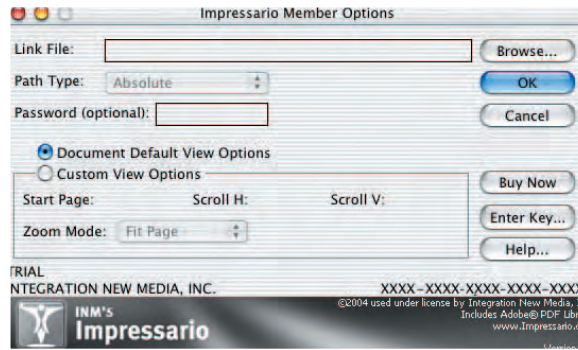
\$439 U.S.

protected documents, and, at the drag and drop of a behavior, the ability to provide word search on a document without leaving the Director MX environment.

As Director MX refuses to even look at a PDF document unaided, Integration's Impressario from Integration New Media (INM) adds a menu item to the "Insert" menu that allows you to link to a PDF as you would an MPEG video with the mpegadvance Xtra. Simply browse for the file, add a password if necessary, and voila. Once in the cast, you can add multiple PDFs to the stage and target them independently. It is entirely self-contained and doesn't even need Acrobat installed to work.

In a shrewd marketing move, INM has provided a selection of "open source" tools and files for download, including a

complete, Acrobat-style MIAW PDF reader for the unashamedly lazy. If you can't find what you need in the libraries that come with the Xtra, chances are that it will be in one of the online examples. A small selection of generic toolbars and buttons is also available that appears to have been carefully crafted to be totally inoffensive. I hated them the moment I set eyes on them. The free buttons are more insipid than ugly, but I can't deny that they do speed the process from concept to delivery and, when combined



with the pre-rolled behaviors, you can be throwing your PDFs about in no time.

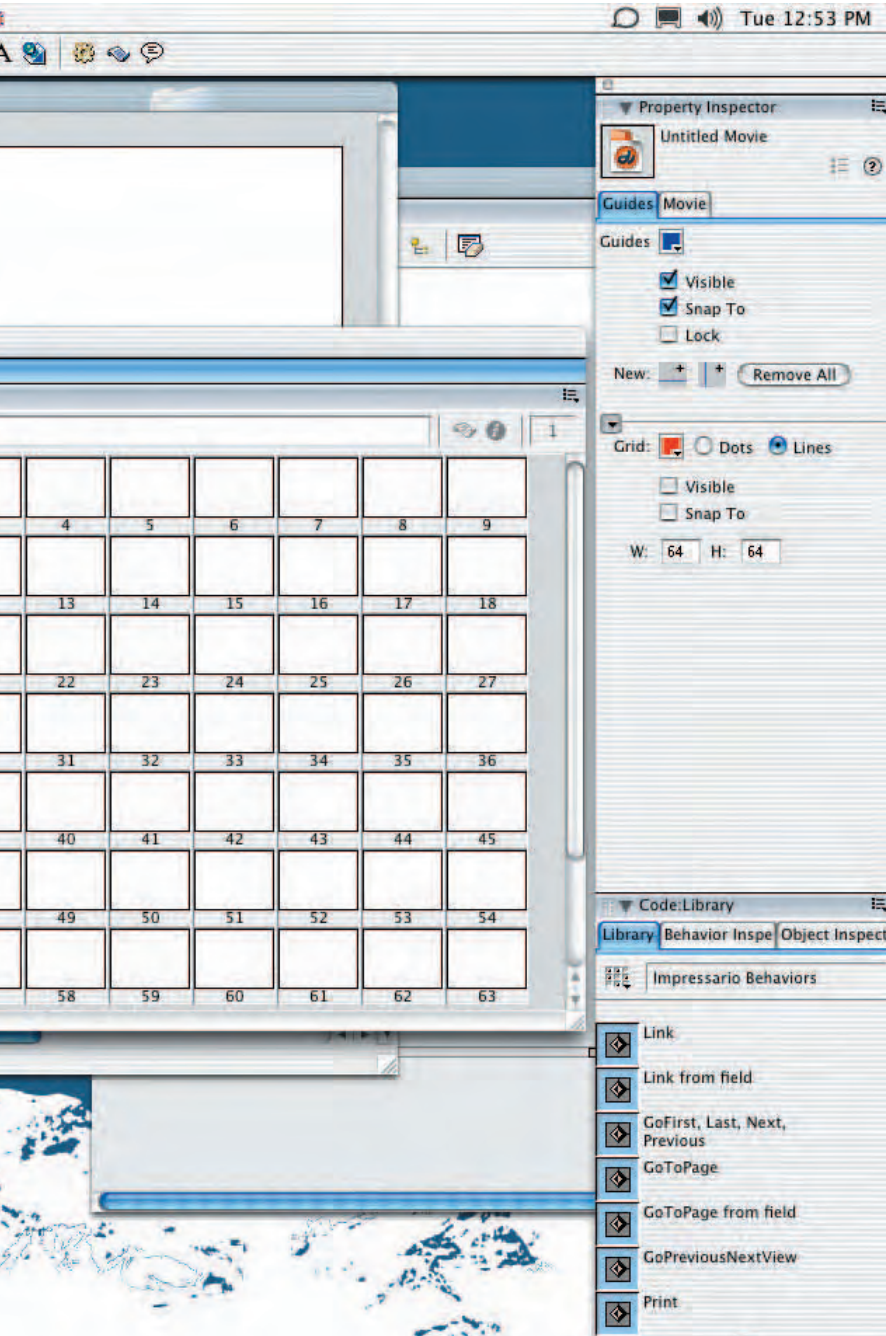
If, for example, you want to personalize PDF documents for individuals, you can use Impressario's PDF forms support and its "save" feature to write a PDF document to a local drive with the completed form data in place – instant personalization. This could be useful for creating contracts, personalized sales tools for companies, and a host of other applications.

