


# MX


developer's journal


preview issue

# Toppling the...

 Weaving Works into Tools

 Understanding Levels in Fireworks

 The Logo Browser Project

 ColdFusion Components Under a Red Sky

 Introducing Director MX

# T o w e r o f B a b e l

with  MX

*How to produce the intense application:*

# BabelChat

\$5.99US \$6.99CAN 01>



0 71486 02978 6

This is for you.

The hardcore, the dedicated, the enthusiast. The one with a passion for figuring things out. The one who builds things ... and takes them apart again.

You give us new ways to see things. You show us things we never saw in the first place. You make things better. You make us better.

MX 2004 is here. [Run with it.](#)



Now available.  
**Macromedia MX 2004.**

[www.mx2004.com](http://www.mx2004.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.

# contents

preview issue

# MX

developer's journal

## 7 From Great Experiences to Great Business

Norm Meyrowitz, Macromedia's president of products, sets the scene for MXDJ's upcoming role as a strategic part of the community of MX customers, underlining Macromedia's commitment to maintaining an active dialogue with customers, "because the success of our products was built on learning from and teaching one another."



## 8 Welcome to Macromedia MX

John Dowdell provides a 35,000 ft. view for MX developers.



## Toppling the Tower of Babel with MX

Giacomo 'Peldi' Guilizzoni uses Macromedia Flash Communication Server components and some minor ColdFusion MX and Macromedia Flash Communication Server code to create "BabelChat" – a real-time application that automatically translates chat from one language to another.

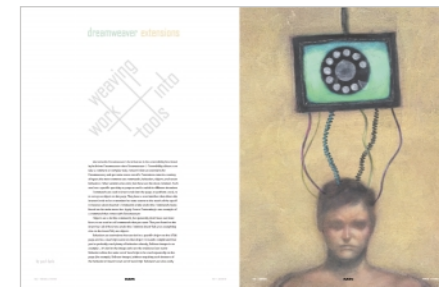


# 22

## Understanding Levels

Kleanthis Economou puts Macromedia's graphics design production program through its paces.

# 36



## 46 ColdFusion Components Under a Red Sky

Raymond Camden, coauthor of the "Mastering ColdFusion" series (Sybex), examines how CFMX 6.1 will now run on the latest version of Windows.



## 50 Introducing Director MX

Miriam Geller provides a guided tour of some of the newest features.



# 50



Dreamweaver Extensions: Weaving Work into Tools  
Paul Davis makes extensions accessible.

# 12



## The Logo Browser Project

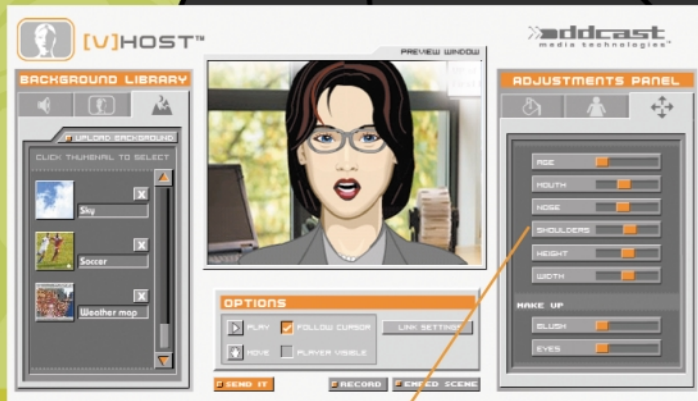
Joe Sparks helps you build an interactive logo browser – a simple, interactive presentation with FreeHand MX that you can publish and view as a Macromedia Flash movie.



# 38



# CREATE TALKING CHARACTERS IN A FLASH.



**oddcast**  
media technologies™

The [V]Host Studio is a tool that enables professionals to design and deploy interactive talking characters in minutes. Output as a .swf file, the characters can be embedded in any Flash movie. Record a voice or use our Text-To-Speech features to read dynamic text, in 14 different languages, then watch the characters breath, talk and come to life. Non-programmers can learn to use the [V]HostT studio in minutes. Developers can use our ActionScript and JavaScript API's to make the character interact with users through wide range of actions and behaviors.

Winner of the following awards:  
European Excellence in Digital Media Award 2003  
Silver Medal, Spin Awards 2003  
Bronze Medal, I.D. Magazine, Interactive Design 2003  
Best Application award, FlashForward 2002

**www.oddcast.com**  
**Call (212) 375-6290**  
**for developer's kit**

- Main Features:**
- Export as swf
  - Lip Synchronization
  - Dynamic Text-To-Speech (with no Plug-in!)
  - Record by mic or telephone.
  - Advanced ActionScript and JavaScript APIs.
  - Advanced content management and workflow features
  - Usage and performance reports
- Personification features:**
- Set Age
  - Set Color (Skin, Eyes, hair)
  - Set Size (Shoulders, head, nose, mouth)
  - Set Clothing
  - Set Accessories
  - Set Background
  - Set Audio

**Group Publisher** Jeremy Geelan  
**Art Director** Louis F. Cuffari

**EDITORIAL BOARD**  
**Dreamweaver Editor**  
Dave McFarland  
**Flash Editors**  
John Tidwell, Jesse Warden  
**Fireworks Editor**  
Kleanthis Economou  
**FreeHand Editors**  
Sandee Cohen, Louis F. Cuffari  
**ColdFusion Editor**  
Robert Diamond  
**Director Editor**  
Gary Rosenzweig

**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**

**EDITORIAL**  
**Executive Editors**  
Jamie Matusow, Gail Schultz  
**Editors**  
Jean Cassidy, Nancy Valentine,  
Jennifer Van Winckel  
**Online Content Manager**  
Alan Williamson

**Subscriptions**  
For subscriptions and requests for bulk orders, please send your letters to Subscription Department [subscribe@sys-con.com](mailto:subscribe@sys-con.com). Cover Price: \$5.99/issue. Domestic: \$29.99/yr. (12 Issues) Canada/Mexico: \$49.99/yr. Overseas: \$59.99/yr. (U.S. Banks or Money Orders) Back Issues: \$12.00/ea.

**Editorial Offices**  
SYS-CON Media, 135 Chestnut Ridge Rd., Montvale, NJ 07645  
Telephone: 201 802-3000  
Fax: 201 782-9600  
To submit a proposal for an article, go to <http://grids.sys-con.com/proposal>.

**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. Postmaster: Send address changes to: MX Developer's Journal, SYS-CON Publications, Inc., 135 Chestnut Ridge Road, Montvale, NJ 07645.

**Worldwide Newsstand Distribution**  
Curtis Circulation Company, New Milford, NJ.

**For 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)

**Copyright © 2003** 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. For promotional reprints, contact reprint coordinator Carrie Gebert, [carrieg@sys-con.com](mailto:carrieg@sys-con.com).

**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.



# From Great Experiences to Great Business

*The key feature in MX 2004 is...you*  
by norm meyrowitz

Macromedia and its developer community have a unique bond based on a mutual fascination with what the Internet could be, a desire to create great experiences, and a need to keep pushing the limits. The active dialogue that takes place between Macromedia and its customers pushes both sides forward.

Together, we are building a world where every digital interaction – whether in the living room, the office, the beach, or the car – is a smart, simple, efficient, and engaging experience. We believe it will happen. It's who we are and why we are here. Macromedia, along with you, our customers, will help make the future better, one interface at a time. Together, with our software and your expertise, we are going to convert great experiences into great business.

Macromedia collaborates deeply with our customers, because you live and breathe our tools and are in the best position to challenge us to make them better. We respond with the features you ask for and try to anticipate trends and add features that will inspire you to take experience to new heights. You build things that amaze us, and challenge us to put even more features in future versions of the products.

**MX Developer's Journal** is an important venue in which we can converse, and a testament to both the popularity of the products in Macromedia Studio MX 2004 and your expertise.

This first issue of *MX Developer's Journal* launches at the same time that you will begin to experiment with our new MX 2004 product line, providing a perfect place to learn the tips and tricks of the new MX 2004 products together.

The MX 2004 products are evolutionary in their approach, as you will see when you dig deeper into each one. Dreamweaver MX 2004 offers enhanced Cascading Style Sheets (CSS) support, cross-browser validation, built-in graphics editing, and updated support for

ASP.NET, PHP, and ColdFusion server technologies. Flash MX 2004 has new timeline effects, easy import of Adobe Illustrator and PDF content, and CSS support. Fireworks MX 2004 runs faster and delivers a broad array of new design tools and effects. Flash MX Professional 2004 enables advanced content, applications, and video experiences to be created using a forms-based metaphor as an alternative to the traditional timeline development.

We've also expanded the MX universe beyond products with MX Elements and Halo. MX Elements are a series of interface building blocks with a new generation of interactive design patterns embedded – making it faster to build interfaces. These elements bring together a range of technologies including components, templates, CSS style sheets, and behaviors. MX Elements, by default, have a distinctive new look and feel we call Halo – we think it will make the digital world a brighter, friendlier place.

Each MX 2004 product provides a rich palette from which the best online experiences will be drawn. Each product is built to help you get more done, to automate routine tasks, and to free your time to create. We think we've made major strides with these releases and look forward to hearing from you about what you'd like to see us take on next time. We're already on the road to planning for the next version – let us know where we should meet up for next time.

With this amazing set of tools, the exhaustive information on macromedia.com, the vibrant community sites, and great new resources such as this magazine, our customers have a broad palette of information from which to create great digital experiences that fuse our community and shared knowledge with their creativity and passion.

We can't wait to see what you're going to do with these new products. ☺

*As president, Macromedia products, Norm Meyrowitz oversees development and marketing for all Macromedia product divisions. Meyrowitz is a recognized authority on the evolution of Web development software and media technology for the Internet. Through his experience at Macromedia, Meyrowitz has overseen the teams creating a vast array of multimedia and Web development products, including Director, Shockwave, Dreamweaver, and Macromedia Flash. [nmeyrowitz@macromedia.com](mailto:nmeyrowitz@macromedia.com)*

# Welcome to Macromedia MX

*The platform that isn't a platform*  
by john dowdell

**W**elcome to the first issue of *MX Developer's Journal*. I hope the upcoming resources are of great use in your work in creating presentations and applications for your clients, to further their audiences' needs.

Macromedia MX is sort of an odd bird among Web technologies. Similar things are often described as "platforms," but I'm not quite certain that we fit into such a description. Here's why...

The tools in Macromedia MX can produce a variety of final outputs – from presentations you render yourself as hard deliverables (such as going to paper through FreeHand MX), to standalone packages of OS-specific executable code and media (as in Director MX), to document descriptions that are rendered in the audience's application of choice (using Dreamweaver or ColdFusion to deliver to the various browsers), to files of media and instructions that are rendered on the audience's machines in a

single predictable engine (the SWF-producing tools).

The term "platform" usually refers to producing a single set of instructions for a single OS or device. But the MX toolset works across browsers, operating systems, and device types. It's one workflow that can end up going to various destinations. It's not locked in to a single type of use.

So the overall toolset isn't what we usually call a "platform," because it delivers your work in a variety of ways. But what about its individual parts?

Is Dreamweaver a platform? People create text documents in Dreamweaver to render within various browser implementations, even adding styling instructions and interactivity, which most browsers today can interpret. It also creates text files for servers to process, such as ColdFusion templates. But Dreamweaver just as easily produces PHP templates, ASP templates, .NET templates, Java templates, or other types of files – it's not locked in to a single destination. Dreamweaver is a common development environment to get out to a range of different platforms, but isn't really a platform itself.

What about ColdFusion? Maybe...It lives on the server, and people have certainly created many applications with it over the years. The ColdFusion server extends the Java 2 Enterprise Edition with the fastest way to develop server-side applications, so it's certainly bound to certain types of systems. But the final deliverable will work in any browser. Because J2EE abilities are common on most servers today, the ColdFusion installation isn't even as locked-in as most other things called "platforms" these days. It works atop a variety of processors, operating systems, and Web servers. ColdFusion installations float above many true platforms, providing a service layer that doesn't constrain you to a particular operating system or device type.

The Macromedia Flash Player? This probably has the best chance of being called a "platform," because it's a pre-

SYS-CON MEDIA  
**President & CEO**  
Fuat Kircaali  
**Vice President, Business Development**  
Grisha Davida  
**Group Publisher**  
Jeremy Geelan  
**Technical Director**  
Alan Williamson

ADVERTISING  
**Senior Vice President, Sales & Marketing**  
Carmen Gonzalez  
**Vice President, Sales & Marketing**  
Miles Silverman  
**Advertising Sales Director**  
Robyn Forma  
**Director, Sales & Marketing**  
Megan Ring  
**Advertising Sales Manager**  
Alisa Catalano  
**Associate Sales Managers**  
Carrie Gebert, Kristin Kuhnle

PRODUCTION  
**Production Consultant**  
Jim Morgan  
**Lead Designer**  
Louis F. Cuffari  
**Art Director**  
Alex Botero  
**Associate Art Director**  
Richard Silverberg  
**Assistant Art Director**  
Tami Beatty

SYS-CON.COM  
**Vice President, Information Systems**  
Robert Diamond  
**Web Designers**  
Stephen Kilmurray, Christopher Croce  
**Online Editor**  
Lin Goetz

ACCOUNTING  
**Accounts Receivable**  
Kerri Von Achen  
**Financial Analyst**  
Joan LaRose  
**Accounts Payable**  
Betty White

EVENTS  
**President, SYS-CON Events**  
Grisha Davida  
**Conference Manager**  
Michael Lynch  
**National Sales Manager**  
Sean Raman

CUSTOMER RELATIONS  
**Circulation Service Coordinators**  
Niki Panagopoulos, Shelia Dickerson,  
Edna Earle Russell  
**JDJ Store Manager**  
Rachel McGouran



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.*

# MXDJ Section Editors

## Dreamweaver

Dave McFarland

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



## Flash

John Tidwell

Having started and created FlashCore, the largest dynamic media creative and technical community in the world, and then a successful company producing technology trade shows and conferences, John is currently CEO of Perennial Light, Inc.



## 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

Sandee Cohen

A teacher with New School University in New York City, and also the computer graphics coordinator for the school's Computer Instruction Center, Sandee is the author of 10 versions of the Visual Quickstart Guides for FreeHand, Fireworks, InDesign, and Kai's Power Tools, and has also been a speaker for Seybold Seminars, MacWorld Expo, and PhotoPlus conferences.



## 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.



## ColdFusion

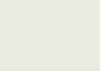
Robert Diamond

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)

## Director

Gary Rosenzweig

Founder and owner of CleverMedia, Gary is the creator of many Shockwave and Flash games, and the author of Director and Flash books.



dictable engine you can build upon within the final delivery devices. But, even more so than ColdFusion, it floats as a service layer above many browsers, operating systems, and device types. There are no system-level hooks accessible to a SWF file... the Macromedia Flash Player is platform-neutral, and fulfills the promise of client-side Java.

## Macromedia Flash Player

Let's take a closer look at the Macromedia Flash Player for a moment. How does it differ from other client-side engines, such as the browsers, client-side Java, .NET's Common Language Runtime, and other distributed engines?

There are three angles that make the Macromedia Flash Player unique:

### Cooperation

The little Macromedia Flash Player is happy on various operating systems and devices. There's no need to shoehorn the audience into a particular configuration to use it. Mac, Win, Linux... computer, handheld, mobile phone... the Macromedia Flash Player provides a local service layer that will work with the audience's choice of technologies.

JavaScript is great, but you have to worry about what will be rendering the results of the interactions, which is even more of a pain if you want to go beyond text and graphics to audio and video rendering, or want to make live network requests. For devices, the Flash Player is small enough so you rarely have to worry about additional "micro" or "compact" versions of the engine... it's the same across device form. Being self-contained and predictable lets you confidently hook into more systems.

More, it will cooperate with a variety of server-side mechanisms. It doesn't matter if you prefer to use PHP on your own machines, ASP or .NET, handrolled Java, whatever... just so long as the server can handle live XML transfers and SOAP requests, the Flash Player will cooperate with your server as well as with the audience's hardware choices.

### Velocity

It's one thing to make and distribute a better mousetrap, because each person can make his or her own individual deci-

sions to use that mousetrap or not. But it's much harder to make a new type of radio transmitter, telephone system, or television signal. That's because you have to rely on enough other people choosing the same technology in order to have it work for any of them. Network effects are a definite gating factor in distributed technology.

The Macromedia Flash Player is the single most widely used software on the Internet. It is bigger than any browser... certainly bigger than any operating system. (Look at the "Zeitgeist" section of Google sometime, and compare it to the Flash Player census... people do slowly improve their browsers, but new operating systems lag far behind browsers, and hardware changes are even slower than that.)

What's more is that new versions of the Player are adopted more quickly than other technologies. It's a small piece of code to download – it does not force the audience to change their existing UI or work habits – and its ongoing use is driven by the most popular sites on the Web. A new version reaches majority consumer viewership within a year. It's the fastest way to reach the largest audiences with new capabilities.

