Flash Application Design Solutions

The Flash Usability Handbook

Ka Wai Cheung
Craig Bryant

friendsof
DESIGNER TO DESIGNER™
an Apress® company

Flash Application Design Solutions: The Flash Usability Handbook

Copyright © 2006 by Ka Wai Cheung and Craig Bryant

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-59059-594-7

ISBN-10 (pbk): 1-59059-594-7

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit www.apress.com.

The information in this book is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is freely available to readers at www.friendsofed.com in the Downloads section.

Credits

Lead Editor
Chris Mills

Technical Reviewer
Paul Spitzer

Editorial Board
Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser, Matt Wade

Associate Publisher
Grace Wong

Project Manager
Beth Christmas

Copy Edit Manager
Nicole LeClerc

Copy Editors
Nicole LeClerc and Marilyn Smith

Assistant Production Director
Kari Brooks-Copony

Production Editor
Ellie Fountain

Compositor
Dina Quan

Proofreader
Dan Shaw

Indexer
Lucie Haskins

Artist
April Milne

Cover Designer
Kurt Krames

Manufacturing Director
Tom Debolski

To Mom and Dad
— Ka Wai Cheung

To the fondest reader I've known, Peggy M. Bryant
— Craig Bryant

CONTENTS AT A GLANCE

CONTENTS

PART TWO: THE USABILITY SOLUTIONS

ABOUT THE AUTHORS

Ka Wai Cheung has been creating web applications ever since he was a child (he was still a child at 19). He currently leads the user interaction design of Gritwire (`www.gritwire.com`), a Flash-based RSS reader and social network for daily Internet users.

Ka Wai has written for several online publications and resource sites, including Digital Web Magazine (`www.digital-web.com`), ActionScript.org (`www.actionscript.org`), and HOW design online (`www.howdesign.com`). He writes about topics from web standards to usability design to software development theory, but his main passion is Flash application development. He logs his past web projects and writings on his site, Project99 (`www.project99.tv`).

Ka Wai majored in Math, Computer Science, and Integrated Science at Northwestern University, where he first toyed with Flash by tweening a wingding character across his 640-pixel wide screen.

When not working on the Web, Ka Wai enjoys playing guitar, eating foods from all four corners of the world, and watching his beloved Chicago sports teams occasionally win.

Craig Bryant is a full-time Flash engineer who has been advocating the use of Flash in delivering rich Internet applications to top-ranked online communications agencies around the world for the past five years. Currently, Craig is a Senior Art Director at Arc Worldwide in Chicago.

Craig's devotion to Flash has driven his creative intrigue, while providing him with a technical expertise and perspective that only a Flash developer can attain.

Like many other Flash developers, Craig's background is not strictly programming-oriented. He has a degree in musical composition from Berklee College of Music in Boston.

When he isn't racking his brain in Flash, Craig is seeking out great music, catching up on twentieth-century American fiction, or awaiting a deep-dish pizza being delivered to his doorstep.

ABOUT THE TECHNICAL REVIEWER

Paul Spitzer is a software engineer living in Berkeley, California. Paul's Flash experience began in 1998 with Flash 3, and his primary focus has been user interface engineering in Flash/ActionScript. He has also worked with a variety of other technologies, including C#/.NET, WinFX, Java, and ColdFusion, to name a few. In his current job at San Francisco–based Fluid, Inc., Paul works closely with information architects, usability experts, and visual and interactive designers to realize state-of-the-art user experiences.

ACKNOWLEDGMENTS

I have learned an unbelievable amount about Flash, book writing, and the benefits of sleep over the past several months. First, I'd like to acknowledge my bed for being there for me through thick and thin. Sorry for neglecting you for so long; you'll be seeing a lot more of me now. As far as humans go . . .

First, Craig Bryant, this book's coauthor, for working on this book while juggling an already demanding schedule, and writing a ton of code.

Chris Mills, our editor, for giving us this great opportunity to write and for his overall wisdom and sage advice throughout the entire process.