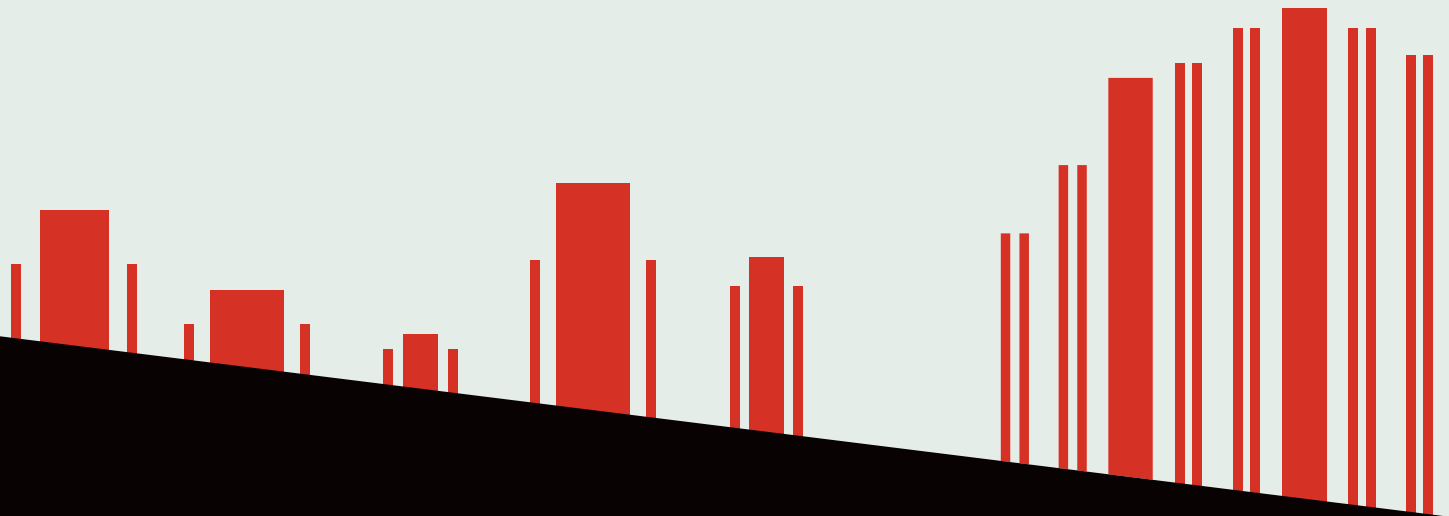
There is a dark side to all this, though. Unless you are willing to pay a surcharge, the licensing agreement states that you must make a small sacrifice to the sacred Lord Logo and wear his symbol at all times. But as Director and Quicktime have this caveat too, there's no great loss in waving the "Enhanced By INM's Impressario" flag on your project.