Large downloads, or those that require investment or changes of habit on the part of the audience, don't have the velocity that this small, platform-neutral engine has. If you can't force installations on your audience's machines, then the Macromedia Flash Player gives you the fastest route to new types of abilities.

### Security

Many platforms are integrated top-to-bottom, tying tightly into the operating system. This summer we saw the disastrous results of even such simple architectural choices as allowing e-mail attachments to execute native code in response to clicking a message. Lock-in to a single system can offer great power, but in a networked world it just doesn't work to continually try to patch over others' abuses of such power.

Keeping a sandbox around a certain service layer is intrinsically safer. Instead of deep top-to-bottom control, limiting abilities to a certain layer of services over a wide range of environments is naturally more secure.

–continued on page 48

## overview

# 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.

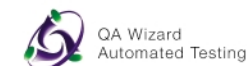
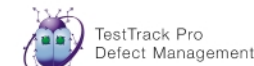
Automatically notify team members of changes to source files—push changes through your organization.

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

Weaving  
Work  
Tools  
into

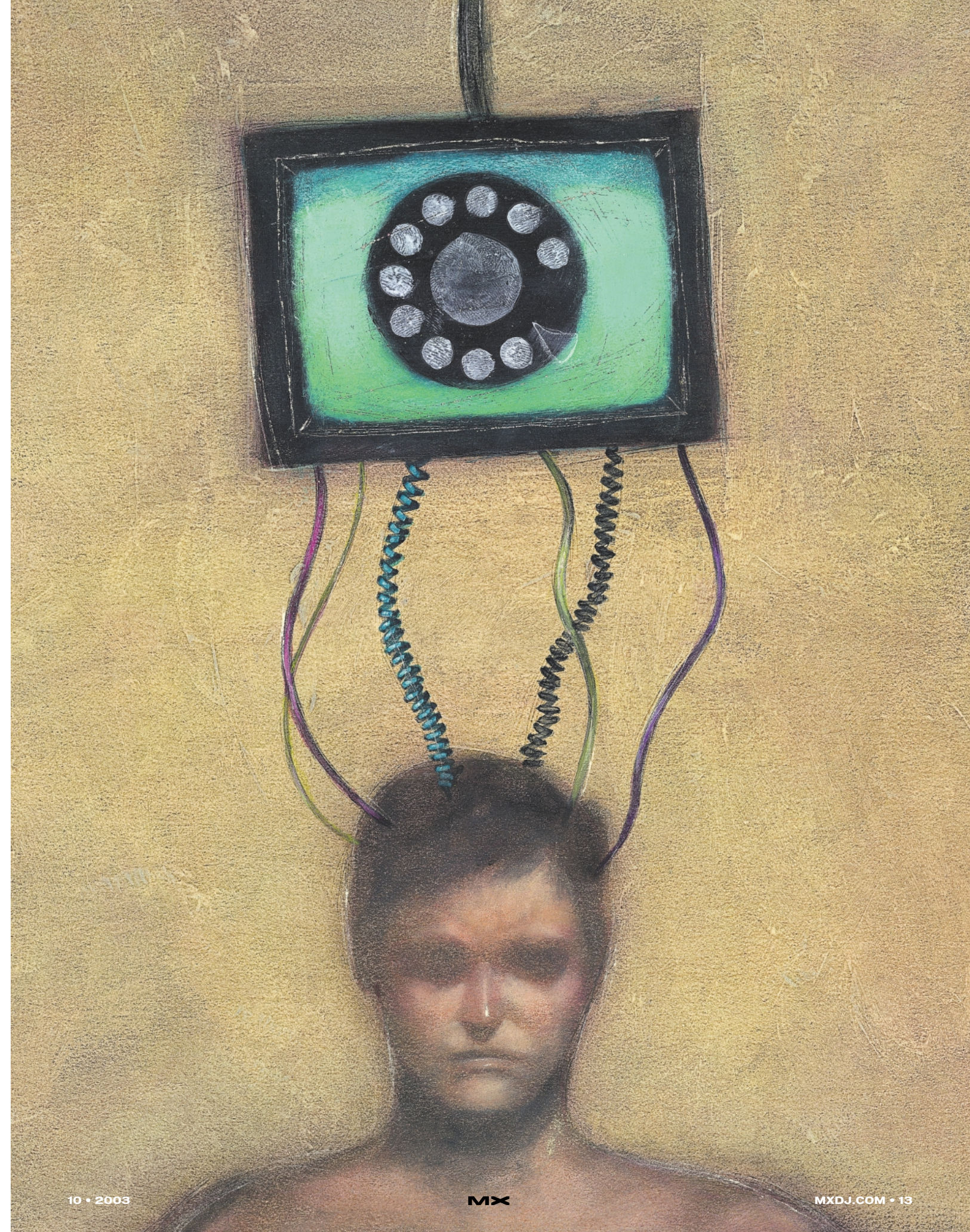
Macromedia Dreamweaver's best feature is the extensibility functionality built into Dreamweaver since Dreamweaver 2. Extensibility allows us to take a common or complex task, convert it into an extension for Dreamweaver, and get some reuse out of it. Extensions come in a variety of types; the most common are commands, behaviors, objects, and server behaviors. Other varieties also exist, but these are the most common. Each one has a specific specialty or purpose and is usable in different situations.

Commands are used to insert code into the page, to perform a task, or to set up an object on the page. They have a user interface that allows the inserted code to be customized to some extent to the needs of the specific instance when inserted. Commands reside under the Commands menu found on the main menu bar. Apply Source Formatting is one example of a command that comes with Dreamweaver.

Objects are a lot like commands, but generally don't have user interfaces or are used to call commands that you want. They are found on the Insert bar. All of the icons under the Common Insert Tab (as is everything else on the Insert Tab) are objects.

Behaviors are extensions that are tied to a specific object on the HTML page and to a JavaScript event on that object. It sounds complicated but you've probably used plenty of behaviors already; Rollover Images is an example – it's tied to the image and uses the onmouseover event. Behaviors allow the same set of JavaScript to be used repeatedly on the page (for example, Rollover Images) without requiring each instance of the behavior to have its own set of JavaScript. Behaviors are also easily

by paul davis



edited because you can select the object they are attached to and they show up in the behavior panel and double-clicking on them will bring up the behavior and the values that were used in the behavior. Behaviors also utilize a user interface to enter the needed values for the behavior.

Finally, server behaviors are similar to behaviors in some regards but are also similar to commands. Server behaviors make the complex easy when it comes to using a database to manipulate the content of the page, inserting a form's data into a database, updating

database records, deleting database records, or even handling security for logins and logouts. Dreamweaver supports Active Server Pages (ASP), ColdFusion (CF), JavaServer Pages (JSP), PHP, and MySQL, and it even supports ASP.NET for server languages it will have server behaviors for. Examples of server behaviors include Repeat Region, Insert Record, Delete Record, and User Authentication.

In Image 1 the location of the extensions are noted by numbers. 1 is for commands, 2 for objects, 3 for behaviors, and 4 for server behaviors.

Okay, now we've gone over the various aspects of the extensibility capabilities of Dreamweaver, even got in a sneak peek at the upcoming Dreamweaver MX 2004 and one of the new Kaosweaver site, but you might be thinking it would take rocket science to create an extension. Well, let's walk through the creation of a command and you'll see it really isn't difficult. First we need to make some preparations in order to make the creation process easy and quick. The fastest way to set up the site in Dreamweaver is to find a file already on your hard drive in the folder we want the site to point to. To do this, we'll need a common file for everyone so we'll use the Breadcrumbs extension from Kaosweaver to do this.

Download the Breadcrumbs extension from [www.kaosweaver.com](http://www.kaosweaver.com) or from

Macromedia's exchange at [www.macromedia.com/cfusion/exchange/index.cfm](http://www.macromedia.com/cfusion/exchange/index.cfm).

Install the extension by either double-clicking on it or dragging the icon over the Exchange Manager icon in your Dreamweaver folder. Once it is installed, we need to do a search for a file called "breadcrumbs.html" on the drive where Dreamweaver is installed. Those who have Windows 2000 or XP will have something similar to this for a path:

```
C:\Documents and
Settings\userName.CompName\Application
Data\Macromedia\Dreamweaver\Configuration\
```

The important thing isn't to match my path, but to get the path to the file because we need it to make a site in Dreamweaver. Now, armed with the path information, load Dreamweaver and create a new site. This can be done under the Manage Sites menu option found under the Site menu option from the main menu bar by clicking on New when the Managing Sites dialog box appears. Name the site myConfig; we need to enter the path right under the site name (the entry is called Local root folder.) After that, click OK and the site will generate the cache for the new site. For those who don't have Windows 2000, XP or Mac OSX will have a few thousand files to go through so it may take a little time, which is normal.

Now that the site is created, you can see a folder list that includes a folder called commands. You can navigate to this folder and open the breadcrumbs.html file

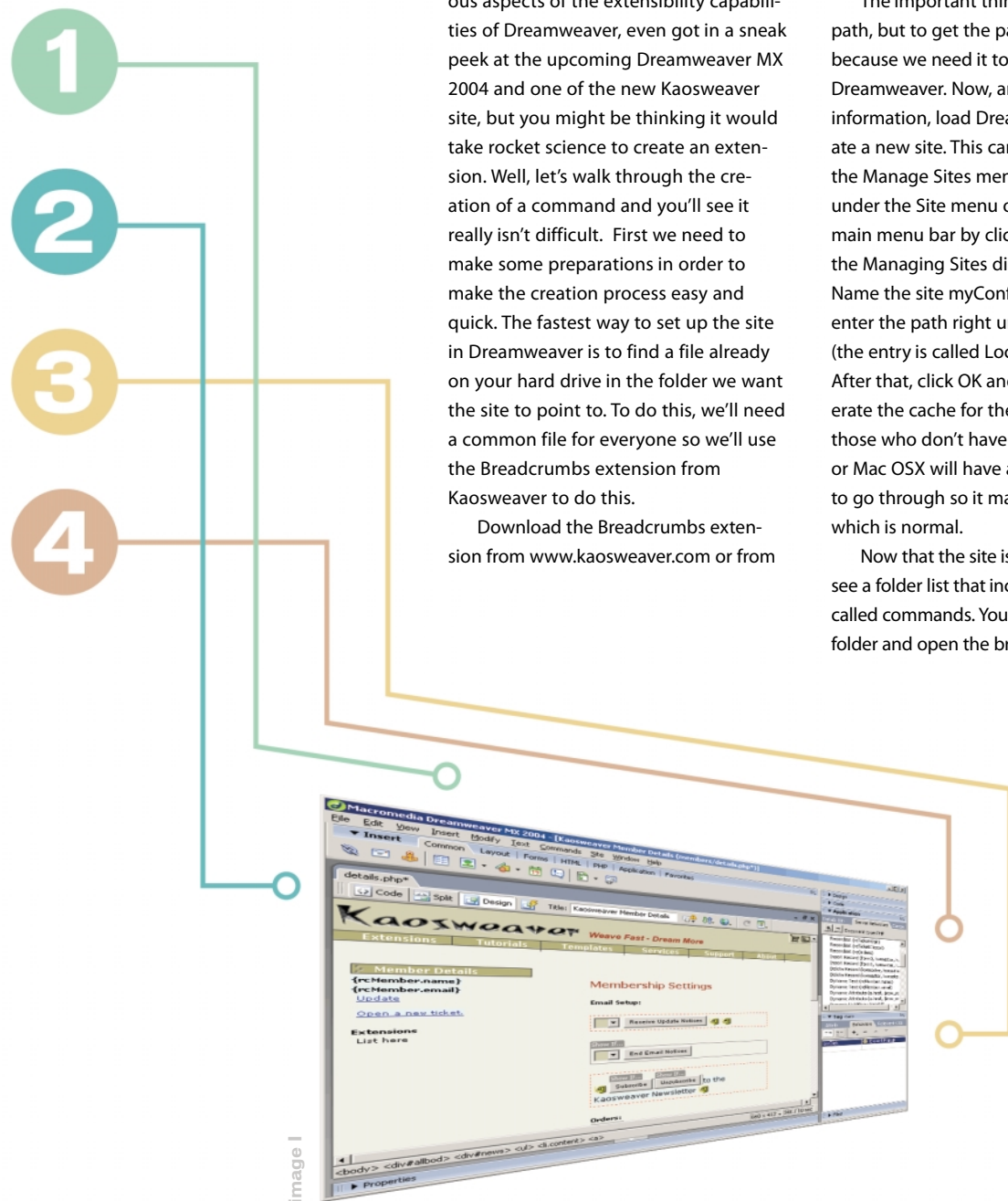


Image 1

## WHERE COLDFUSION EXPERTS HOST



*"CFDynamics is everything I look for in an ISP. With more features and fewer restrictions than similar provider plans, consistent top-notch customer service and incredible attention to detail, CFDynamics immediately comes to mind whenever CF developers ask me to recommend a host that delivers great service at a decent price."*

Simon Horwith

Macromedia Certified Instructor  
Certified Advanced ColdFusion  
MX Developer  
Certified Flash MX Developer

At CFDynamics, we put the warmth back into client relationships. Our technical support staff consists of personable and intelligent CF system administrators who really enjoy solving problems. We don't employ machines to do our telephone work; and we never mix sales with service.

At CFDynamics, ColdFusion web hosting is our complete focus. You'll speak with a real system administrator every time you call. You'll also enjoy real benefits, not just promotional tidbits. With accounts starting at \$16.95/month, our clients enjoy free SQL Server Access, free account setup, unlimited email accounts and a 30-day money-back guarantee. Real service. Real satisfaction. Forget about cold computer voices and confusing phone loops - let CFDynamics show you the warmth in ColdFusion hosting.

**CFDynamics**  
A Division of Konnections Inc.

Are you going to MAX in November 2003?

Take the:

**MAX  
Challenge!**  
2003

[www.cfdynamics.com/maxchallenge](http://www.cfdynamics.com/maxchallenge)

866 - CFDYNAMICS

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



“the word *function* tells dreamweaver that this is a set of code it can access by the name”

image II

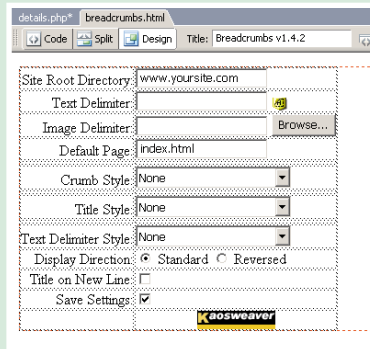


image III

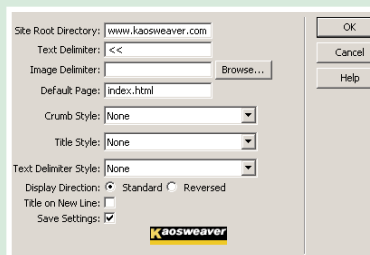


image IV

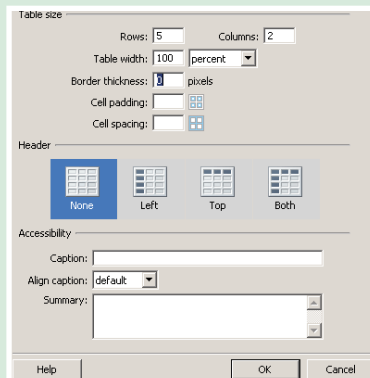


image V

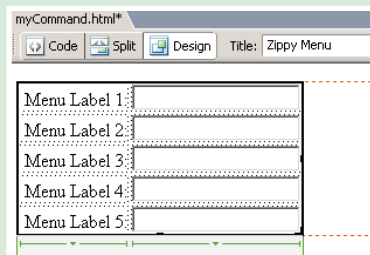
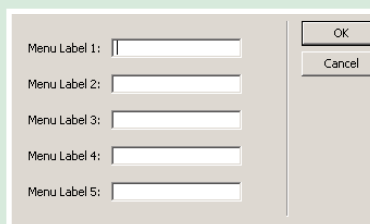


image VI



we looked for in the first step. It will look similar to Image II, depending on your OS.

Looks strangely familiar doesn't it? Just like an HTML page with a form. However, when you run Breadcrumbs the extension looks like Image III.

The first thing to notice is the extra buttons on the right that weren't in the HTML display when we were looking at the source design. Those come from using special JavaScript functions that control what the buttons will display and what they will do. Now, close the extension and then the breadcrumbs.html file and let's create our own file in the commands directory for creating your first extension.

Create a new file called myCommand.html under the commands folder in the site (myConfig) we just created. Load the file in Dreamweaver and go to the source view. You will see the standard HTML page layout at this point with the things you are used to seeing in a HTML page. Under the title of the document, change the name to Zippy Menu.

Add a `<script></script>` tag set in the `<head>` of the document. Now go back to the design view, click on the blank white canvas of the HTML page, and then insert a form (from the main menu bar, Insert -> Form -> Form and the familiar red dashed line will appear). Next, insert into the form a table that has the settings shown in Image IV.