Paul Spitzer, our technical reviewer, for policing our code, coming up with innovative refactorings and wise suggestions, and teaching me a new trick or two about Flash. Not all of his great suggestions made it into the final publication, but they should find their way into the source code updates for this book at the friends of ED website.

Nicole LeClerc and Marilyn Smith, our copy editors, for making everything we wrote make sense; Beth Christmas, our project manager, for keeping the house in order and her incredible amount of patience; Dina Quan, our compositor, for laying out these book pages so wonderfully.

Adam Hoffman, my former boss and senior architect at Hubbard One, for giving me a new appreciation and understanding of object-oriented programming and usability over the past three years and who I still rely on for some much-needed expertise.

Ian Carswell, COO of Dizpersion Technologies, whose creative mind and never-say-never attitude have taken me to places in Flash I never thought existed.

Kevin Sidell, founder of Dream Marketing Communications, for literally hundreds of web projects that have allowed me to exercise my design skill set over the past seven years.

Geoff Stearns for allowing us to showcase and discuss his creation, FlashObject, in Chapter 13.

Ka Wai Cheung

First and foremost, thanks to Ka Wai Cheung for doing what it takes to make this a fantastic book and not just a bunch of rowdy-looking code samples. Your dedication and drive are the reason this book exists.

Thanks to the fine folks at Apress and friends of ED for the great opportunity and the patience throughout, as well as doing a great service to the design and development community.

Gratitude to my wife, Heike, for kicking me out of bed and chaining me to my desk. OK, it wasn't that extreme, but you keep me on my toes.

Todd Levy and Matt Macqueen, thanks much for the support you've provided—"uncharted" territory is always easier to navigate with help from guys like you.

Last but not least, a giant shout out to both Macromedia and the Flash development community for the ongoing commitment to Flash and the wealth of knowledge and support they have helped disseminate to the thousands of developers out there. Without you, we'd still be trapped in a <table> somewhere.

Craig Bryant

INTRODUCTION

Have you ever noticed how complicated all the instrument panels looked in those futuristic movies and TV shows from years past? The fact that Captain Kirk could operate the myriad buttons, lights, and widgets on the Starship Enterprise was a feat in itself. Sure, the producers of *Star Trek* probably weren't thinking a great deal about how usable the instruments should appear on-screen. Instead, they created a depiction of what this monster we call technology must look like in the future: a mess of complicated processes understandable only to a select, brilliant few. See Figure 1 for another example.

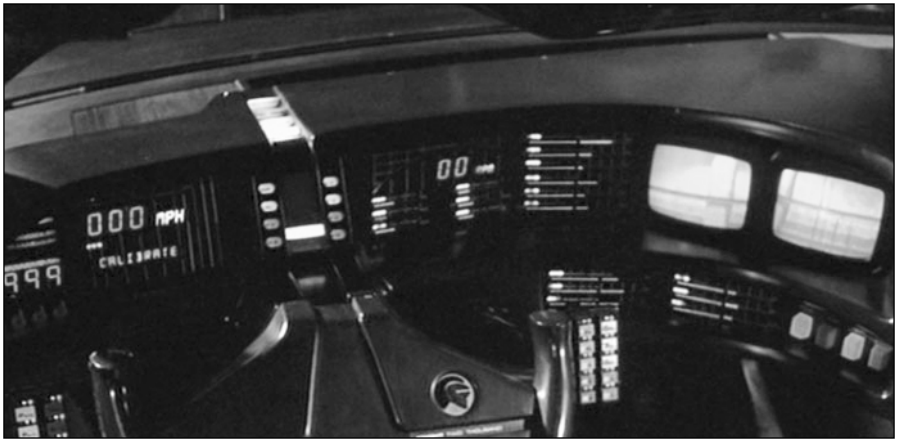


Figure 1. The classic 1980s TV series *Knight Rider* featured a talking car named Kitt that could think on its own. But how would you operate a car with a dashboard as confusing as Kitt's?