## Summary

I played with Impressario for a little while and liked it. I zoomed, scrolled, printed, and searched a few PDF documents and found it fast, solid, and simple to use. It is available for both Mac and PC, which particularly endears it to me, as I tend to spend a lot of time on Macs too. But, and it's a big but, it's a very specialized Xtra and that is reflected in the cost. I would have to think long and hard about whether the job really required these features or if there were a more economical solution (a mixture of Flash and XML would provide many of the accessibility features) before I bit the \$439 U.S. bullet for a one-time license that allows me to use the product on as many projects as I want. An educational license is available for \$299 U.S. Integration New Media does offer a discount to MMUG members (contact your preferred one for details), but even so it's quite a price to add to your already-stretched budget, unless, of course, it adds an essential value to your project. ☺

*Alec East is the media director at Tomorrow London (www.tomorrowlondon.com). alec@tomorrowlondon.com*





# moa city

by tab julius



I promised  
to take you to see a  
foreign city, one where you were  
not fluent in the language, and didn't know your  
way around, and then actually flew you there to visit, would you  
want me to just drop you off in the center of town and leave you there?



Well, that's sort of what has happened in my last two articles on Xtras (MXDJ, Vol. 2, issues 2-3) – space considerations precluded us from doing everything at once.

In my last article, I got you to the city. We went through the steps of building an Xtra and got it so that you could issue a Lingo command that would invoke a “Hello, World” alert box generated from within the Xtra.

And there we left it. No time to discuss what happens next, no time to talk about how to get data in or out of Director, no discussion of what powers you have access to. I just left you in the city and drove away.

Now we're going to rectify some of that. You've made it to MOA (Macromedia Open Architecture) City. What now?

## Structure

The first thing we need to do is to get oriented. Before we can be productive, we have a few basic chores to do.

All of MOA is organized into classes. It is very COM-like. If you need a particular function, you first have to get a pointer to the interface for the class that it exists in, and from there you can call the

function you want. The most common interfaces are typically acquired up front, during create time, and released at the end; then you just keep the pointers around and use them as needed.

The classes I most commonly acquire in advance are:

- **IMoaDrPlayer:** A pointer to the interface for the runtime engine, this lets you get a pointer to the active movie (the one currently executing). From the active movie you can work your way to the score, or to the casts, and so on. This is a Director-only interface.
- **IMoaMmUtils2:** This has some color support, but mostly it has functions to print messages to the message window – great for debugging.
- **IMoaMmValue:** This is an invaluable (no pun intended) interface – it lets you do conversions from the generic value format to most anything else (strings, integers, symbols, lists, and so on). Lingo is a loosely typed language, which means that a variable can be freely assigned to anything, it's not restricted to a certain type of data (as opposed to C/C++, where integers can only be assigned to variables declared to be integer variables; that would be an example of a more strongly typed language). To pull off the loose typing of Lingo, a variable really has two internal components: a struct consisting of the “type” component, a flag indicating what type of data it contains; and a “data” component, which would be a pointer to the data. All variables are assigned to these value structs; to assign a different kind of data, MOA just changes the type indicator and sets a pointer to the new data. That's MOA, though – you won't fiddle with a variable's innards yourself directly. What it all comes down to, though, is that all data passed in or out of the Xtra, to or from Lingo, is in the form of values, and you must use the

IMoaMmValue interface to make the conversions.