In the left column, from top to bottom, enter the labels "Menu Label 1:", "Menu Label 2:", "Menu Label 3:", "Menu Label 4:" and "Menu Label 5:". In the right column put a corresponding text field in each of the table cells next to the labels. Select the entire table and click on the no wrap checkbox in the properties panel. Select the baseline vertical alignment for the entire table as well. Next, select the column of labels and right-align them. Finally, select the entire table and remove the width property (which is set to 100%). Your page should look like Image V when this is done.

The source code for this is shown in Code I.

Everything else we do will be between the `<script>` and `</script>` tags. Commands have certain special JavaScript functions that interact with Dreamweaver to accomplish needed events.

### canAcceptCommand()

This function controls whether the command will be displayed in gray when the command menu is accessed. Usually this is where you would do some sort of check to see if the command was being accessed properly. We will be using a default setting that will make the command available at all times. Normally, for a command that will insert code into the HTML page, we wouldn't want it to be available when the user doesn't have an HTML page open. Since we aren't doing that, our command would look like this:

```
function canAcceptCommand() {
    return true;
}
```

The word *function* tells Dreamweaver that this is a set of code it can access by the name – just like in JavaScript functions. In this case, the name is `canAcceptCommand()`. Since we aren't doing anything, we simply return true. If we were actually going to validate whether the command was supposed to be used, we could return false to indicate that the command name in the menu should be grayed out.

### commandButtons()

This function sets the button that will appear on the right side of the extension. At a minimum, we need two buttons, OK and Cancel. If we want to provide help or other abilities (for example, remove), they are added here. Our function will look like:

```
function commandButtons() {
    return new Array ("OK", "if (zmenu()) {win-
        dow.close();}",
        "Cancel", "window.close()");
}
```

Breaking this down, it returns an array, which contains sets of information. First a label ("OK"), then what JavaScript to execute when the button is pushed ("if (zmenu()) {window.close();}").

Now we get to the part of the extension where we make it do something. Notice in the `commandButton` function above we had a statement that included a function call named `zmenu()` if the OK button was clicked. This is what we will create next.

### zmenu()

The main function for the extension is the `zmenu()` function. Unlike the other two functions, this one is created by the developer and isn't part of the Dreamweaver extensibility code. In our case, our function will be used to put a Zippy Menu on the page. The Zippy Menu is basically a table with five columns and one row with each item in the extension user interface populating one of the table cells with a default link already set up. Let's set up the function first and then we'll take the parts of the function piece by piece.

```
function zmenu() {
}
```

The first thing we need to do is get the form input that the user entered into the form we provided. We do this like we would if we were trying to access the form elements on a HTML page. Macromedia used the document object model that was developed for the Internet Web pages for the extension processing. We have five textfields set to the Dreamweaver default names – `textfield`, `textfield2`, `textfield3`, `textfield4`, and `textfield5`. The form is named `form1` and everything else is standard; we will put the five labels into variables named `label1` through `label5`, and the code looks like this:

```
var label1=document.form1.textfield1.value
var label2=document.form1.textfield2.value
var label3=document.form1.textfield3.value
var label4=document.form1.textfield4.value
var label5=document.form1.textfield5.value
```

Normally you would error check to see if they have entered the expected information, and provide them with an alert about what they may have entered incorrectly or omitted, but for our demonstration we'll skip this part. If you plan to make an extension and have it for public use, error checking is critical to insure that users are able to utilize the extension to its maximum potential.

Now we need to compose the table with all of the variables we need in order to insert the end code into the page. This is done by storing the actual HTML code in a variable we'll call `str` (which is





is after the += to str. It is equal to str=str+new Stuff, so the code in Code II adds it all up into one big line. On the \ marks, what the backslash (\) does is tell JavaScript that the next character is part of the string instead of the end. When the string is written, it is removed. The last thing we need to do is write the str contents to the HTML page. Again we will rely upon the code Macromedia put in Dreamweaver to get the str contents onto the page. The code is as follows:

```
var theDOM = dw.getDocumentDOM();
theDOM.insertHTML(str);
```

The first line sets up theDOM to point to the HTML page's document object model (in other words, makes it so we can change, insert, or modify the page

view, click on the Commands menu from the main menu bar, and look for our command, called myCommand, and it will open Zippy Menu, similar to Image VI.

Enter five labels (one, two, three, four, and five) and click OK; you should get output like Image VII.

That is how a command is made. I hope I didn't lose anyone during the dash through the creation of Zippy Menu. You may be wondering why it said myCommand in the command menu instead of Zippy Menu. That's because the extension wasn't installed with a .mxd file via the extension manager. This requires packaging the extension and is another article all by itself. This should get you started and thinking of ways to turn common processes and things you



“rely upon the code macromedia put in dreamweaver”

short for String) and we will add the entered values in where they need to be. To start, let's get the basics of the table together.

```
var str="";
str="<table border='0'>";
str+="<tr>";
str+="<td>";
str+="<a href='\"+javascript:;\">";
```

Here is where we need to insert our first entry from the user.

```
str+=label1;
```

We need to close off the <a> tag and continue with the table; the remainder of the code would look like Code II.

Now we have the table all constructed in the str variable. The above comments use an operand, +=, which means take what is in str and add what

we want the extension to work on by accessing it through the variable theDOM). The next line uses the insertHTML() function to insert HTML into the page (I'll bet you didn't guess that...) The str in the insertHTML() function is what is being inserted. Finally, the last thing we need to do in creating the extension is return a value (look back at the commandButtons function; it is expecting this to return a true or false so it can close the window,) which is simply:

```
return true;
```

The entire code between and including the script tags is shown in Code III.

Now save the myCommand.html page, exit Dreamweaver, and then reload Dreamweaver (this resets the menus so the command will appear in them). Load a page and put it in design

# How Flexible is YOUR Shopping Cart?



© 2003, ComCity® Corp.

Our E-Commerce Solution can adapt to ANY Product and ANY Customer

Sure, you could build your own shopping cart from scratch, but should you...

ComCity® Corporation has been building shopping carts and hosting E-commerce solutions since 1995. Our experience extends from small shops to multi-server, multi-geographic Enterprise E-commerce.

• **ISP Friendly™** script-based checkout system without complicated dlls, available for Windows (ASP) as well as UNIX (PHP) hosting providers – host with 99.9% of all ISP's without special server installation support.

• **Dual product mode** allows designers to create a cart quickly in *static* mode or use *dynamic* mode for script-based flexibility.

• **Tight Dreamweaver MX integration** allows developers to use Database Server Behaviors to modify Catalog design with ease and flexibility.

A Shopping Cart *Foundation* for Dreamweaver E-commerce Developers

- Intuit® Quickbooks Integration
- Review orders with any browser
- Customer Control Panel
- Marketing Tools
- Sales Summaries
- Market/Auction based pricing
- Online self-paced training

Mention Ad Code: PUP0903 and receive one on-line SalesCart course FREE with your SalesCart purchase.

# SalesCart™

The first **Design-Tool E-commerce software** solution in the world™



For more information, visit [www.salescart.com](http://www.salescart.com) or call 1-888-826-6248 (Toll Free)

may do repeatedly into a command. Or it may have helped you appreciate the work that goes into the creation of even a simple extension.

### Conclusion

My site, [www.kaosweaver.com](http://www.kaosweaver.com), is a great resource for both commercial and free extensions. Another awesome resource is [www.dwfaq.com](http://www.dwfaq.com), where there is a resource page with an extensive list of extension developers. Of course, [www.macromedia.com](http://www.macromedia.com) has the Exchange, which is home to plenty of extensions. Get out there and extend Dreamweaver, expand your possibilities and "Weave Fast – Dream More"! ☺☺

*Paul Davis is an extension author, programmer, and owner of [Kaosweaver.com](http://www.kaosweaver.com). He lives in Overland Park, Kansas, with his wife and two children - with one more due in January 2004. "Weave Fast - Dream More" [kaosweaver@kaosweaver.com](mailto:kaosweaver@kaosweaver.com)*

code I

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Zippy Menu</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script>
</script>
</head>
<body>
<form name="form1" method="post" action="">
<table border="0">
<tr valign="baseline">
<td nowrap><div align="right">Menu Label 1: </div></td>
<td nowrap><input type="text" name="textfield"></td>
</tr>
<tr valign="baseline">
<td nowrap><div align="right">Menu Label 2: </div></td>
<td nowrap><input type="text" name="textfield2"></td>
</tr>
<tr valign="baseline">
<td nowrap><div align="right">Menu Label 3: </div></td>
<td nowrap><input type="text" name="textfield3"></td>
</tr>
<tr valign="baseline">
<td nowrap><div align="right">Menu Label 4: </div></td>
<td nowrap><input type="text" name="textfield4"></td>
</tr>
<tr valign="baseline">
<td nowrap><div align="right">Menu Label 5: </div></td>
<td nowrap><input type="text" name="textfield5"></td>
</tr>
</table>
</form>
</body>
</html>
```

code II

```
str+="

```

code III

```
<script>
function canAcceptCommand() {
    return true;
}

function commandButtons() {
    return new Array ("OK", "if (zmenu()) {window.close();}",
        "Cancel", "window.close()");
}

function zmenu() {
    var label1=document.form1.textfield.value
    var label2=document.form1.textfield2.value
    var label3=document.form1.textfield3.value
    var label4=document.form1.textfield4.value
    var label5=document.form1.textfield5.value

    var str="";
    str+="




```



NEED A  
**BETTER  
FITTING  
WEB HOST?**

## Try us on for size.

Web designers, application developers and marketing companies are switching to **HostMySite.com** for dedicated servers and shared hosting. We're a self-funded, debt-free, web hosting company founded in 1997 by technologists. On the first call, our support engineers can answer 99% of your questions, 24/7/365. Remember, all servers look pretty much the same. It's the service that makes us different.

Call us toll-free at 1-877-248-HOST (4678) to learn what we mean.

Shared hosting from **\$8.95!**  
Dedicated servers from **\$129!**  
**FREE** domain name!

Supporting ASP, .NET, ColdFusion MX, SQL Server 2000 and much more!

Visit [www.hostmysite.com/MXDJ](http://www.hostmysite.com/MXDJ) for this month's special offer.



Toppling the

# Tower of Babel

with MX

Have you tried developing for the Macromedia Flash Communication Server yet? If the answer is no, and you call yourself a Flash developer, you'd better head to [www.macromedia.com/downloads](http://www.macromedia.com/downloads) and download the free Developer Edition right away. Trust me when I say that you'll have the most fun you've had in a while. Developing client/server applications is hard enough with asynchronous operations and unreliable network conditions – throw in multiuser interactivity and you really have a challenge that will give your brain the workout it's been longing for.

by Giacomo 'Peldi' Guilizzoni





Flash Communication Server (or as people affectionately call it, "flashcom") provides you with a clean programming model, with communication objects (NetConnection, NetStream, the very powerful shared objects) spread across the Flash Player and the server side, where you code in our brand new flavor of JavaScript, called "ActionScript for Communication" (or ASC for short).

If you are not familiar with Flash Communication Server and its communication objects, head over to [www.macromedia.com/support/flashcom/documentation.html](http://www.macromedia.com/support/flashcom/documentation.html) and download the award-winning documentation there. I highly recommend "Developing Communication Applications." I always say, "Don't even consider coding a complex Flash Communication Server application unless you've read it front to back," and I mean it.

Flash Communication Server comes with a variety of prebuilt components that allow you to create a number of powerful and slick-looking applications with very little scripting. (I wrote a tuto-

You should have all of the above installed and ready to go in your development environment before continuing.

You should know where your Flash Communication Server's application folder is, the URL to your ColdFusion Flash Remoting gateway, and the location of the cfdocs folder.

### The Application: BabelChat

Chat rooms are great because they get people together in the same virtual space at the same time, regardless of geographical location. One last hurdle to communication that some users face is the language barrier. Wouldn't it be great if we could each simply type in our own native tongue, with the software taking care of the translation for us?

Well, only a few years ago this would have sounded like science fiction, but it's exactly what we're going to build in this tutorial.

Each user can choose to write, and read, in one of the following languages:

- Chinese
- English

Standard Communication Components such as the SimpleConnect, the UserColor, the ConnectionLight, and the PeopleList.

As you've probably already guessed, the translation is done by AltaVista's BabelFish software. If you haven't tried it yet, visit <http://world.altavista.com>. The same software also comes packaged as a Web service, which allows us to use it for our BabelChat application!

You can download the source code from [www.sys-con.com/mx/sourcecode.cfm](http://www.sys-con.com/mx/sourcecode.cfm). Are your fingers itching yet? Let's get cracking then!

### Architecture

Always start with a pen and paper. A whiteboard and felt pens are even better. After a few sketches, the application's architecture will look something like Image II.

Basically, the Flash clients (SWF movies) will talk to the Flash Communication Server (using the RTMP protocol). The Flash Communication Server will then talk to a ColdFusion Component (through Flash Remoting),

component will in turn invoke the BabelFish Web service to translate the sentence, catch the return value, and pass it back to the Flash Communication Server. Code I is the code for the CFC.

For the ColdFusion experts out there (I'm not one of them), this is pretty straightforward. We define a component with one function called translate. The function takes two arguments, an src string (the string we want to translate), and a code, which define the language to translate from and to. Valid values for the code string are:

- "en\_zh" (English to Chinese)
- "en\_fr" (English to French)
- "en\_de" (English to German)
- "en\_it" (English to Italian)
- "en\_ja" (English to Japanese)
- "en\_pt" (English to Portuguese)
- "en\_es" (English to Spanish)

Then:

- "zh\_en" for Chinese to English
- "fr\_en" for French to English

...you get the idea.

Save the file as main.asc in a directory called babeltest in your Flash Communication Server's applications folder and test it with the Communication App Inspector. Just connect to your server, type "babeltest/" in the App/Inst text field, click on "Load", then click on "View Detail". Click on "Reload App" and look at the traces. If after a few seconds you see "val: hello", you're golden. If not, the Flash Communication Server couldn't reach your ColdFusion Component. Check the connection string in the createGatewayConnection string and try again until you're ready.

Now we're ready to create the real application. Start an empty main.asc in a BabelChat application folder under the applications folder and type in:

```
load(*components.asc*);
load(*NetServices.asc*);
```

Nothing special there. We know we'll use some components, so we need to

## “Create a number of powerful and slick-looking applications with very little scripting”

al on how to create such an app here, see [www.macromedia.com/devnet/mx/flashcom/articles/first\\_comm\\_app.html](http://www.macromedia.com/devnet/mx/flashcom/articles/first_comm_app.html).)

As **MXDJ** readers know, components are great for demos and rapid prototyping, but are often not enough for your application's needs. Custom features are what make each Rich Internet Application unique and valuable.

In this tutorial we'll create an application that uses components for some of its features and custom client- and server-side code for the more advanced ones.

### Requirements

- This application uses the following:
- Flash MX with up-to-date Communication Components
  - Flash Communication Server 1.5, with up-to-date components framework
  - ColdFusion 6.1 with the Flash Remoting gateway

- French
- German
- Italian
- Japanese
- Portuguese
- Spanish

Users will even be allowed to write in one language and follow the chat conversation in another!

Image I shows a screenshot of the final application. Each user has a little avatar (the little guy with the user name next to it), which is used to display presence and also to choose which language to read in. Clicking on a country will move your avatar to it and switch the language displayed in the chat below it. Each user also has a combo box for selecting which language to write in, and the usual chat interface to send and read messages. The application also uses stan-

which will in turn invoke the Web service (through XML), which will deliver the translated information.

Once the architecture is in place, we need to think about the interaction between these pieces. Image III is a rough interaction diagram.

Fourteen steps to see one word translated in French...nobody said it was going to be easy!

Now that we have our rough architecture and interaction diagrams in place, we can start coding with a clear mind.

### The ColdFusion Component

Flash Communication Server needs to be able to talk to the Web service. It doesn't talk directly to Web services, so we have to go through remoting. A quick and easy way to do it is by creating a "shimmy" ColdFusion Component that the Flash Communication Server can invoke. The

The only thing that the translate function does is invoke the Web service by passing it the two variables we passed to babelfish.cfc; wait for the result (which gets stashed in the aString variable) and pass the result back to the caller as a return value.

Save the code in a file called babelfish.cfc in your ColdFusion cfdocs directory and you're done with it. This was the easy part.

### Server-Side Code

I always find it easier to start with the server-side ActionScript for Communication code. There are fewer things to think about there; it's all just code. No MovieClips, time lines, colors, or graphics to distract me.

The first thing we need to do is make sure the remoting calls work. A simple test of Code II assures that they do.

load components.asc, and we need to load NetServices.asc to enable our app to do Flash Communication Server-to-ColdFusion remoting.

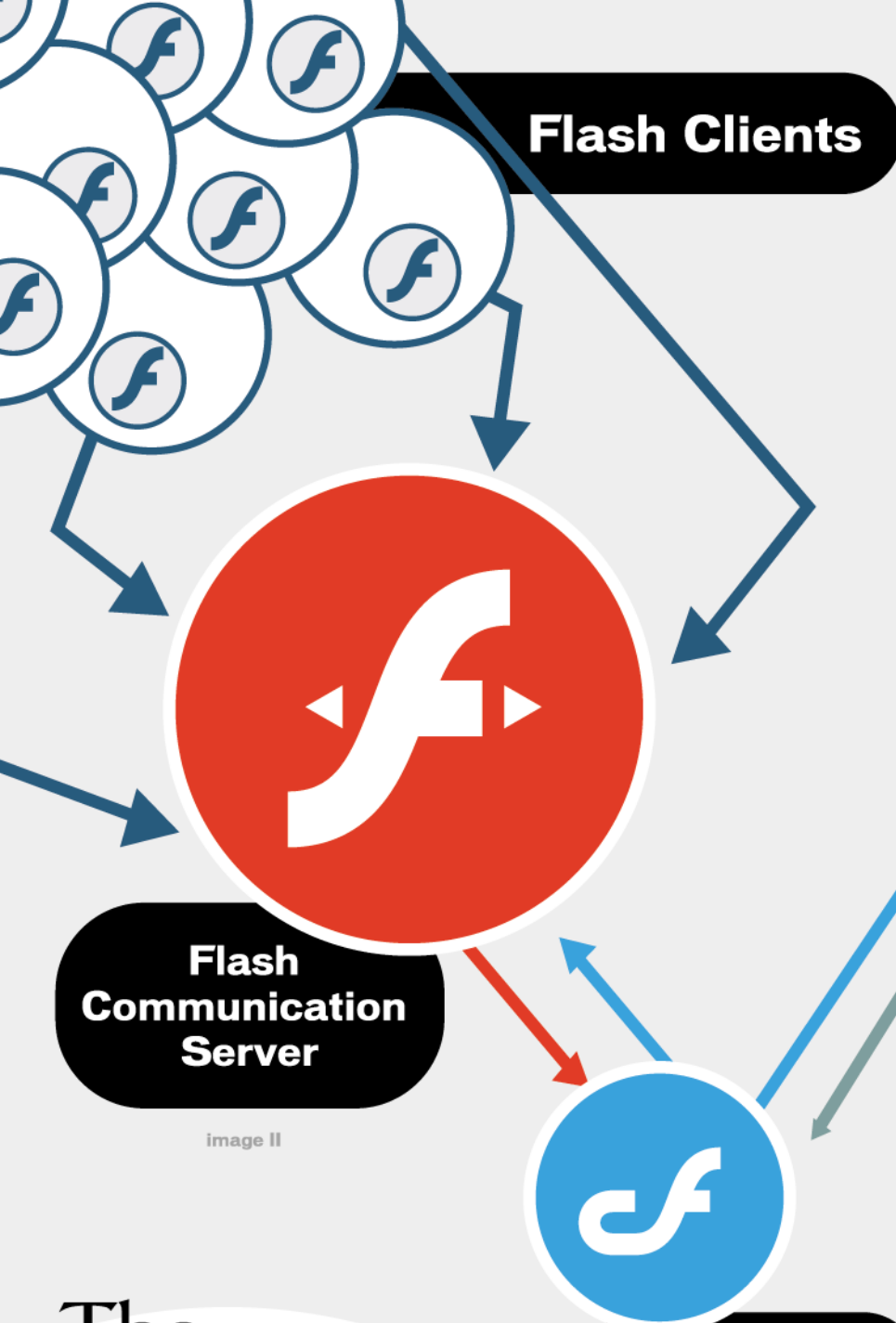
Next we define the BabelFishResultHandler object (see Code III).

This object is in charge of receiving a result from babelfish.cfc (a translated string), and either returning it to its "owner" or calling babelfish.cfc again to translate the string again (this happens every time we "go through English").

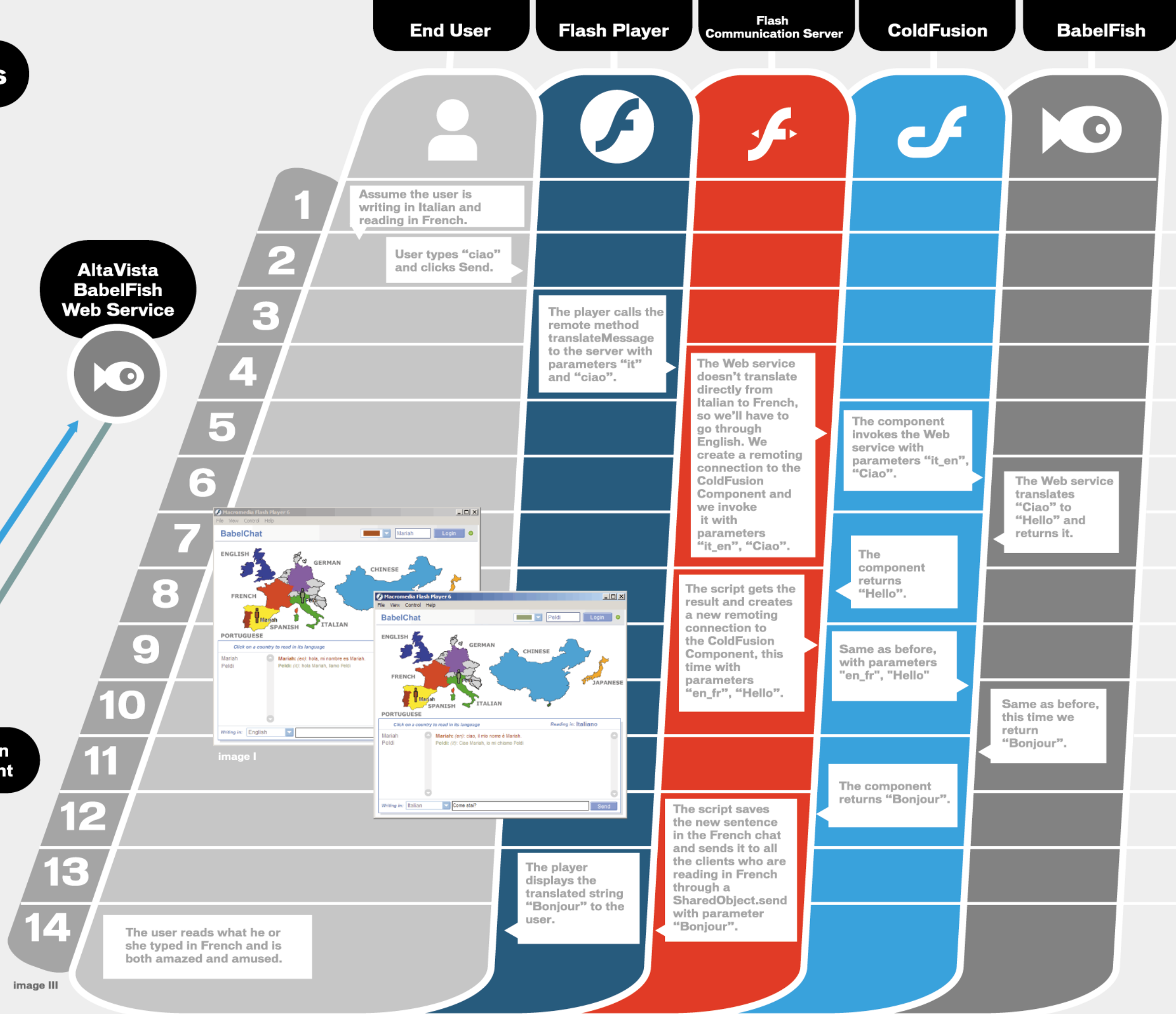
Next we define the BabelFishTranslator object, which will create instances of BabelFishResultHandler (see Code IV).

Our application will have an instance of the BabelFishTranslator for each combination of languages we want to translate from and to. In its constructor it decides whether it should use 0 (same language), 1 (to and from English), or 2





# The Tower of Babel



# “Time to wake up the designer in you and launch Flash MX

passes (“through” English) to translate from the “from” language to the “to” language.

The BabelFishTranslator object has only one method, called translate. Its job is to translate a string invoking the babelfish.cfc component using an instance of the BabelFishResultHandler object.

When the BabelFishResultHandler instance has finished translating, it will invoke the onResult method for this BabelFishTranslator, which by default does nothing. We’ll see later that our script will overwrite this method to deliver the translate sentence to the Flash clients.

Next we define some startup code for our Flash Communication Server application (see Code V).

Code V is worth explaining carefully. First, we create an array of language codes (application.language

Codes). We initialize a few variables:

- translatorObjects is going to be a table of instances of BabelFishTranslator.
- sentenceCounter is a unique counter for each sentence that gets sent by the Flash clients. We need this counter to maintain the order of each sentence, regardless of how long it took to translate it.
- languageSharedObjects is also a table, this time of shared objects. We’ll keep a shared object for each language, and Flash clients will subscribe to only one of them at a time, depending on which language they want to read the chat in.

Then we actually populate the languageSharedObjects table by looping through the language codes. For each language we get a nonpersistent shared object (called “lang\_en”, “lang\_zh”, etc.) on which we define a few variables (a lastSent counter, a history string, and a queue Array), and a method called deliverInOrder.

This method will be called by each

BabelFishTranslator object when a new sentence has been translated and is ready for delivery. It is in charge of maintaining the order of delivery of messages, and it does this by using the lastSent property in combination with the queue array. When a message needs to be delivered to a client, it is done through a SharedObject.send call – the this.send(“newMsg”, msg) calls. We’ll see later how to make the Flash clients respond to such a call.

The last part of our startup is to get a littleGuys SharedObject, which we’ll use to maintain a shared view of the users’ position, name, and color across users. Basically, whenever a user clicks on a country, their little avatar will move to that country, and this movement will be reflected in all the clients connected to the app.

To implement this feature we need to define the two methods shown in Code VI.

onConnectAccept is called when every Communication Component has decided to accept a client’s connection. In it, we create a new slot in the littleGuys\_so SharedObject, using the client.\_\_ID\_\_ property, which was automatically assigned by the component’s framework during this client’s onConnect. We prefix the ID with a “u” for “user” because it’s always better to start array indexes with a letter. We assign a default position to this client’s avatar. This is just some placeholder data; we’ll see in the section on client-side code that each client overwrites this data with data retrieved from a local shared object. A SharedObject.setProperty on the server side will trigger an onSync on the client side. That’s how the client script will decide to create a new little guy for the user who’s connecting.

In the onDisconnect function we delete the slot for the client disconnecting. This will trigger an onSync on the client side, which will remove the appropriate little guy.

One last thing that we do in onDisconnect is clear the history in every language if this user is the last one to leave the room.

Back to the translation feature: we need to provide a function for our Flash clients to call whenever they want to send a new message. To do so we attach the translateAndSendMessage method in Code VII to the Client object prototype.

The function takes as input the user

name of whoever sent the message, the language code it was written in, and the message itself.

Server-side ActionScript is single-threaded, so it’s great to assign or increment global counters. That’s just what we do with sentenceCounter: even if multiple clients click “Send” at the same exact time, thus invoking translateAndSendMessage, Flash Communication Server guarantees us that these calls will be put into a queue and executed only one at a time.

Then we translate the sentence in all the available languages by looping through the language codes. If it’s the first time we try to translate from this language (lang) to another (application.languageCodes[i]), we create a new BabelFishTranslator to take care of the translation. We also overwrite the onResult method for this translator. In it, we “prettify” the message by adding the user name of the sender and the code of the language of origin of the message, and changing the color of the HTML string. Then we invoke the deliverInOrder method on the appropriate languageSharedObject instance, to deliv-

er the translated message to the clients.

After instantiating the translator object (only if it was necessary), we finally invoke the translate method on it. We repeat this for each language.

Do you see the round-trip? Quickly: client calls translateAndSendMessage, which calls translate on eight different BabelFishTranslator instances, each of which does the remoting call to babelfish.cfc, which returns through BabelFishResultHandler, which invokes onResult on the translator, which invokes deliverInOrder on its destination SharedObject, which calls send(“newMsg”) back to the client. Woah. Good stuff.

The hard part’s over; it’s all downhill from here.

On the server-side, we need to finish with two more methods. One is getHistory (see Code VIII).

This method is called by a client when it switches reading languages. All it does is return the history string that has been building up in the deliverInOrder method seen in Code V.

The last one is a utility method taken straight from the standard Chat component

that comes with Flash Communication Server (chat.asc). It’s a function called hiliteURLs, which finds URLs in a string and turns them into links. It’s not shown here for brevity, but you can find it in the tutorial source code that you can download.

Congratulations on completing the hardest part of the tutorial. Time to wake up the designer in you and launch Flash MX, we’re going to the client side!

## The Client-Side UI

First of all we need to place a few components on the stage:

- Grab a UserColor component and name it color\_mc
- Grab a ConnectionLight component and name it light\_mc
- Grab a PeopleList component and name it people\_mc
- Grab a SimpleConnect component and name it simpleConnect\_mc

In the Properties Inspector for SimpleConnect, type rtmp:/babelchat/ (or rtmp://yourdomain/babelchat/ if your Web server and Flash Communication Servers are on different machines) for Application Directory and add color\_mc,

**Macromedia Authorized Training**

**Future Media Concepts**  
Training a New Generation of Digital Artists™

**FMC OFFERS THE COMPLETE MX CURRICULUM:**

- Coldfusion
- Flash
- Dreamweaver
- Director
- Fireworks
- Freehand

**macromedia MX**

macromedia AUTHORIZED TRAINING PARTNER

Nationwide On-site Training  
877.362.8724

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

New York • Boston • Philadelphia • Washington DC • Miami • Orlando



- Create an input text field and call it msg\_txt.
- Drag an FCPushButton component from the library, call it send\_pb. Using the property inspector give it the label Send, and Click Handler doSend.
- Drag an FCComboBox from the library and name it writingIn\_cb. In the labels array, add Chinese, English, French, German, Italian, Japanese, Portuguese, and Spanish. In the Data array, add zh, en, fr, de, it, ja, pt, and es. In the Change Handler parameter, type updateWritingInIndex.
- Add a title bar and whatever else you need to make your application look just the way you like it.

The last thing you need to create is a MovieClip for the little guys. Insert menu,

event notification to other scripts that the SimpleConnect has set up all communication components and is ready to go. This is a good place for us to connect our non-componentized parts of the application.

First we save a reference to the main NetConnection (saved in SimpleConnect under main\_nc) in a \_global variable, for easier access.

Then we get a local shared object (think of it as a cookie) from the user's hard drive. If one of its properties is null it means that the cookie was not found, so we assign some default values to it. Then we adjust the UI to reflect the saved values and we call updateReading, passing the saved reading language as a parameter. We'll see what updateReading does shortly.

Next, we define a simple Key listener to trap ENTER keys, so that users will be able to send a message just by pressing enter at the end of it. We'll see what the

“...add Chinese, English, French, German, Italian, Japanese, Portuguese, and Spanish”

people\_mc, light\_mc to the Communication Components array.

Then you need a map. A quick Google search pointed me to the excellent Zoom Map 1.2 by WISECAP on FlashKit ([www.flashkit.com/movies/Interfaces/Navigating/Zoom\\_Map-WISECAP-5423/index.php](http://www.flashkit.com/movies/Interfaces/Navigating/Zoom_Map-WISECAP-5423/index.php)). I downloaded the FLA, took out the countries I was interested in, colored them, and placed them on my stage along with the Communication Components. I turned each country's color filling into a MovieClip, so that they could be clicked on. I called them map\_fr\_mc, map\_en\_mc, map\_it\_mc, and so on.

We need a couple more things:

- Create a dynamic text field and call it readingIn\_txt. This will hold the name of the language the user is reading in.
- Create another dynamic text field, this time multiline and HTML enabled, and name it history\_txt. This will be the chat history.
- Drag an FCScrollBar from the library (Communication Components/Com-

New Symbol..., call it "LittleGuy"; click on "Export for Actionscript", and type "LittleGuySymbol" in the Identifier. Add two dynamic text fields, name\_txt (black) and nameShadow\_txt (white, behind it and offset a little). Draw a little person, select the part you want to change colors, and turn it into a MovieClip. Call this MovieClip's instance bg\_mc.

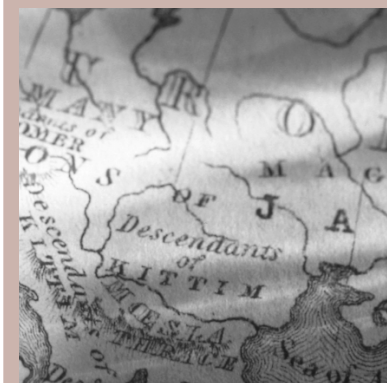
Okay, we're ready to write the last bit of code.

#### Client-Side Code

I like to keep my ActionScript files external, but you can put this code in the first frame of your movie within Flash MX if you like. First of all, there's some setup code (see Code IX).

gLanguageStrings is a table that gives us a string for each language code. We'll use this later.

Then we overwrite the onComponentsConnected method of our simpleConnect\_mc component. This was introduced in 1.5 as a simple kind of



# AMERICAN MEDIA

<http://www.americanmediatraining.com>

800 278 7876

# TRAINING

Macromedia Authorized Partner since 1995

Experienced instruction from beginner to expert level

Clients include major studios, government agencies, fortune 500

The first step to certification

Always competitive pricing

Free technical support

Custom courses - tailored to suit

On-Site

Los Angeles

Denver

Las Vegas

Worldwide

Flash

ColdFusion

Dreamweaver

Fireworks

Contribute

Authorware

Director

Coursebuilder



# “Flash designers with no programming background are asked to understand client/server programming”

doSend function does shortly.

Then we need to write some code to make the avatars work. We get a remote shared object. (Do you remember it? It's the same we got on the server side). We define my favorite onSync on it ([www.peldi.com/blog/000008.html](http://www.peldi.com/blog/000008.html) for more information), and we connect it to the global NetConnection.

Now for the part I'm not proud of. We need to be notified by SimpleConnect if the user changes his or her user name, but the SimpleConnect component doesn't throw such an event. One of the beauties of free components is that you can dig into their code and tweak it to your needs, which is just what I had to do in this case. Open the SimpleConnect component in the library, go to line 204 (at the end of the changedName function), and add the following line:

```
this.onNameChanged(newName);
```

This will create a little event invocation that we can trap by defining an onNameChanged handler on our instance of SimpleConnect, which is exactly what we do in Code IX.

Another change we want to make to SimpleConnect is in line 174, in the onResult handler for the connect call. Add the following line:

```
_global.userID = val;
```

This way we can use the global ID assigned to us by the server-side framework on the client side. This is what we do in the body of onNameChanged: we save the new user name in our slot. This will trigger an onSync on all the clients and update the UI accordingly.

The userColor component offers a more complete event architecture, so adding the gFlashCom.userprefs.addListener(\_root); line sets up \_root as a listener to it, which means that

whenever a new color is chosen from the list, the function onColorChange() will be invoked. We will define that function a little bit later.

Next we define the doSend function (see Code X).

Very simple. All we do is a client-to-server call to the translateAndSend Message function described in the server-side code section, passing in the user name, the selected language code, and the text of the message. Then we clear the msg\_txt textfield, and we're ready for a new message.

So how do we get the translated message back from the server? Code XI shows a function that will do the trick.

We call this function every time we change the \_global.readingIn variable. What this function does is connect to the correct shared object for the chosen language and define a newMsg method on it (that's the one the server uses to send us new translated messages as they arrive!).

The function also makes a client-to-server call to getHistory to retrieve the history in the selected language – not much more to do now (see Code XII).

First we define onRelease functions for each country's movieclip. All they do is call updateReading with the appropriate language code and updatePosition, defined in Code XII.

The updateReading function sets the global readingIn variable to the selected language, saves it in the local shared object, updates the UI and calls connectSharedObject.

The updatePosition function sets the new position of this client's little guy to the mouse's position in the remote Shared Object. This will trigger an onSync that will update the UI for all the users. We also save the new position in the local shared object, for the next time we log on.

The updateWritingInIndex is the change handler for writingIn\_cb, and just saves the new selected writing language in the local shared object.

Code XIII is the last bit of code, I promise. First we define the onColorChange function, which is called whenever a user changes his or her color in the userColor component. All we do is update the usercolor variable in this client's slot of the littleGuys shared

object. This will trigger an onSync...you know the drill.

The onSlotChanged is called by the onSync whenever a new slot is created or an existing slot is changed by anyone. In it, first we check if this slot is new to us (i.e., we didn't attach a little guy for this user). If it is, we attach a LittleGuySymbol MovieClip for this user. If the user is ourself (id == \_global.userID), we populate our slot of the shared object with the data we know from the components on the stage or the local shared object. These changes will trigger another onSync, which will call onSlotChanged on our ID again. The second time, the slot won't be new to onSlotChanged anymore, so we'll skip this part (and avoid an infinite loop, which is always good).

After adding the new little guy (if necessary), we update the UI with data from the remote shared object: we update its position, the text of the name, and the color of the little guy.

The onSlotDeleted function is called whenever a user leaves (remember how we set the slot to null on the server side in the onDisconnect function? It all

makes sense now). All we do in it is remove the MovieClip for that user.

That's it! Save your FLA, publish it, open the SWF in two browser windows, and you're ready to start making international friends!

## Conclusion

There's no doubt about it: doing this stuff is hard. There's a lot to think about, lots of pieces that need to fit together. Flash designers with no programming background are asked to understand client-server programming (among the hardest kinds of programming IMHO), or hard-core back-end XML coders need to "lower themselves" to a tool originally created for animators. Such is the world we live in, my friends. I say embrace the spectrum. Look at how much fun you can have with a team of one! Judging by the fact that Studio MX is the best-selling product in Macromedia history, more and more people are doing the same.

You are now able, in a few hours, to build something that might have taken months and months of development only a few years ago. This is the power of

MX. Write your code in Dreamweaver; ColdFusion does tons of work for you (look at how easy it was to build that powerful CFC); Flash Communication Server handles most of the client/server interaction for you (have you noticed how powerful shared objects are?); and Flash lets you create assets in no time (also with the help of prebuilt components). Finally, and perhaps most important, you can draw from an enormous pool of resources because the Macromedia community of developers is so huge and willing to share (look at my maps, for example).

Good luck! Bonne chance! Viel glueck! Buona fortuna! Boa sorte! and ¡Buena suerte! ☺☺

Giacomo "Peldi" Guilizzoni (<http://peldi.com>) is a software developer for Macromedia working on Breeze Live. He maintains a Flash Communication Server blog at <http://peldi.com/blog/> and often writes for Macromedia's DevNet. His grandparents do not understand a single word of this article, but nonetheless think it's "neat." [gguilizzoni@macromedia.com](mailto:gguilizzoni@macromedia.com)

## Kaosweaver Weave Fast, Dream More

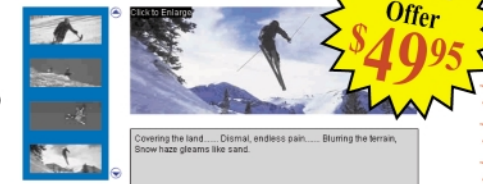
### Your Solution for

- 40+ **FREE** Extensions
- Custom Built Extensions
- ASP and PHP Programming
- Script Debugging
- Form Calculation and Validation
- Custom Shopping Carts
- Database Design
- SQL Coding



### inTensa Design Gallery Extension

- Quick loading photo gallery
- DHTML scroller for thumbnails
- Full HTML captions for each image
- Automatic settings for fast gallery setup
- Full size image pop-up window
- Configurable and editable extension



Only **\$49.95** Reg. \$59.00 <http://www.kaosweaver.com/mxoffer.asp>

### Calendar Pop-Up Extension

- International language support
- Configurable color selection
- Start weekday on Sunday or Monday
- Control selectable days for past only, future only
- American and European date formats supported
- Works cross browsers



Only **\$19.00** [http://www.kaosweaver.com/Extensions/Calendar\\_Popup](http://www.kaosweaver.com/Extensions/Calendar_Popup)



code I

```
<cfcomponent>
<cffunction name="translate" access="remote" returnType="any">
<cfargument name="src" type="string" default="">
<cfargument name="code" type="string" default="">

<CFINVOKE
WEBSERVICE="http://www.xmethods.net/sd/2001/BabelFishService.wsdl"
METHOD="babelFish" RETURNVARIABLE="aString">
<CFINVOKEARGUMENT NAME="translationmode" VALUE="#code#" />
<CFINVOKEARGUMENT NAME="sourcedata" VALUE="#src#" />
</CFINVOKE>

<cfreturn aString>
</cffunction>
</cfcomponent>
```

code II

```
load("NetServices.asc");

this.gatewayConnection =
NetServices.createGatewayConnection("http://localhost/flashser-
vices/gateway");
var babelService =
this.gatewayConnection.getService("cfdocs.babelfish", this);
babelService.translate( {code:"it_en", src:"Ciao" } );

function translate_Result(val) {
trace('val:'+val);
}
```

code III

```
// BabelFishResultHandler Object
//+
BabelFishResultHandler = function(owner, secondPass,
sentenceCounter)
{
this.owner = owner;
this.secondPass = secondPass; //might be undefined
this.sentenceCounter = sentenceCounter;
}
BabelFishResultHandler.prototype.translate_Result = function(val)
{
if (this.secondPass == undefined) {
this.owner.onResult(this.owner.username, this.owner.hexColor, val,
this.sentenceCounter);
} else {
var gatewayConnection =
NetServices.createGatewayConnection("http://localhost/flashser-
vices/gateway"
);
var babelService = gatewayConnection.getService("cfdocs.babelfish",
new
BabelFishResultHandler(this.owner, null, this.sentenceCounter));
babelService.translate( {code:this.secondPass, src:val } );
}
}
BabelFishResultHandler.prototype.translate_Status = function(info)
{
trace(info.code);
}
//-
```

code IV

```
// BabelFishTranslator Object
//+
BabelFishTranslator = function(from, to)
{
this.from = from;
this.to = to;

if (this.from == "en") {
if (this.to != "en")
this.firstPass = "en_"+this.to;
} else {
if (this.to != this.from) {
this.firstPass = this.from+"_en";
if (this.to != "en")
this.secondPass = "en_"+this.to;
}
}
this.gatewayConnection =
NetServices.createGatewayConnection("http://localhost/flashser-
vices/gateway"
);
}
BabelFishTranslator.prototype.onResult = function(val)
{
//stub
}
BabelFishTranslator.prototype.translate = function(username,
hexColor,
theString, sentenceCounter)
{
this.sentenceCounter = sentenceCounter;
this.username = username;
```

```
this.hexColor = hexColor;
if (this.firstPass == undefined) {
this.onResult(this.username, this.hexColor, theString,
this.sentenceCounter);
} else {
var babelService =
this.gatewayConnection.getService("cfdocs.babelfish", new
BabelFishResultHandler(this, this.secondPass,
this.sentenceCounter));
babelService.translate( {code:this.firstPass, src:theString } );
}
}
//-
```

code V

```
application.onAppStart = function()
{
application.languageCodes = ["en", "zh", "fr", "de", "it", "ja",
"pt", "es"];

application.translatorObjects = new Object();
application.sentenceCounter = 0;
application.languageSharedObjects = new Object();

for (var i in application.languageCodes) {
if (typeof(application.languageCodes[i]) == "function")
continue;

application.languageSharedObjects[application.languageCodes[i]] =
SharedObject.get("lang_"+application.languageCodes[i], false);
application.languageSharedObjects[application.languageCodes[i]].lang
= application.languageCodes[i];
application.languageSharedObjects[application.languageCodes[i]].last
Sent = 0;
application.languageSharedObjects[application.languageCodes[i]].his-
tory = "";
application.languageSharedObjects[application.languageCodes[i]].queu
e = new Array();
application.languageSharedObjects[application.languageCodes[i]].deli
verInOrder = function(msg, counter)
{
if (counter == (this.lastSent+1)) {
this.history += msg;
this.send("newMsg", msg);
this.lastSent++;
var i = (this.lastSent+1);
while (this.queue[i] != undefined) {
this.history += this.queue[i];
this.send("newMsg", this.queue[i]);
delete this.queue[i];
this.lastSent = i;
i++;
}
} else {
//add To queue
this.queue[counter] = msg;
}
} //deliverInOrder
} //for

application.littleGuys_so = SharedObject.get("littleGuys", false);
}
```

code VI

```
application.onConnectAccept = function(client) {
application.littleGuys_so.setProperty("u"+client.__ID__, {x:100,
y:100});
}

application.onDisconnect = function(client) {
application.littleGuys_so.setProperty("u"+client.__ID__, null);
}

if (application.clients.length == 0) {
for (var i in application.languageSharedObjects) {
if (typeof(application.languageCodes[i]) == "function")
continue;
application.languageSharedObjects[i].history = "";
}
}
}
```

code VII

```
Client.prototype.translateAndSendMessage = function(username, lang,
msg)
{
application.sentenceCounter++;
for (var i in application.languageCodes) {
if (typeof(application.languageCodes[i]) == "function")
continue;

if
(application.translatorObjects[lang+"_"+application.languageCodes[i]]
==
undefined) {
application.translatorObjects[lang+"_"+application.languageCodes[i]]
= new
```

code VIII

```
BabelFishTranslator(lang, application.languageCodes[i]);
application.translatorObjects[lang+"_"+application.languageCodes[i]]
.onResul
t = function(username, hexColor, val, counter) {

var msg = hiliteURLs(val);

msg = "<font color='\" + hexColor + \"'><b>\" + username + \": </b>
<i>"+this.from+"</i>: " + msg + "</font><br>\n";

application.languageSharedObjects[this.to].deliverInOrder(msg,
counter);
} //onResult
} //if

var cglobal = gFrameworkFC.getClientGlobals(this);
if (cglobal == undefined) {
cglobal = new Object();
cglobal.usercolor = "0x000000";
}
var hexColor = "#"+cglobal.usercolor.substring(2, cglobal.usercol-
or.length);

application.translatorObjects[lang+"_"+application.languageCodes[i]]
.transla
te(username, hexColor, msg, application.sentenceCounter);
} //for

//Called by the clients
Client.prototype.getHistory = function(lang) {
return application.languageSharedObjects[lang].history;
}
```

code IX

```
gLanguageStrings = {en: "English", de: "Deutsch", it: "Italiano", zh:
"Chinese", ja: "Japanese", pt: "Portuguese", es: "Español", fr:
"French" };

simpleConnect_mc.onComponentsConnected = function() {
_global.nc = this.main_nc;

local_so = SharedObject.getLocal("BabelChat", "/");
if (local_so.data.readingIn == null) {
local_so.data.readingIn = "en";
local_so.data.writingInIndex = 1;
local_so.data.x = 110;
local_so.data.y = 110;
}
writingIn_cb.setSelectedIndex(local_so.data.writingInIndex);
updateReading(local_so.data.readingIn);

this.typeListener = new Object();
this.typeListener.onKeyDown = function() {
if ((Selection.getFocus()+")" == (msg_txt+"")) && (Key.getCode() ==
Key.ENTER))
doSend();
}
Key.addListener(this.typeListener); //Little Guys stuff

//+
littleGuys_so = SharedObject.getRemote("littleGuys", _global.nc.uri,
false);
littleGuys_so.firstTime = true;
littleGuys_so.onSync = function(list) {
if (this.firstTime) {
this.firstTime = false;
for (var i in this.data)
onSlotChanged(i);
for (var k in list) {
if ((list[k].code == "change") || (list[k].code == "success"))
onSlotChanged(list[k].name);
else if (list[k].code == "delete")
onSlotDeleted(list[k].name);
}
}
}
littleGuys_so.connect(_global.nc);

simpleConnect_mc.onNameChanged = function(val) {
littleGuys_so.data[_global.userID].username = simpleConnect_mc.user-
name;
}

gFlashCom.userprefs.addListener(_root);
//-
```

code X

```
//Send button handler
function doSend() {
_global.nc.call("translateAndSendMessage", null,
simpleConnect_mc.username, writingIn_cb.getSelectedIndex().data,
msg_txt.text);
msg_txt.text = "";
}

function connectSharedObject() {
history_txt.htmlText = "";
lang_so = SharedObject.getRemote("lang_"+_global.readingIn, _glob-
al.nc.uri,
false);
lang_so.newMsg = function(msg) {
history_txt.htmlText += msg;
history_txt.scroll = history_txt.maxScroll;
}
lang_so.connect(_global.nc);

var res = new Object();
res.onResult = function(val) {
history_txt.htmlText = val;
history_txt.scroll = history_txt.maxScroll;
}
_global.nc.call("getHistory", res, _global.readingIn);
}

//Map onRelease handlers
map_fr_mc.onRelease = function() { updateReading("fr");
updatePosition(); }
map_es_mc.onRelease = function() { updateReading("es");
updatePosition(); }
map_en_mc.onRelease = function() { updateReading("en");
updatePosition(); }
map_pt_mc.onRelease = function() { updateReading("pt");
updatePosition(); }
map_de_mc.onRelease = function() { updateReading("de");
updatePosition(); }
map_it_mc.onRelease = function() { updateReading("it");
updatePosition(); }
map_zh_mc.onRelease = function() { updateReading("zh");
updatePosition(); }
map_ja_mc.onRelease = function() { updateReading("ja");
updatePosition(); }

function updateReading(lang) {
_global.readingIn = lang;
local_so.data.readingIn = _global.readingIn;
readingIn_txt.text = gLanguageStrings[_global.readingIn];
connectSharedObject();
}

function updatePosition() {
littleGuys_so.data[_global.userID].x = _root._xmouse;
littleGuys_so.data[_global.userID].y = _root._ymouse;
local_so.data.x = _root._xmouse;
local_so.data.y = _root._ymouse;
}