Unfortunately, it's not just at the movies or from prime-time TV where we get the impression that technology is difficult to use. From radio clocks, to stovetops (see Figure 2), to computers, to the Internet, you'll find plenty of examples of poorly designed interfaces masking otherwise great engineering. Take a multi-disc DVD player, hook it up to a TV with multiple video ports, and try to navigate through to the favorite scene of your movie with an all-in-one TV/DVD/VCR remote. If you can get through it all without a few mental stumbles along the way, and without the aid of the instruction booklet, maybe you can help the rest of us!

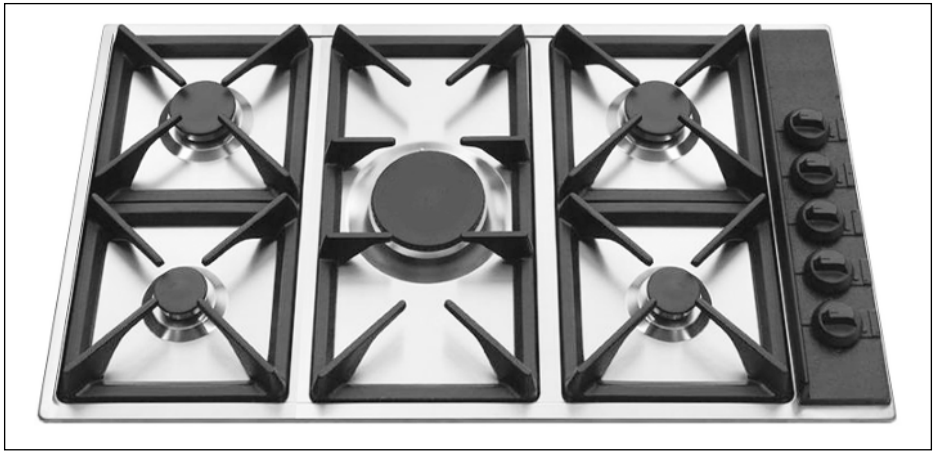


Figure 2. The classic stovetop usability problem: which knobs turn on which burners?

Over the past decade, we've encountered the same kinds of frustrations with the Internet. Superfluous Flash intros, slowly loading sites, obtrusive JavaScript alert boxes, complex navigation, excessively long pages, broken links, unexpected pop-up ads, and unintuitive forms are just a few of the usability problems we've seen in the brief history of the Web. As web designers and developers (if you're reading this book, you more than likely fall somewhere within or near this category), we know that technology works best when we can interact with it like it's our best friend rather than our worst enemy. In general, we've coined this concept under the term *usability*. But what exactly does usability mean? Is it something more than just a sophisticated term for easy to use?

Defining Flash usability

Because this is a book on developing usable web applications with Flash, we're going to define usability as follows:

> *Usability measures how intuitive, efficient, and pleasurable the experience of using a Flash application is, as well as how effective the application is in achieving a user's end goals.*

Let's take a closer look at what this means:

- **Usability should be intuitive.** The more obvious you can make the features of your web application, the better. Users should be thinking less about how to *use* your application and concentrating more on how to *benefit* from your application. The most intuitive interface is one that the user isn't even aware of—an "invisible interface." The interface is so fluid that the user and the application become one.
- **Usability should be efficient.** The user should be able to get from point A to point B as efficiently as possible. Every interaction you have with your software should be directed toward the goal you are trying to achieve. As developers, we need to hide all of the processes that occur behind the scenes as much as possible. Users should be presented with only what they need to know, not everything that is actually occurring at a particular moment.
- **Usability should be pleasurable.** The design of your application should be pleasing and engaging. This means understanding good design principles, as well as knowing how to handle user input in an elegant way.
- **Usability must be effective in achieving a user's end goals.** In the end, if your application is intuitive, efficient, and pleasurable, but doesn't get the user to complete the task at hand, you've failed. While keeping in mind the three previous points, ultimately, you need to make sure that the darned thing actually does submit a customer order, print a patient's medical records, or whatever it was originally designed to do!