- **IMoaMmList:** Gives you the power to manipulate lists in Lingo. You can parse ones sent in, create ones to send back, and so on.

There are many others to work with, though, since there are well over 100 class interfaces. There are classes for manipulating files, memory, and streams, getting app info; creating dialogs; and more. Some are for obscure uses, some you may never use, some you may use a lot. It all depends on what your project entails.

The interfaces are documented in the DOCS section of the XDK. This is a good time to quickly review the files in the docs, since you'll be referencing them a lot from this point forward. The DOCS section is organized into folders, such as MOADG, MOREF, MMDG, MMREF, DRDG, DRREF, AWDG, and AWREF. Clearly there is a pattern here.

The first two or three letters refer to the category. MOA and MO mean MOA itself, MM is the multimedia functions, DR refers to classes specific to Director, and AW refers to classes specific to Authorware. The last two or three refer to the type of documentation – DG means “design guide” (more of a how-to discussion), and REF means “reference” (where you would go to look things up). So to look up a Director-specific interface, such as getting a pointer to the score, you would look in DRREF. Once in there, though, you will find over 30 files, so you will have to root around, but each folder has an index.htm file that you can start with and work your way from there.

So, how do you “get” an interface anyway? You do so by calling QueryInterface and telling it what interface you want, and where you want the pointer for it to be kept. As mentioned, I liked to get the common interfaces at Create time and

# “you've made it to moa (macromedia open architecture) city”

keep them around, so most of my QueryInterface calls are made in MoeCreate\_CScript, which is invoked when the Xtra is instantiated. First, though, it is necessary to declare variables for the pointers. The variables should go in the class instance variables section in CSCRIPT.H, as in:

```
EXTERN_BEGIN_DEFINE_CLASS_INSTANCE_VARS
(CScript)
    PIMoaMmValue    pMmValue;
    PIMoaMmUtils2   pMmUtils;
    PIMoaMmList     pMmList;
    PIMoaDrPlayer   pDrPlayer;
```

```
/* your other variables are defined here */
EXTERN_END_DEFINE_CLASS_INSTANCE_VARS
```

Now, any time your Xtra is called the functions will have access to these variables (they'll stay static). However, we need to set them up when the Xtra is instantiated and, as mentioned, we'll do that in MoeCreate\_CScript like so:

```
err = This->pCallback->QueryInterface(
&IID_IMoaMmValue,
(PPMoaVoid)&This->pMmValue);
err = This->pCallback->QueryInterface(
&IID_IMoaMmUtils2,
(PPMoaVoid)&This->pMmUtils);
err = This->pCallback->QueryInterface(
&IID_IMoaMmList,
(PPMoaVoid)&This->pMmList);
err = This->pCallback->QueryInterface(
&IID_IMoaDrPlayer,
(PPMoaVoid)&This->pDrPlayer);
```

I have never had the common interfaces (Value, Utils, etc.) fail, as they are common to most of MOA. If you want to be a good citizen you could check the return code in case there is an attempt to call your Xtra from some Macromedia product other than that for which it was intended. More likely you might run into problems with Director-specific interfaces, like DrPlayer, which would not be available in, say, Authorware. If you were trying to make a cross-product Xtra, you'd have to test (via IID\_IMoaAppInfo) to see what product you were running in and get the corresponding interface (or refuse to run).

Now you should have those interfaces for the life of your Xtra's instance. The

other key thing we need to do now is to remember to release them upon our destroy, which would be done as shown in Code I.