//Change handler for writingIn_cb
function updateWritingInIndex() {
local_so.data.writingInIndex = writingIn_cb.getSelectedIndex();
}

//Little Guys Stuff
//+
function onColorChange(val) {
littleGuys_so.data[_global.userID].usercolor =
gFlashCom.userprefs.color;
}

function onSlotChanged(id) {
if (_root["littleGuy_"+id] == null) {
_root.attachMovie("LittleGuySymbol", "littleGuy_"+id, id.sub-
str(1)/1+100);
_root["littleGuy_"+id].nameShadow_txt.autoSize = true;
_root["littleGuy_"+id].name_txt.autoSize = true;
if (id == _global.userID) {
littleGuys_so.data[id].username = simpleConnect_mc.username;
littleGuys_so.data[id].usercolor =
color_mc.ColorCombo.getSelectedItem().data;
littleGuys_so.data[id].x = local_so.data.x;
littleGuys_so.data[id].y = local_so.data.y;
}
}
_root["littleGuy_"+id]._x = littleGuys_so.data[id].x;
_root["littleGuy_"+id]._y = littleGuys_so.data[id].y;
_root["littleGuy_"+id].nameShadow_txt.text =
littleGuys_so.data[id].username;
_root["littleGuy_"+id].name_txt.text = littleGuys_so.data[id].user-
name;
var c = new Color(_root["littleGuy_"+id].bg_mc);
c.setRGB(littleGuys_so.data[id].usercolor);
}