So, is that it? Well, yes and no. For the purposes of our book, we wanted to keep the discussion of usability *itself* as concise as possible. Instead, we're going to devote the majority of our pages (Part Two) to showing you how we solve specific usability problems with Flash. Think of this book as an analysis of usability case studies, rather than an A-to-Z primer on usability. After digging into these pages, you should feel comfortable with how to solve specific usability issues and how to approach new ones down the road. In Part Three, we'll focus on how to plan for usability in the design process and present a final application that integrates all the solutions.

At the same time, we don't want to dismiss the more theoretical study of usability. A host of great books, websites, and people are dedicated to this ever-evolving subject and many equally important related topics, such as user research, goal-oriented design, and task analysis. We've provided an appendix to this book to point you to other resources if you'd like to dig further into usability theory.

Building for usability

Usability engineering is not just about good interface design. While the visual appeal of a web application is important, the underlying structure and programming can equally contribute to "good usability." Usability doesn't just begin and end with the user. A usable website that's built poorly won't survive the kinds of revisions and feature additions that are inherent to all living web applications. It's akin to a magnificent-looking house that's being eaten through by termites. Eventually, it will fall apart and have to be redone. This is the part that is left out of most books about usability.

In the solutions presented in this book, we won't just talk about creating Flash applications with user-friendly design. We'll also introduce different techniques for designing your code, letting you better integrate the discussed solutions into your own Flash projects. After all, the easier it is to make your applications more usable, the more likely you'll strive to achieve it!

Who is this book for?

We hope you've gotten the idea that this is not your prototypical usability book. As we mentioned, we're not going to dig that deeply into theory. Likewise, this book isn't your usual "how-to" Flash book. We're not going to start from the ground up. We will lay out each usability solution in detail, but assume you already have a good familiarity with Flash. These solutions are geared toward Flash developers who are comfortable with the Flash development environment. You should know what buttons and movie clips are, and the differences between, say, frame-based actions and object-based actions.

To get the most out of this book, you should have a basic knowledge of ActionScript 2.0 (AS2) and a solid understanding of object-oriented programming (OOP). The solutions in this book will rely heavily on using AS2 classes, and we will look at some key concepts in OOP to implement these solutions. If you are relatively new to Flash or aren't familiar with AS2, that's still OK! There will be plenty of take-aways from each chapter that don't necessarily have to do with the development process. We recommend that you read this book alongside any good AS2 programming book (see this book's appendix). Rather than wrapping them up into rigid components, we've made our solutions fully exposed to you and available for download in their entirety from this book's page at `www.friendsofed.com` (just search for this book in the menu on the front page, and then go from there). We want you to be able to take what we've done and enhance it in your own projects!

This book is the ultimate synthesis of a usability discussion and programming guide. We'll talk about how Flash can best enhance the usability of your applications. We'll also give you our perspective on the best way to develop these solutions, from designing the interface all the way down to how to structure your ActionScript code.

So, while this book is geared toward more experienced Flashers who are well acquainted with AS2, these solutions offer something for Flash developers at any level looking to improve the usability of their projects.

A note about our approach

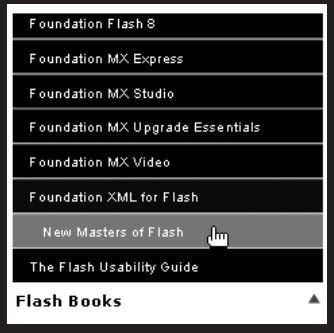
Before we get started, we'd like to stress one very important point to you. As anyone who has developed applications in a programming language knows, programming is an iterative process. There are always multiple ways of approaching a solution to a problem. The solutions in this book are no different.