## Using the Interfaces

Now that you've got some interfaces, let's put them to use! In the last article we discussed the message table, which is where you define scripting commands and the parameters you can allow to be passed in, and we managed to make a call into the Xtra itself. But we didn't have a chance to look at how to access those parameters, or how to pass a return value back out.

The example function we had with parameters was:

```
/* FixCertainBug integer bugNum,
string fixName\n"
```

As a quick refresher, the asterisk in front means that it's a global command (you don't have to explicitly instantiate the Xtra) and requires two parameters – an integer and a string. The given names bugNum and fixName are for user readability only; they have no impact on anything and are, in fact, optional.

The base function we used last time for FixCertainBug looked like this:

```
MoaError
CScript_IMoaMmXScript::XScrpFixAllBugs
(PMoaDrCallInfo callPtr)
{
    UNUSED(callPtr);

    MoaError err = kMoaErr_NoErr;

    MessageBox(NULL, "Hello, World", "",
    MB_OK);

    return(err);
}
```

We'll now strip it down a bit to:

```
MoaError
CScript_IMoaMmXScript::XScrpFixAllBugs
(PMoaDrCallInfo callPtr)
{
    MoaError err = kMoaErr_NoErr;

    return(err);
} /* fix all bugs */
```

I mainly just took out the UNUSED(callPtr) line and the MessageBox call. UNUSED() is just a macro to keep the compiler from complaining because callPtr is passed in but is not necessarily used in all functions.

You might immediately think that one part is obvious – returning values back to Lingo, where at the end of the function there's a line:

```
return(err)
```

But, in fact, it's deceptive. That's not how you return a value to Lingo. What you are returning there is a result code that indicates to the Director runtime engine whether or not you were successful in processing the call. Err is set to kMoaErr\_NoErr by default. You would only send back something different if you wanted Director to throw an error showing that you could not process the command.

One example of where you might want to send something other than kMoaErr\_NoErr back might be if you failed to, for example, allocate memory. You could choose to handle this yourself, or you might wish to have Director throw an error. If so, you could send back kMoaErr\_OutOfMem. Or, possibly the user passed in a parameter that you don't allow; you could send back kMoaErr\_BadParam and let Director do the complaining.

The error codes are defined in three files: MOATYPES.H, MMTYPES.H, and DRTYPES.H. These are not only codes that you can send back, but also codes that you might receive as a result of some failed call to a function that you make internally to MOA. DRXLNGO.H says you can just send back \_ArgOutOfRange, \_OutOfMem, \_InternalError, and \_ValueTypeMismatch, but I've had success sending back other values.

You send back a result code to Lingo itself via the callPtr, which is a parameter that is a pointer to a structure that has all the pertinent information about Lingo's call to your Xtra – specifically methodSelector, resultValue, nargs, and pArgs.

- **methodSelector:** The index into your method table so you know what function of yours was called. We coupled this with an enum to get as far as



- invoking the right function in your Xtra, so we've done that part already.
- **resultValue:** What is returned to Lingo. This is what you would stuff with the value of whatever you want to send back – a string, a symbol, an integer, a list, and so on.
  - **nargs:** A count of the parameters passed in. If you set your message table to have a fixed number of arguments, you won't need this, but if the argument amount is variable (by using an asterisk as the last parameter on the parameter line), then you need to test to see how many were actually passed in.
  - **pArgs:** A pointer to the args, but you don't really use it directly. Instead, there's a macro we use to get at them called `AccessArgByIndex`.

The args passed in may or may not be preceded by an initial arg for the object. This would depend on whether or not the command is a global command. A child command, as discussed last month, where the Xtra would need to be instantiated, always passes in the object instance as the first parameter, often in the form of:

```
createFile object me, string fileName
```

This allows you to call it from Lingo after having instantiated it, as in:

```
CreateFile(fileObj, "C:\TEST.TXT")
```

What this means to you on the Xtra side is that you have to account for that object when you access your args. Failure to do so will mean that you get an object instead of an integer or string or whatever you expected.