function onSlotDeleted(id) {
_root["littleGuy_"+id].removeMovieClip();
}
//+
```

# Understanding Levels

Your first step to photo correction  
by Kleanthis Economou

The Levels dialog is commonly one of those dialogs that users open to see what it is, look at it for a few seconds, don't understand a thing, and then close it right away. The more ambitious users might actually try to figure it out by clicking on different things left and right but they don't understand why things happen the way they happen, so they close it.

In this article, I explain what Levels are and how to use them properly. Levels are not specific to Fireworks. Pretty much any program that offers image manipulation includes a Levels command that does the same thing, and even looks very similar. For example, the Levels command in Photoshop is the same as it is in Fireworks (although the one in Photoshop has more features).

I don't fully agree with the classification of Levels as being a filter but since Fireworks lists it as such in its menu system, I'll call it a filter here too.

Very rarely, if ever, will you need to apply Levels to a vector object. Levels are something you adjust on a photo most of the time. However, since you can use it on vectors as well, I call the target item an object.

## Applying Levels

The Levels filter in Fireworks can be applied to bitmap and/or vector objects depending on where you apply it from. If you apply it from Filters-> Adjust Color-> Levels to a bitmap, it can be used only on bitmaps. If you actually have a vector object selected, Fireworks will ask you if it's okay to convert it. If you add Levels to

an object as a live effect from the Property Inspector, this object can be either a vector or a bitmap.

Unless you have a very specific reason not to do so, it is preferable to apply Levels as a live effect. The main advantage is that you can remove it and/or alter it at any time.

A typical sequence of applying Levels would be:

1. Select an object in your document with the Pointer tool.
2. In the Property Inspector click on the plus (+) sign on the Effects area and select: Adjust Color-> Levels...

## Histogram

If I told you to take each pixel of the selected object and record its brightness, it would take you some time, but you could do it. The brightness of each pixel has a value of 0-255, with 0 being black and 255 being white.

Now, if I told you to take this list of values and make a graph, the first thing you would need to know would be what each axis of the graph represents. So let's agree that the horizontal axis represents brightness and the vertical the amount of pixels. If you plot each pixel's brightness on such a diagram, you would have what is known as a histogram. In a way, you can say this histogram is the blueprint of the brightness (of the selected object).

This is what Levels does, only a bit faster than you and I can. It examines every pixel of your selected object and displays its histogram. And it can do so for each channel individually (Red, Green, Blue) and of course for the composite RGB.

Seeing this histogram (see Image 1) gives you a pretty good idea of the overall brightness and contrast of a photo. For example, a big bump on the left side means that you have a lot of dark pixels. This may be all right, depending on the subject, but it may not (for example, the shot may be too dark). Some high-end

consumer/pro digital cameras give you the histogram of a photo you take on the fly.

All this information is valuable just to look at and most of the time the first thing you do when you're doing photo retouching is to check the Levels. But all this data becomes more valuable if you can manipulate it and take advantage of it. I will show you how you can do just that.

## Input/Output Levels

These two work side by side. What you do here is take a brightness range and map it to a specific brightness. I'll discuss this later.

The Input Levels have three values that from left to right in Image 1 are black, gamma, and white. Some users prefer to think of gamma as exposure, or midtones. In either case, it maps the mid-gray. The tooltip of those fields in Fireworks calls them minimum intensity, gamma, and maximum intensity respectively. But that's just a fancy way of saying black, mid-gray, and white.

By default, their values are 0, 1.00, and 255. Interesting things happen here if we start changing them. I can say that Black's brightness isn't 0, it's 25. That means that every pixel that has a brightness between 0 and 25 will become black because I say so - by setting the input level of black to 25. Or, I can say that white's brightness isn't 255, but 200. So, every pixel that has a brightness between 200 and 255 will turn pure white.

You can set these values directly in the text fields, or you can use the sliders at the baseline of the histogram. As you can see, there are black, white, and mid-gray sliders. Those sliders and the text fields of the Input Levels are connected. It is obviously easier to use the slider because by doing so you can see how you group the brightness and how many pixels will be affected. For example, when you move the black slider to the right all the pixels to the left will turn black.

But do they turn really black? That depends on the output levels. Using output levels, you essentially define what is black and what is white for you. You define the brightness of "black" and the brightness of "white." We have two fields that hold the new value of black and the new value of white. By default, those values are as we know them, 0 and 255 respectively. But nothing can stop you from changing them to something else. Again, there are two sliders that can make your life easier, allowing you to choose from the black-to-white gradient.

The gamma setting represents the brightness level of the medium-gray value in the image. By making changes to this, you can change the amount of contrast in the image by lightening and darkening grays (also known as midtones) without affecting the shadows and highlights. By default when you alter the minimum or maximum intensity, the gamma value changes automatically to be in the middle. You can, however, change this by entering a gamma value manually, or by using the gray slider.

As an example of using input and output levels, let's say that I change the minimum intensity (black) of the Input Levels to 50, and the minimum intensity (black) of the output levels to 128. What happens? For one, all the pixels in my image that have a brightness of 0-50 will have the same minimum intensity (black), but since I tell the output levels that minimum intensity (black) for me has a brightness of 128, they will turn to 128 (in other words, gray).

So, input and output levels determine how many pixels will be affected by specifying a range of input brightness, and how much their brightness will change by specifying what brightness those pixels will be mapped to.

An important thing to note here and to remember about levels is that when you specify a range of pixels, the brightness for all of the pixels is remapped with the same intensity. In our example it doesn't matter if the brightness of a pixel is 0 or 50, it will be mapped to 128. This is the fundamental difference between levels and curves.

## Eyedroppers

Understanding how the histogram is conducted and how to use the input/output levels is all you really need. The eyedrop-

pers are just a convenient way to set the Input Levels by sampling your target object.

The black eyedropper can be used to sample an area (one pixel really) on your target object. The brightness of this pixel will be used as the value of the minimum intensity (black) of the input levels. Likewise, by using the white eyedropper you can specify the maximum intensity and by using the gray eyedropper you can specify the gamma.

## Auto Levels

The Auto button on the dialog does exactly the same thing as the auto levels filter. This is a no-brainer way to alter the levels of an image.

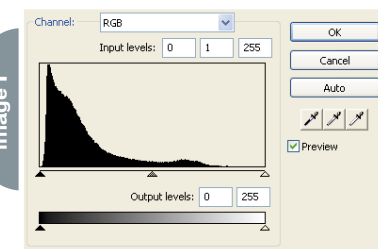
Auto levels go into each channel of the image and change the input levels. The minimum intensity (black) is set to the darkest pixel in the channel; the maximum intensity (white) is set to the brightest of the pixels in the channel, while the other shades of gray are stretched to fill out the spectrum. This method of setting the minimum and maximum intensity is also known as auto-black point and auto-white point.

Personally, I never use auto levels. The main reason is that I can achieve better results by looking at the histogram and making the adjustments myself. However, when you go into the Levels dialog, you can try Auto Levels. If you're happy with the result, by all means use it.

## Other Notes

If you want to use levels on pixel selections, you obviously can't use them as a live effect. However, this is not the only problem. Levels will be applied only to the pixels you selected, but the histogram showing on the dialog is of the whole image and not just the pixels you've selected. In other words, in my opinion this renders the histogram useless. ☹

*Kleanthis Economou has been a Web developer/software engineer since 1995 and specializes in .NET Framework solutions. He is a contributing author in various Fireworks publications and is the technical editor of the Fireworks MX Bible. As an extension developer, Kleanthis contributed two extensions to the latest release of Fireworks. kleanthis.economou@projectfireworks.com*





Using FreeHand MX with Macromedia Flash

# The Logo

## Browser Project

by Joe Sparks

Macromedia FreeHand MX is an indispensable tool for Macromedia Flash MX users. I have used FreeHand as an important part of my Macromedia Flash arsenal since I took up Flash animation – way back with the release of Macromedia Flash 4.



I often have FreeHand open while working in Macromedia Flash. I use FreeHand daily to draw complex shapes, copy them, and simply paste them right into Macromedia Flash.

The first animated character I created in Macromedia Flash was a certain floating skull named "The Radiskull." I created this character using FreeHand 7 before I started animating in Macromedia Flash. I set up all of my animated layers and movable parts in FreeHand. Check out the original article about using FreeHand 7 on the Radiskull & Devil Doll animation series ([http://joesparks.com/radiskull/production/productionnote\\_01.htm](http://joesparks.com/radiskull/production/productionnote_01.htm)).

Little did I know that several million people would soon view this silly FreeHand illustration!

Since those days, FreeHand has come a long way. The Macromedia Flash integration is much tighter and more convenient. Macromedia Flash MX does a great job supporting native FreeHand files, and vice versa. The new artistic effects and vector effects are amazing and very useful.

As you will discover, Macromedia Flash support has come so far with FreeHand MX that in some cases, you do

not even need to use Macromedia Flash to create a complete interactive Macromedia Flash movie. FreeHand MX contains authoring support for Macromedia Flash with interactive tools like the Action tool and the Navigation panel. These let you set up buttons and frames with Macromedia Flash actions that respond to the mouse and animation – and much more.

I gave FreeHand MX a good workout on a small design project, creating promotional materials and assets for San Francisco DJ Matt Hite. This tutorial explores how FreeHand MX can be a central tool for creating Macromedia Flash media. You will build an interactive logo browser – a simple interactive presentation with FreeHand MX that you will publish and view as a Macromedia Flash movie.

## Introduction

Let's say I designed a small assortment of logos with FreeHand MX. Now I want to create a useful presentation of the logos, providing a quick overview and access to the source files. I decide to try out the Macromedia Flash authoring features built into FreeHand, using them to build a simple interactive browser. The end result of this FreeHand document will be a SWF that plays in Macromedia Flash Player.

In the Logo Browser project, you will learn the following FreeHand MX tools and concepts:

- Using the Action tool to define Macromedia Flash actions inside FreeHand MX
- Using different actions from the Navigation panel
- Creating and using FreeHand MX symbols
- Using Swap Object to switch a symbol in the document
- Previewing a Macromedia Flash movie using the Test Movie button in the Controller
- Exporting Macromedia Flash Movies (saving them as SWF files)
- Fixing Macromedia Flash/FreeHand rendering differences

The browser needs a menu screen with buttons that link to pages dedicated to each logo.

The Action tool provides a simple way to specify a relationship between objects in your document and pages. You can quickly drag links between button graphics and screens and then fine-tune their relationships with the Navigation panel.

I started off this Logo Browser project by completing the menu screen and one sample logo destination page. I only created one logo destination page at this stage in the project (instead of the five pages we need) because I want to perfect that page before duplicating it for the remaining logos. Image I shows the two pages, side by side, as they appear in FreeHand MX running on Mac OS X.

Now I need to create a button or "hotspot" definition on the menu page that will link a logo menu item to its destination page. Here are the steps I took to create the button link. Refer to these steps as you build your own similar project.

1. Choose an object to serve as the button area. I could choose any element already on this page (such as a text or graphic object) to hold the link. Instead, I am going to define a larger clickable area by drawing a big rectangle in a new layer. This new object will soon serve as an "invisible" button. Take a look at the Layers panel (see Image II), where I have created a layer named "buttons" to store the rectangle.
2. Use the Action tool to drag a link from the button object to the destination page. Click the Action tool in the Tools panel. Click the button object and hold the mouse down as you drag it toward the destination page. Release the mouse when your pointer is over the page you want to link to. This will create a default link that you can see and modify in the Navigation panel (see Image III).
3. Review the Action settings in the Navigation panel. At this point, I could simply export a Macromedia Flash SWF and browse the link, but first, I want to double-check the results produced by the Action tool. In Image III, you can see the Navigation panel.

The Action pop-up menu reads "Go To and Stop," which is the default setting. The Parameters pop-up menu reads "Page 2" – the page you linked to. "Event" is set to "on (press)," which simply means that the action will take place when the user clicks on the button region.

Notice that the "Link" value is still empty (see Image III). Even though the "Go To and Stop" Macromedia Flash action does not require a Link setting (it uses the Parameter setting "Page 2"), I do this just in case I might need to export this presentation as HTML using the FreeHand "Publish as HTML..." feature (see Image IV).

Note: I will go deeper into the Navigation Panel when I work on special buttons for the logo destination pages in the next part of this project.

To complete the entire menu of buttons, I repeat the process for each logo and forge links to each of the logo pages. After creating all of the buttons and logo pages, I drag the button layer down the Layers panel, moving the yellow buttons behind all of the menu artwork and making the buttons' regions "invisible."

## Using Different Actions from the Navigation Panel

You can build interactive, Web-ready presentations right inside FreeHand MX. The Navigation panel gives the FreeHand user the power to create interactive HTML and feature-rich Macromedia Flash movies. Using built-in Macromedia Flash export capabilities, FreeHand designers can now build sophisticated Macromedia Flash movies that are ready for publishing without ever opening Macromedia Flash.

The heart of FreeHand/Macromedia Flash authoring capability is on the Navigation panel. The Action settings provide complex interactivity, special features (like printing), and animation playback controls.

Take a look at our first sample Logo page, which offers several options to the user (see Image V).

The first button, labeled "Back to logo list," is simply another jump-to-page button, which you can create with the Action tool – exactly as you created the first button on the main menu.

The next three buttons are different.

For the next button, "Print this logo," use the Print action in the Navigation panel to activate this print button as follows:

1. Open the Navigation panel and select the button graphic. Notice the Action pop-up menu is set to None by default.
2. Choose Print from the Action pop-up menu.
3. Select the page to print from the Parameters pop-up menu. (In this case it is Page 2, the page you are now editing. It could be any page in your document.)
4. Optional: Choose a layer to target for printing. In this case, I pick the "Logos" layer.

The Macromedia Flash presentation now has the capability to print. If you wish, you can export a Macromedia Flash movie right now and test it.

The next two buttons, "View at higher resolutions" and "Download FreeHand source file," access external files from our Macromedia Flash presentation using the Get URL action. The "View at higher resolutions" button accesses a GIF named Logo1a.gif. The "Download FreeHand source file" button accesses a zipped FreeHand file called Logo1a.zip.

Attach the Get URL action to these buttons as follows:

1. Open the Navigation panel and select the button graphic.
2. Type the URL of the high resolution file in the Link field. (This link will serve double duty if you choose to export your FreeHand document as HTML.) For this example, I designated the following relative URL:

logoart/Logo1a.gif

This relative URL lets me run my Macromedia Flash presentation just about anywhere. The Macromedia Flash movie will point the browser to a folder named "logoart" that exists in the same folder as the SWF. This will work on a Web site or from my hard disk.

3. Choose "Get URL" on the Action pop-up menu.
4. Optional: Choose the page target for the URL on the Parameters pop-up menu. I chose "\_blank" because I want the URL to open in a new window.

image I



image II

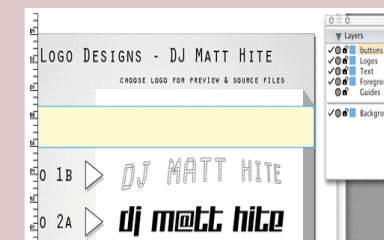


image III

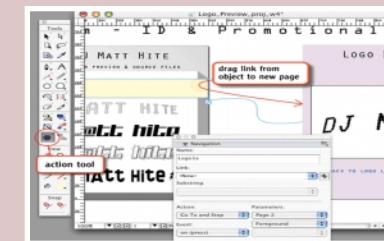


image IV

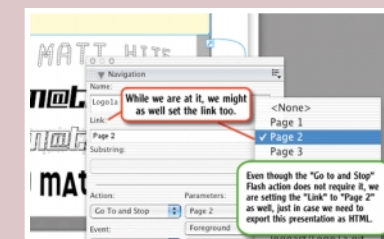


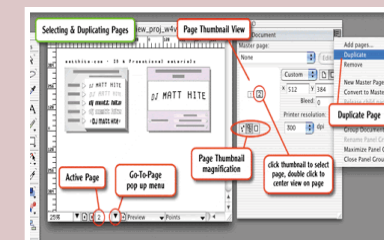
image V



image VI



image VII





5. Optional: Choose an event to trigger this action from the Event pop-up menu. In this case, I wanted a standard mouse click to trigger the action (when the user releases the mouse on the button), so I chose "on (release)." For the last button, "Download FreeHand source file," the process is identical. I simply specified a different URL, in this case:

logoart/Logo1a.zip

By zipping up the FreeHand file, the browser handles the file as a user download. The file will be transferred to the user's hard drive.

### Creating and Using FreeHand MX Symbols

Symbols give you a way to store graphics and objects in a library and then use "instances" of these symbols

throughout your FreeHand document. This is very similar to the symbols concept found in Macromedia Flash and Fireworks. The benefits can be enormous. Typically, you create a symbol for an item that you intend to use in multiple places. This saves memory and file size, because FreeHand only needs to store one copy of the object, even though it appears in multiple places and pages. The larger benefit comes when you need to edit an element in your symbol library. You edit the symbol once, and the element automatically updates everywhere it appears in your document. For example, you could have a special date symbol or a legal notice symbol that appears on every page. These are examples of elements that change frequently, therefore they are good candidates for turning them into symbols.

In the logo browser, I am going to store the logo designs as symbols. This is a very easy process:

1. Open the Library panel. In the work space, select the object you wish to make into a new symbol. In Image VI, I have selected the first logo.
2. Drag the object from the FreeHand work space into the Library list to create a new symbol from your selection. Your original selection will transform into an instance of the new symbol. You can also perform this task by choosing Modify > Symbol > Convert to Symbol, or by pressing F8.

You can also create a new symbol by clicking the Create Symbol button at the bottom of the Library palette, but your selection will not become an instance of that symbol. In other words, a symbol will be added to the Library, based on your selection, but the selection itself will remain a FreeHand object. This is the equivalent of choosing Modify > Symbol > Copy to Symbol in the FreeHand menu.

Your new symbol will appear in the library with a default name.

3. Rename the new symbol to a mean-

ingful name. You can click the symbol name to edit it, or select the symbol and choose "Rename" in the Library panel pop-up menu.

4. Add as many symbols as you need. For the logo browser, I added all of the logos from the menu page.
5. Manage your symbol library by creating folders or groups to contain symbols. Click the New Group button at the bottom of the Library panel to add a group folder to the library.
6. Rename the folder and drag symbols into it.
7. To edit a symbol, select it in the Library Panel and choose "Edit" from the panel pop-up menu. A new window will open for the symbol.
8. Edit the symbol with FreeHand tools and close the window when you are finished. FreeHand will update all instances of the symbol in the document.
9. To use the symbol anywhere in your FreeHand project, simply drag the symbol from the Library panel onto your document. This is exactly what I did on the destination page for the first logo.

### Using Swap Object to Switch a Symbol in the Document

When you need to remove a symbol on a page and replace it with another from your library, you really only have two choices:

1. Delete the existing symbol instance from the page. Drag out its replacement and slide, scale, rotate, or nudge it to position the replacement symbol somewhere near the location of the original one.
2. Swap the symbol with one button click. Swapping a symbol is often far better than the previous alternative.

Before getting into the symbol swapping, I need to duplicate some pages, which will give us some logo symbols to swap.

At this point in the logo browser project, I have completed all of the features on our logo destination page. All I need to do now is duplicate this page four times and customize it for each of the remaining logos.

Here's a quick overview of selecting and duplicating pages. You can select

the page you wish to duplicate by using the Go-to-page pop-up menu at the bottom-middle of the document window or you can use the Document page to select the page directly. Once you are sure you have the right page selected, you may use the Duplicate command from the Document pop-up menu (see Images VII and VIII).

Now I am ready to customize each duplicated page for the rest of the logo design. I will use swap symbol to change the logo first. Here are the steps:

1. Select the symbol that you want to swap.
2. Open the library and pick the object you wish to swap with. In this case, I have selected a logo in the document and then selected the second logo in my library (see Image IX).
3. Click the Swap Symbol button. Immediately the instance will change to the selected symbol in the library. The new symbol instance will inherit all of the scale, skew, and other attributes of the previous instance (which can be good or bad, depending on the native aspect ratio of the two symbols) (see Image X).

That's all there is to swapping a symbol instance. Now I need to update the rest of this page. All I need to do is change the title from "Logo 1a" to "Logo 1b" and modify the links on the three custom buttons.

Modifying the buttons is very simple. The "Back to Menu" button is already done, since that button always goes to the same page. For the other three buttons, I simply open the Navigation panel and make the following edits:

1. Choose Print button and pick "Page 3" on the Properties pop-up menu
2. Choose View at Higher Res button and change the link to "logoart/Logo1b.gif"
3. Choose Download Source button and change the link to "logoart/Logo1b.zip"

One logo page down, and three more to go! I simply repeat the above steps using the proper logo and files names (swap symbol, change title, edit buttons) for each of the remaining pages.

To complete this project, there's only one more thing left to do. I must

go back to the main menu page and make buttons for the last four logos. I must draw four more button regions and use the Action tool to link to my new logo pages (performing the steps outlined at the beginning of this project article). When I am done, the five buttons will appear as shown in Image XI.

In this case, I decided to simply hide this layer of buttons behind all of the other layers. I didn't use the layer for anything else, and the end results work great in Macromedia Flash.

### Previewing the Macromedia Flash Movie Using the Test Movie Window

As you develop Macromedia Flash projects with FreeHand, you can test your Macromedia Flash work from within the FreeHand application. This can save time and development cycles for you by removing the extra steps of exporting/saving, naming, finding the exported file, and viewing with Macromedia Flash Player.

To preview your FreeHand document as a Macromedia Flash movie:

1. From the Window menu, choose Movie > Test.
2. Control playback with tools found on the Movie Preview window and/or the Controller toolbar.

FreeHand will convert the document into a Macromedia Flash movie file (a SWF file), and this may take a little time to process, depending on the size and complexity of your project.

After this is complete, FreeHand will open a new window for playback. FreeHand users who are working on Macromedia Flash movies should open the Controller toolbar (Windows > Toolbars > Controller), because it provides quick access to key tools for Macromedia Flash authoring. In Image XII, the Movie Preview window and the Controller toolbar are featured.

As soon as the Movie window opens, you may begin testing your Macromedia Flash movie. You can try out your Macromedia Flash Actions, click your buttons, play back animations, and so on. The Movie playback controls on both the Preview window and the controller let you stop, rewind, step backward, play,

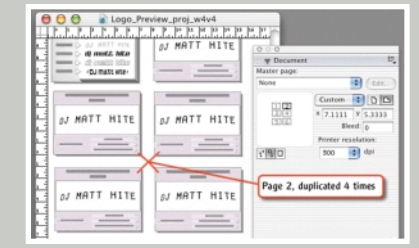


image VIII



image IX



image X

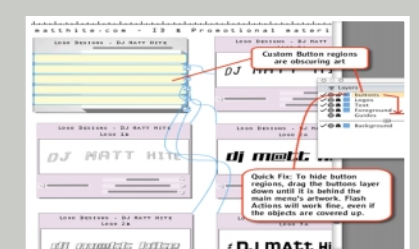


image XI



image XII

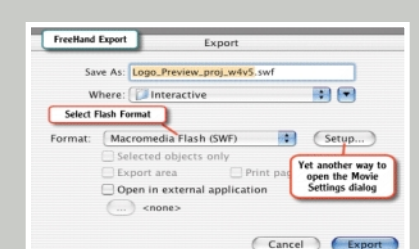


image XIII

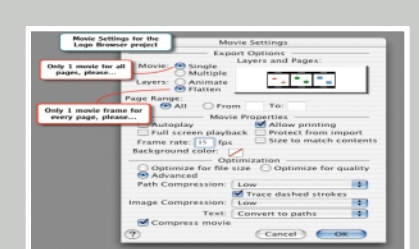


image XIV





step forward, and fast forward – in that order.

The Controller toolbar duplicates all of the functions found on the Preview window, but adds a helpful Test Movie button, an extra Action tool, and a shortcut to the Navigation panel.

### Exporting Macromedia Flash Movies (Save as SWF)

A Macromedia Flash movie is one of the best ways you can share your presentation with the world. Nearly every computer can play a SWF file (the compressed Macromedia Flash file format). Even illustrators and FreeHand users who have no need for interactive presentations will find that exporting a single-image SWF is often the very best way to share your work. The end result is a very lightweight file that can scale to any size.

Exporting a SWF for our little Logo Browser project is as simple as it gets.

1. Select Export... from the File menu. The dialog box shown in Image XIII appears.

2. Select Macromedia Flash (SWF) from the Format pop-up menu.

Next to the Format menu, there is a Setup button. When the format is a SWF, this Setup button will take you to the Movie Settings dialog box, where you can instruct FreeHand on the particulars of SWF export. If everything played well during the "Test Movie" stage, then you should be all ready to export. Since you are not doing animation, there is very little to worry about; the default Movie Settings should handle it just fine. Even so, you should double-check the settings (see Image XIV).

The Movie Settings (also accessible from the Preview window, the Controller toolbar button, or off the menu at Windows > Movie > Movie Settings) are extremely important for Macromedia Flash authors. This is where you tell FreeHand how to export the FreeHand document as a movie and how to interpret layers as animation. There are also other important settings in this area.

For now, I will export the project "Project 1 Result: Logo Browser for Flash Player" as a SWF file. You can check it out by downloading the source files from [www.sys-con.com/mx/sourcec.cfm](http://www.sys-con.com/mx/sourcec.cfm)

### Working Around Macromedia Flash and FreeHand Rendering Differences (Line Scaling)

Every now and then, an element that you have created in FreeHand (or even from within Macromedia Flash) ends up looking far different than you would expect under certain situations. This can be frustrating. In this section, I will clarify some things about how to deal with how scaled vector lines render in Macromedia Flash and show you strategies for working around some rendering problems. If you follow this section, you may save a great deal of time later on.

If you were paying attention to the fourth logo in Image XII, you may have noticed that the stars around the logo seem very dark compared to how the logo looked inside FreeHand MX and the higher resolution GIF. Take a closer look at Image XV.

This "different look" in Macromedia Flash Player has little to do with the FreeHand SWF export, rather, it stems

from the way Macromedia Flash renders fixed or minimum line widths – regardless of scale. This happens with artwork created inside Macromedia Flash as well. In fact, a fixed line width is truly a generic feature of all vector illustration software – and the root problem in this particular situation.

If you have worked in Macromedia Flash for a while, you may have encountered some rendering problems where the line color starts to crowd out the fill color on vector shapes. In this section, you will find some valuable tips for coercing better rendering out of Macromedia Flash when this problem occurs.

Here's is a demo of the line weight problem from within Macromedia Flash MX. First, I will draw a simple graphic right into the time line, using the Macromedia Flash drawing tools.

Next, I will take this simple drawing and scale it, in a number of ways, to demonstrate what can happen when you scale the line art (see Image XVI).

In Column A, you can see the heart of the problem. When you scale vector graphics that exist solely in the time line (not as a symbol), the tool scales the size of the shape, but it does not scale the line width on the graphics – it remains fixed. Grouping the graphic before you scale also does not help.

In Column B, see the results of our "Primary Solution," which is to take the art from A1 and turn it into a symbol in the Macromedia Flash library (a movie clip in this case). When you scale the symbol in the Macromedia Flash time line, Macromedia Flash will scale the line width relative to the scale of the symbol. This works quite well in most cases.

There's only one catch – Macromedia Flash has a minimum line width setting of .25. Once your line width reaches this size, Macromedia Flash cannot draw them any smaller. Therefore, certain kinds of art with many tiny lines to begin with will still reach a point where the rendering is all line and no fill.

In Column C, see the "Extreme Solution," which can really work wonders with certain kinds of problematic line-rich artwork. In this case, I also created a symbol, but took it another step by selecting all of the lines in the graphic and converting them to fills. This is "extreme" because it makes the artwork much more difficult to edit. You should

create a backup of the symbol before you convert the lines to fills. To convert lines to fills, simply select the lines you wish to convert, and select the menu command: Modify > Shape > Convert Lines to Fills.

In general, it is very important to move your "actual size" (100% scale) artwork into a symbol before you scale it to a parent time line, especially if you intend to convert lines to fills. If you select actual vectors and scale the original artwork down in Macromedia Flash (as opposed to scaling a symbol down) you can destroy, erode, or even delete vectors altogether.

So what does this have to do with the FreeHand logo, which displays differently in the Macromedia Flash export? Knowing the above information about line widths, export the logo as a SWF and import it into Macromedia Flash MX. This way, you can see how FreeHand groups the vectors during the conversion process (see Image XVII).

Our goal is to make the Macromedia Flash version of the logo look more like it does in FreeHand when rendered at smaller scales in the Logo Browser project. As a first test, let's put this whole logo into a new Macromedia Flash symbol and scale it down (see Image XVIII).

Remember that the "Primary Solution" called for putting the graphics into a symbol, but FreeHand has already put the stars in a symbol for us, thereby causing a minimum line size problem. Those tiny stars are so tiny, the fill gets obscured by the smallest line width inside Macromedia Flash.

It is time to bring out the Extreme Solution!

In the example shown in Image XIX, I simply selected the lines on the most frequently used star symbol and used the "Convert Lines to Fills" command. Image XX shows the results of that maneuver.

As you can see, many of the stars around the logo now appear almost exactly as they do inside the FreeHand window. We can do the same trick on the remaining stars symbols, and call this operation a success.

You may encounter the inverse problem to the minimum line width, which is the maximum line width problem! There are situations where you scale art work higher than 100%, and the lines appear too thin. You can use the techniques discussed here to deal with this problem too.

You could instead use an alternate route: Convert to Image.

However, there is another option inside of FreeHand MX. Although this solution is neither the most elegant solution nor the best choice for file size and quality issues, it is quick, easy, and very reliable. Find this feature, called "Convert to Image," in FreeHand, on the menu: Modify > Convert to Image.

To turn any FreeHand selection into a bitmap, choose Convert to Image from the Modify menu. Say goodbye to vectors and hello to safe, reliable bitmaps! Never use your original: always perform this operation on a copy of your original. You do not want to lose your vectors forever.

By converting the problem logo to an image (bitmap), your Macromedia Flash presentation would look the same as it does in FreeHand. However, you should consider this a hack method with some drawbacks: you increase file size and your printing quality suffers. In short, your Macromedia Flash presentation would no longer scale as effectively.

You can work around some of this problem by increasing the resolution of the bitmapped image during conversion (see Image XXI). However, a higher resolution can greatly increase the file size of your presentation.

If you can accept the drawbacks of using bitmaps in place of pristine vectors, then always remember that you have the Convert to Image option in FreeHand MX, and this option is completely compatible with SWF export.

Note: There is another technique that I have used, that I will mention in passing – separating lines and fills into two separate layers, with the fill layer in front of the line layer. In situations where you really need to preserve your line art while moving drastically between different sizes, this has some advantages worth looking into.

Joe Sparks is an artist, animator, musician, and game designer. His work has garnered many industry awards, including New Media's Award of Excellence and Macworld Magazine's Game of the Year. Sparks is known for pioneering the multimedia games *Spaceship Warlock* and *Total Distortion*, and more recently, for his popular animation series *Radiskull* & *Devil Doll*. [joe@joesparks.com](http://joe@joesparks.com)



image XV

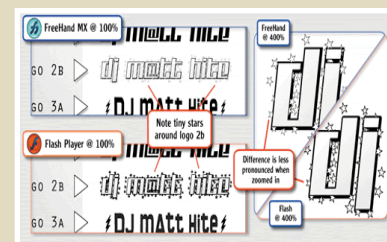


image XVI

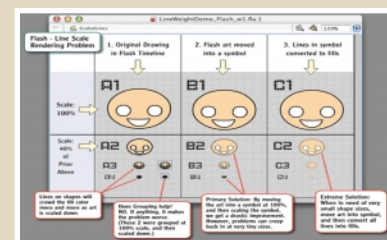


image XVII

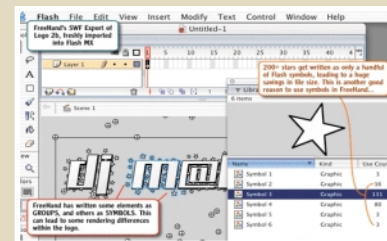


image XVIII

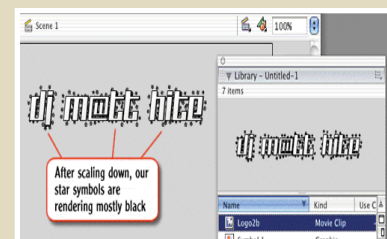


image XIX

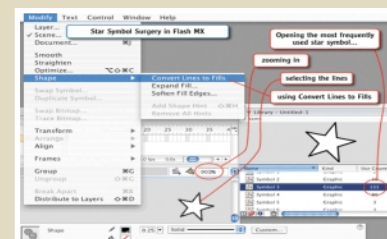


image XX

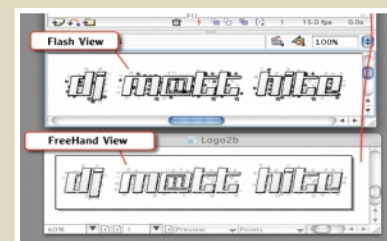
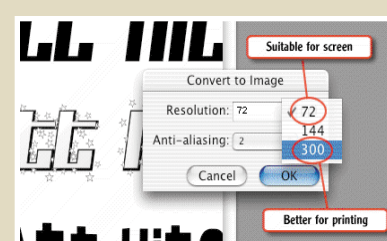


image XXI



# ColdFusion Components Under a Red Sky

*Now is the time to discover how much they can aid in your development*  
by raymond camden

By the time you read this, the next update of ColdFusion MX, code-named Red Sky, will have been released. This new version is called ColdFusion MX 6.1. Much will be made of the improved speed under MX 6.1, as well as the dramatic improvements to cfmail and other areas.

However, one of the most exciting aspects of MX 6.1 is the CFC bug fixes and improvements. In this article we are going to discuss a few of the more important bug fixes, as well as some of the new features supported in MX 6.1. Then, I'll show you a simple way to add an "isRedSky" check to your code to ensure your CFCs run only under the MX 6.1 version of ColdFusion.

Let's begin by taking a look at a few of the big bugs that are fixed in MX 6.1.

## Using Includes Inside CFC Methods

Imagine you have a CFC that is quite large. You decide to break up the file a bit by moving some of the larger methods into include files. However, you immediately run into an issue trying to use arguments. In other words, if a CFC method takes two arguments, name and age, they would not be available to the included template. This is just one of the bugs fixed in MX 6.1. However, there is one thing to remember. While you can move most of your code into the included file, `<cfargument>` tags, var statements, and the `<cfreturn>` tag must be in the main CFC template. So, an example method might look like that shown in Code I.

This method looks like any other – we begin by defining a set of arguments followed by a set of var statements. Then we have our `cfinclude`. Finally we have the `cfreturn` tag. Our method code inside `_test.cfm` could be anything.

## The 'Page Context' Bug

The so-called "page context" bug is probably the biggest issue corrected in MX 6.1, which is wonderful since this bug alone has probably caused many CFC developers quite a few hours of frustration. This bug is hard to debug since it behaves so oddly. To see this bug in action, let's take a look at some sample code. Code II defines a very simple CFC.

This method has one method, `dump`, that uses the `<cfdump>` tag to dump both the `This` scope as well as the result of `getMetaData()` on the `This` scope. Note that it dumps this result directly instead of using `<cfreturn>` to return the data. Normally a typical CFC would have many other methods. Code III simply demonstrates how we might use this CFC.

This script simply creates an instance of the CFC we defined earlier and stores it in the Application scope. As you can tell, it will only do this on the first hit since variables in the Application scope will be cached. Last, we call the `dump` method on the CFC.

If you run this code on a non-MX 6.1 version of MX, something interesting will happen. The first time you run the code, it will work perfectly. The second time you run it, nothing will be displayed. You won't get an error, nor will you get any output. I have to say, the first time I ran into this bug I thought I was going crazy. On a whim I decided to restart MX, which of course cleared the Application scope. This just drove me even more nuts. Finally I got confirmation that this was an example of the "page context" bug.

Another example of this is a CFC that uses the Application scope. You may be saying, aren't you already doing that in Code III? No – Code III shows an example

of storing a CFC in the Application scope. The CFC has no idea it's being stored there. What I'm talking about is a CFC that inside itself references the Application scope. For example, a CFC method with a database query may reference `application.dsn` for the data source. This method would fail when the CFC is cached as in Code III. The CFC simply "loses" access to the Application scopes, as well as all other scopes. It also loses the ability to "write" directly from the method, as we did in our `dump` method.

If all this sounds a bit confusing, the important thing to remember is that it all goes away in MX 6.1. Some people will say you shouldn't reference "outside" scopes from within the CFC anyway. And outputting directly from a CFC method should rarely be used since it means the CFC method cannot be called from Flash Remoting.

So, we've talked about two issues that are fixed with CFCs in MX 6.1. Other issues fixed in MX 6.1 include the use of `<cftransaction>` around CFC calls, the use of the Variables scope in CFCs, as well as issues with CFCs under load. Let's talk about some of the new features of CFCs in MX 6.1.

## It's a Bird, It's a Plane, It's Super Method!

One of the coolest new features of CFCs in MX 6.1 is `super()` support. Those of you who have done some Java development, or other truly OO development, will recognize what this means. For those of you who have not, let's consider a simple case. Imagine you have a CFC for your products. One of the methods generates the price for the product. This isn't as simple as returning the price field for the particular item. The method actually takes taxes into consideration as well as

any other promotions. This method is called `getPrice()`.

Now consider another CFC that extends the product CFC, `Book`. One of the ways `Book` products differ from other products is that their prices are generated differently, but only slightly differently. We want to use the same formula that the Product CFC uses, but we then need to further modify that result. Before MX 6.1, we would have to either use a different method name, or cut and paste the code from `Product.cfc`. However, under MX 6.1, we can use `super.methodName` to call a method in a parent class. This means `Book` can have its own `getPrice()` method that looks like that shown in Code IV.

This method uses the `super` feature to call the `getPrice` method in its parent. Since `Book` extends `Product`, this means `super` directly references the code for `Product's getPrice` method. What's great about this syntax is that it means I can greatly simplify the code in `Book's getPrice` method(). Once I have the result of the more generic `getPrice` call, I can then apply the particular pieces of logic that books need when determining their price. (In this example I simply add 5 then multiply by 2. In the real world you would have more meaningful calculations.)

## Checking for MX 6.1

As you know, each version of ColdFusion MX (from 1.0 to the various updates to "Red Sky" or 6.1) has had various build numbers that represent the version. These numbers are not very easy to remember. However, if all you care about is if a server is MX 6.1 or not, you can use a simple trick. (This trick can be modified to check for MX compared to CF5 as well.) Simply use the results of `getFunctionList` to check for something that exists in MX 6.1. In our example UDF, we will use `wrap`, a function added in MX 6.1 (see Code V).

code I