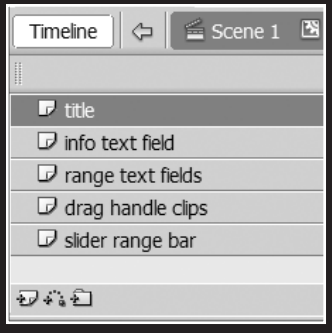
None of the coding solutions in this book should be taken "as is." We highly encourage you to download our source code; examine our examples; and find ways to improve on them to make them more scalable, reusable, and maintainable. In fact, we're fully aware there are more optimal implementations of the solutions we've provided in this book. However, in some cases, we've held back from a "better" approach to a problem because we introduce a concept inherent to the better approach in a later chapter. In other instances, we've held back in order to simplify the method for discussion purposes. Our goal is that, after reading the book, you'll be equipped with a few different kinds of strategies you can take in ActionScript programming as it relates to usability.

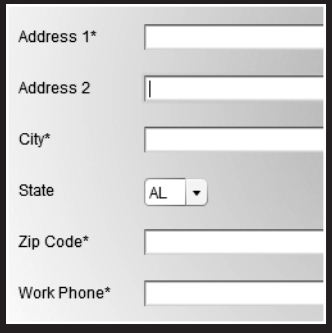
Be sure to check the page for this book on the friends of ED website (`www.friendsofed.com`)—we will post revisions of our code examples as we (or maybe you) find ways of improving them!

PART ONE INTRODUCING FLASH USABILITY

1 FLASH: THEN, NOW, LATER







To get a better understanding of where Macromedia Flash stands today, it's important to look at its beginnings. Only after examining why many Flash-based web applications suffered from poor usability design in the past can we apply the lessons learned from those mistakes. It's up to us, the designers and developers, to figure out how to get around these issues and use the powerful flexibility of Flash to aid—rather than obstruct—our users.

In this chapter, we explore Flash's history and lay out some of the advantages it has over traditional web media. We then discuss the role it should play in the web community as a truly powerful tool for usability. We'll accomplish this by covering the following topics:

- The brief, turbulent history of Flash
- How the Flash environment changed with MX 2004 and ActionScript 2.0
- Flash's advantages (and disadvantages) compared to HTML with Flash
- Flash versus Ajax, a competing technology
- Breaking the Flash usability stigma
- The increasing importance of usability on the Web

The brief, turbulent history of Flash

Today, Macromedia Flash is a robust piece of software, with its own application framework and object-oriented programming (OOP) language. With Flash's remoting and XML parsing capabilities, creating truly dynamic, sophisticated software in Flash is a reality. But it wasn't always this way.

When it began as FutureSplash Animator, Flash was just an animation tool with modest drawing capabilities. When Macromedia bought FutureSplash in 1996 and built upon its authoring tool and plug-in, the web design revolution broke loose in both good and bad ways.

Many interactive designers began exploiting the early versions of Flash, creating magnificent, dynamic websites that juxtaposed the otherwise mundane landscape of the Web. Beauty and aesthetics can take you only so far, though. Companies in the mid-to-late 1990s clamored for their own lengthy Flash intros until they started realizing that users were turning away from all this glitz and glamour. For users, the initial excitement of seeing a new kind of technology hit the Web quickly dissipated. Long intro animations, fancy page transitions, and complex navigation metaphors kept users from seeking the basic information they wanted from their web experience. The added showiness of web design meant users had to wait longer, search harder, and become more vigilant. As soon as how things *looked* became more important than how things *worked*, many users' patience began to wear thin.

Flash has since become synonymous with "flashy, engaging sites" that lack true usability. However, it's an unfair assumption, as it has little to do with what kinds of applications Flash's tools can produce and everything to do with how Flash developers decided to use

these tools. When Macromedia hired Jakob Nielsen (one of the world's most well-known web usability experts), it became clear that the company wanted Flash to garner the same attention for building usable web applications as it had for merely creating fancy interfaces.

Shortly thereafter, Flash software noticeably became more like HTML. As you probably know, Flash text fields can render basic HTML tags like `<b>`, `<i>`, `<img>`, `<a>`, and others. With the release of Flash MX, a set of form components appeared on the scene that mimics the behavior of many traditional HTML form widgets (e.g., select boxes, text areas, and radio buttons). It seems as if the Flash development team figured that in order to compete with the mainstream popularity of HTML, they had to produce components that look and feel like HTML with the special enhancements that Flash can offer.