I like to set up a define called `ARG_BASE` and set it to 0 or 1 depending on whether or not the commands are global (typically I have them all one or all the other; I usually don't mix globals and children in the same Xtra, although there's no technical reason why not). Then I can just reference the parameters in the order I'm expecting them, without worrying about remembering to account for any preceding object parameter. If I choose to convert all of them from global to child (or child to global), I only need to change the message table and the

define, and not all the code I have internally.

I might define:

```
#define ARG_BASE 1
```

(as you would for an Xtra that had child commands) and then could just reference each arg that I was expecting as `ARG_BASE + 1` for the first arg, `ARG_BASE + 2` for my second arg, etc. Try it; you'll see that it helps.

At any rate, we need to look at how to access args, which are all in `MoaMmValue` form. As I mentioned earlier, the `IMoaMmValue` interface provides functions for converting to and from values. Normally it is your responsibility to release any values you create (because they do allocate memory). If you were to create a value for a symbol for the purposes of supplying it to some function, it would normally be your responsibility to release that value when you were done. The one notable exception to this has to do with parameter passing through the `callPtr`, which is what we're discussing here. You do not release the values passed in to you, nor do you release the value that you send back via `resultValue`. If you do release them, Director will die a most horrible death upon returning from a supposedly successful call to your Xtra, and you would otherwise have quite a time figuring out why.

The `AccessArgByIndex` macro lets you get your fingers on the parameters. The `FixCertainBug` example function takes two parameters, an integer `bugNum` and a string `fixName`. We could process the call as shown in Code II.

A couple of points here: `pObj->pMmValue` is the pointer to the `IMoaMmValue` interface that you got during create time. An interesting distinction between create/destroy time and "the rest of the time" is that in create/destroy, when you reference your class instance variables you do so as `This->pMmValue` but in the rest of the program you use `pObj->pMmValue`. This is a lot more straightforward than it used to be, believe me. I mention it because the compiler will not allow `pObj->` to be used from create/destroy, and it won't allow `This->` to be used anywhere else. FYI.

Also, I added `ARG_BASE +` to when I was referencing the arg number, which is

strictly unnecessary because `ARG_BASE` is 0, but it's a good habit and saves trouble later, as we have already discussed.

Finally, you will notice that I didn't release the value at any time. When we do the `AccessArgByIndex` we are not creating a value, we are simply accessing one that is already created and assigning it to our internal variable, hence no need to release it. Likewise on the `resultValue`, Director will release that as well. As I said, this is basically the only time this is the case. Normally you will always be responsible for releasing the values you create unless otherwise pointed out in the documentation for an interface.

So, we have now accessed our passed-in parameters and also returned values back! The user on the other end would receive back an integer, 1 (`TRUE`).

Since we were also passed in a name for a fix, we can practice printing the information out to the message window. The interface `IMoaMmUtils2` has functions `PrintMessage`, `PrintMessage1`, `PrintMessage2`, `PrintMessage3`, and `PrintMessage4`, all of which print messages to the message window. The difference between them is how many optional parameters they take – `PrintMessage` by itself parses no extra parameters. Parameters are given in C format, as you might do with `sprintf`:

```
pObj->pMmUtils->PrintMessage2("Now  
fixing bug %d, name: %s\n",  
                                whichBug,  
(MoaLong)&fixInfo[0])
```

It's not a 100% carbon copy of `sprintf` form; for instance, you can't pass a string in directly but you have to give the pointer to it like I did here – but it's close enough.

As you have undoubtedly noticed, there are variable types like `MoaLong`, `MoaChar`, and so forth. These are generally mapped 1:1 to regular longs, ints, and chars. You can, in fact, use them interchangeably, but they do provide a layer of abstraction. I usually try to use the `MOA` form when I'm paying attention. At this point in the XDK life cycle there are no conversion functions, and I freely use `MoaLong` to pass to C functions or whatever, but that may change at some point in the future.