```
<cffunction name="test" returnType="string"
    access="public" output="false">
    <cfargument name="name" type="string" required="true">
    <cfargument name="age" type="numeric" required="true">
    <cfset var x = 0>
        <cfset var i = 0>
    <cfset var str = "">

    <cfinclude template="_test.cfm">

    <cfreturn str>
</cffunction>
```

code II

```
<cfcomponent output="false">

    <cffunction name="dump" access="public" returnType="void" output="true">
        <cfdump var="#this#" label="This">
        <cfdump var="#getMetaData(this)#" label="getMetaData(this)">
    </cffunction>

</cfcomponent>

<cfapplication name="bug1">
```

code III

```
<cfif not isDefined("application.pageContextBugCFC")>
    <cfset application.pageContextBugCFC = createObject("component", "page-
contextbug")>
</cfif>

<cfset application.pageContextBugCFC.dump()>
```

–code continued on next page

This UDF simply takes the result of `getFunctionList` (which is a struct for some strange reason), calls `structKeyList` on it, and then searches the list for the new wrap function. This code could be placed in the constructor of a CFC to ensure the CFC runs only on an MX 6.1 box.

## Conclusion

I hope this article encourages you to check out MX 6.1. If you have not yet played with ColdFusion Components, now is a wonderful time to begin discovering how much they can aid in your development. ☺

*Raymond Camden is co-technical editor of ColdFusion Developer's Journal and a senior software engineer for Mindseye, Inc. A longtime ColdFusion user, Raymond is a co-author of the "Mastering ColdFusion" series published by Sybex Inc, as well as the lead author for the ColdFusion MX Developer's Handbook. He also presents at numerous conferences and contributes to online webzines. He and Rob Brooks-Bilson created and run the Common Function Library Project (www.cflib.org), an open source repository of ColdFusion UDFs. Raymond has helped form three ColdFusion User Groups and is the manager of the Acadiana MMUG.*







code IV

```

<cffunction name="getPrice"
access="public" returnType="numeric"
output="false">
    <cfargument
name="productID" type="numeric"
required="true">
    <cfset var subtotal =
super.getPrice(productID)>
    <cfset var total = subto-
tal>

    <cfset total = total + 5>
    <cfset total = total * 2>

    <cfreturn total>
</cffunction>

```

code V

```

<cfscript>
function isRedSky() {

if(listFindNoCase(structKeyList(getFunct
ionList()),"wrap")) return true;
    else return false;
}
</cfscript>

<cfoutput>Is this a MX 6.1/Red Sky box?
#isRedSky()#</cfoutput>

```

-continued from page 10

Ah, but what about lock-in? Once you go to SWF you're committed to that format, so you're locked in as with other technologies, right? Well, no, not really. It's hard to find anything else with the same range of abilities as the Macromedia Flash Player, so direct export to other formats always loses some features. But today there are various ways to parse and abstract SWF files to produce XML representations that can then be constructed into other types of deliverables.

Abstraction models do differ, so when you do a Web search on "swf xml translate" you'll find some XML transforms that focus on visual representations, media reps, or interactivity representations. Translating SWF to XML can now be done in various ways to various ends.

### Conclusion

So, is the Macromedia Flash Player a "platform?" In some ways it is, because it

lets you predictably run certain operations on machines you don't control. In this sense it's the most popular and exciting platform in the world. But if so, it's a cooperative platform that works with the widest range of authoring, delivery, and viewing technologies. Because it's constrained to a platform-neutral layer it's also safer and more sustainable than other platforms out there.

If you want to talk about "the Macromedia MX platform," then I won't argue with you, although you'll probably find others to debate you on this. But I usually just give up and call it "the platform that isn't a platform" myself... those self-negating statements usually tend to lead to shorter e-mail threads anyway... :-)

*John Dowdell came to Macromedia through user groups in the '90s, as tech support across the tools, particularly online. Today he mostly listens to the online discussions, distilling customer comments and evangelizing these within the company. jdowdell@macromedia.com*

# Liberate your ColdFusion Talents

**NQcontent v2**  
WEB CONTENT MANAGEMENT

The only CMS with the developer in mind

NQcontent is the ColdFusion based Content Management platform that eliminates repetitive development work and extends your talents to give you room to deliver robust, dynamic, websites that work all the time!



- Best Content Management Tool 2002 WINNER
- Most Innovative Application 2002 WINNER
- Best e-Business Software 2002 WINNER
- Best Web Application 2002 WINNER



You don't have to be Leonardo to draw a "perfect" circle.



You can use a compass.



You can manually code your dynamic websites, or have a tool do it for you. Use InterAKT products to unleash your creativity when developing complex PHP applications.

**Professional Web Tools**  
for Dreamweaver MX PHP programming

<http://www.interakt.ro/products/>

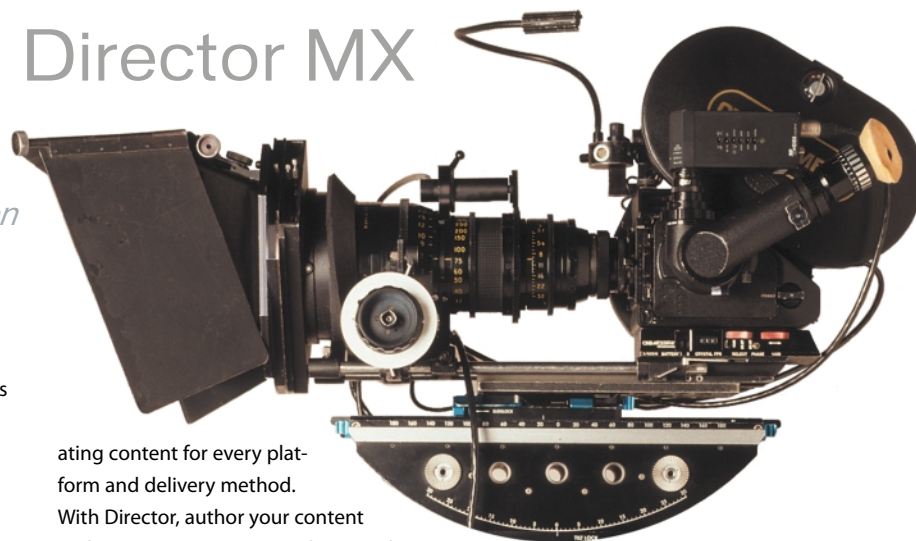


**Interakt**

# Introducing Director MX

*More power, more access, better integration*

by miriam geller



macromedia  
 McDonald's, Toyota, BMW, Toys R Us, Harry Potter, Mattel, Disney. All of these companies and countless others have turned to Macromedia Director during its 14-year history – to get a job done that was otherwise impossible without this powerful application. Since the 1980s, Director has enabled businesses to use multiple media types to communicate messages effectively. It's just that simple. Director allows businesses to communicate with customers in the most interactive and engaging ways possible.

Fourteen years later, still going strong, Director MX is a completely up-to-date application that is just as fresh today as the day it was born. And now it has joined the MX product family with a new user interface and tight integration with the other MX products to ensure that developers can create effective rich content and applications.

With Director MX, you can make content accessible to computer users who have hearing, visual, or mobility impairments by using text-to-speech (TTS) capabilities (with the new Speech Xtra), captioning, and completely customizable tab navigation.

## Powerful

Director MX allows you to build your project efficiently for the Web (through Shockwave Player) or as a stand-alone application. Create presentations, applications, and games using the media you prefer – audio, video, graphics of all sorts – the choice is yours.

Then, deliver your content wherever your message makes the most impact.

Don't waste development time recre-

ating content for every platform and delivery method.

With Director, author your content and your users can view it wherever they are – whether they are on the Web, playing a CD/DVD, or using a kiosk.

Powerful means you're never limited by the application. Director is completely extensible through extensions called Xtras. There are hundreds of Xtras available to take your projects in whatever direction you choose. For instance, if you want to add joysticks to a kiosk-based project, you can...if you use Director.

## Integrated

Macromedia knows that many developers use Macromedia Flash and Director together, but that the workflow between these products has not been as streamlined as it could be – until now. In this release, Macromedia has optimized the workflow and process.

With Director MX, developers can now import Macromedia Flash MX files and then edit them quickly and efficiently with the new Macromedia Flash launch-and-edit functionality. Additionally, Director developers can now access and control elements in a Macromedia Flash movie from directly within the Director application, using Lingo control over Macromedia Flash objects.

For instance, if you were to insert a Macromedia Flash movie that has a blue house and then you decide to change it to green, you can change its color from within Director without having to launch Macromedia Flash. If you want to create a Macromedia Flash XML socket to use in Director, you can create it in Lingo.

Developers now have more control over how they use these applications together.

Director applications can even connect to servers, including ColdFusion MX and Macromedia Flash Communication Server MX. Director MX includes a copy of Macromedia Flash Communication Server MX Personal Edition, so you can get started right away.

## Accessible

Accessibility is especially important for Director developers, the vast majority of whom create content that must adhere to federal accessibility guidelines. Director MX enables developers to create accessible content in unique ways. With Director, you can create applications that are self-voicing for visually impaired users, tab navigable for physically impaired users, and captioned for hearing-impaired users. The self-voicing capability does not require a screen reader, thereby broadening access to accessible content.

You can update existing Director projects easily to fulfill accessibility guidelines by using drag-and-drop behaviors. Moreover, accessible Director applications can be deployed on both Macintosh and Windows machines in a browser or as stand-alone applications.

*Miriam Geller is the senior product manager for Director and Shockwave Player. She has been with Macromedia since 1999 driving development of these products. Miriam has been in the industry for a dozen years focusing on both hardware and software technology development. mgeller@macromedia.com*



how many people does it take to change the web?

macromedia  
 COLDFUSION  
 MX

See for yourself. Upgrading to ColdFusion MX gives you amazing new features and a powerful Java™ architecture. Incorporate XML and web services with ease. Build flexible and maintainable applications with ColdFusion Components. Integrate with J2EE™ and .NET to deliver rich user interfaces with native connectivity to Macromedia Flash™. Easily migrate existing applications and save time with the powerful new Macromedia MX development tools. Try ColdFusion MX today and see what you can do now.

Download the free 30-day trial at [www.macromedia.com/go/cfmxad](http://www.macromedia.com/go/cfmxad)



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



**ADVERTISE**

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

**SUBSCRIBE**

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