## Summary

I will close with a couple of quick notes



on IMoaMmValue and a comment on the naming of interfaces within MOA itself. First, everything we've said so far is predicated on knowing ahead of time what type of data (e.g., integer) a value is. What if you don't know what kind of data it holds? IMoaMmValue has a function called ValueType() which, if passed a value, will return kMoaMmValueType\_XXXX where XXXX would be Void, Integer, Symbol,

Member, and more (the complete list is in mmTypes.h). As a shortcut, though, if you allow a parameter to be either an integer or a string, rather than test for the valueType, you can try to convert it to integer and then check the error code. If it fails, try to convert it to string, and so on.

You may note that I referred to IMoaMmUtils2; what happened to IMoaMmUtils1? Over time, the XDK

has evolved, and the class interfaces are generally backward compatible when they are expanded.

Occasionally, though, they are not, and when that has happened Macromedia has left the original class alone so as to not break any existing Xtras. Instead, they created a new variant that supersedes the old one, usually appending a 2 to the end (there are no 3s yet to my knowledge). So the original IMoaMmUtils is obsolete and you should use IMoaMmUtils2 instead. IMoaFile2 and IMoaStr2 are other examples of this. If I remember correctly, all the value conversions were originally in IMoaMmUtils but were later migrated out into IMoaMmValue, hence the reworking of IMoaMmUtils.

You should now be in a position to start experimenting with passing information in and out of an Xtra, and maybe even acting on that information. Next time we'll delve a little further into MOA and learn a little bit more. Enjoy! ☺

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

code I

```
if (This->pMmValue)
{
    This->pMmValue->Release();
    This->pMmValue =NULL;
}

if (This->pMmUtils)
{
    This->pMmUtils->Release();
    This->pMmUtils =NULL;
}

if (This->pMmList)
{
    This->pMmList->Release();
    This->pMmList =NULL;
}

if (This->pDrPlayer)
{
    This->pDrPlayer->Release();
    This->pDrPlayer =NULL;
}
```

code II

```
#define ARG_BASE 0

MoaError CScript_IMoaMmXScript::XScrpFixAllBugs(PMoaDrCallInfo
callPtr)
{
    MoaError err = kMoaErr_NoErr;
    MoaMmValue value;
    MoaLong whichBug;
    MoaChar fixInfo[256] ={};

    // Get at the integer for the bug num
    AccessArgByIndex(ARG_BASE + 1, &value);
    pObj->pMmValue->ValueToInteger(&value, &whichBug);

    // And the string for the name
    AccessArgByIndex(ARG_BASE + 2, &value);
    pObj->pMmValue->ValueToString(&value, &fixInfo[0],
sizeof(fixInfo));

    // Now fix accordingly (this part you write :)

    // And return success
    pObj->pMmValue->IntegerToValue(TRUE, callPtr->resultValue);

    return(err);
} /* fix all bugs */
```

## The Ultimate 3D Flash Tool

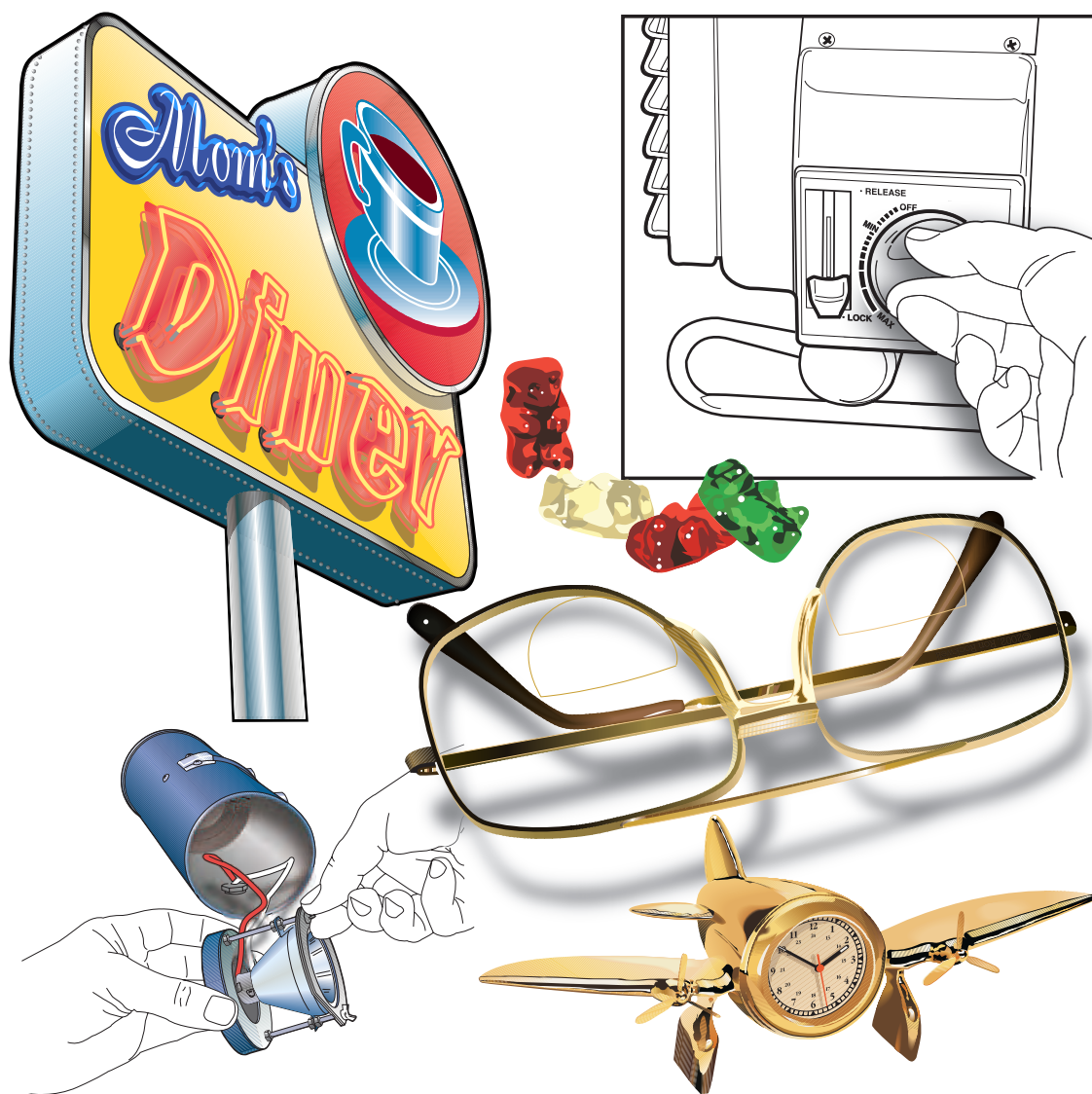


- Powerful Modeling and Animation Tools
- High-Quality, Low-Bandwidth Vector and Raster Output
- Tight Integration with Macromedia Flash (TM)
- Affordable

Add a third dimension to your Flash work  
[www.eraim.com/mxdev.asp](http://www.eraim.com/mxdev.asp)

# Line Art

Technical illustrator Ron Rockwell uses Macromedia FreeHand to create renderings, instruction manuals, catalogs, brochures, print ads, and Web graphics such as these. He is a Team Macromedia member for FreeHand as well as MXD's FreeHand editor, and is currently working on an instruction course for FreeHand MX. You can see more of his work at [www.nidus-corp.com](http://www.nidus-corp.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](http://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.