> *A complete list of supported HTML tags in Flash can be found at* `http://livedocs.macromedia.com/flash/mx2004/main_7_2/wwhelp/wwhimpl/common/html/wwhelp.htm?context=Flash_MX_2004&file=00001040.html`.

While the team had the right idea, we'd like to stress that just because Flash can "act" like HTML, that doesn't mean it has to. Flash offers an out-of-the-box select component, but we shouldn't immediately resort to using it any time we want users to select from a group of items. There may be more intuitive, pleasurable, and ultimately usable ways of solving this problem, as you'll see in our Chapter 6 example.

Flash MX 2004 and the release of ActionScript 2.0

Our perception of Flash has evolved from its early days as strictly a design tool that can create amazing visual effects, to an all-purpose web-authoring tool equally capable of building sophisticated software and crafting stunning visuals. Flash is now synonymous with the development of rich Internet applications (RIAs; more on this a bit later in the chapter), and since the release of Flash MX 2004, Flash developers have been introduced to a new underlying programming language: **ActionScript 2.0** (**AS2** for short).

Sure, by Flash 5, many of the interactive capabilities of the software that we use today were already at our disposal. Many beautifully done pieces of Flash on the Web date back a few years. Even a few versions ago, we had about the same control over Flash's built-in objects as we do now. However, the transition from ActionScript to AS2 (and specifically, the transition to a full-fledged OOP language) was what vaulted Flash into a platform that could compete with—and outperform—traditional HTML-based applications, both from an interface and functional design perspective (as described in the upcoming section, "The advantages (and disadvantages) of Flash over HTML").

New features introduced by ActionScript 2.0

AS2 lets you script code outside of the Flash environment in ActionScript files (text files marked with the `.as` extension). Though you could do this in ActionScript 1.0 by placing code into external files using the `#include` compiler directive, with AS2, you also have the ability to build code with some fundamental object-oriented techniques such as the following:

- Object-oriented architectural structures such as packages and classes
- Object typing
- Class protection (public and private keywords)
- Object-oriented constructs such as interfaces and inheritance

The inclusion of these new features allows you to reuse behavior, maintain your code more easily, and scale up functionality far more rapidly than before. These new features also help assist other developers understand the intent of your code.

Usability benefits of ActionScript 2.0

As the saying goes, there's more than one way to skin a cat. In programming, you have many ways to produce the same end-user functionality. AS2 offers you far more elegant ways of developing your code to achieve the same result than ever before. In addition, it eliminates the old, hard-to-manage techniques that Flash developers employed in the past, like creating movie clips offstage as storage depots for ActionScript code, or using timeline frames to manage states.

Because of these new advantages, you can now start thinking about developing usability solutions that you can easily adapt to new applications. The more modular you make your code, the less time you'll have to spend in the future building out the same kinds of functionality over and over again! Instead, you can spend more time tweaking the behavior or design to accomplish your usability goals. This book will provide a basic open code base for you to use, modify, and enhance in your Flash usability development endeavors.

The advantages (and disadvantages) of Flash over HTML

You should know that we don't mean to condone Flash as the be-all, end-all platform for every usable web application. Every kind of web project brings with it a new set of questions regarding how it's best delivered to an audience. Sometimes, Flash isn't the answer. There are times when a combination of Flash and traditional HTML/CSS/JavaScript provides the optimal solution. There are other times when Flash may not be part of the answer at all.

For example, pure Flash may not be the best route if your audience is primarily using older browser versions or lower-bandwidth connections. The good news is that the penetration

of Flash Player is tremendous. Recent statistics have shown that almost 98 percent of Internet-enabled desktops have a version of Flash Player installed.[1] This level of market penetration is greater than that of many other commonly used programs such as Adobe Acrobat Reader, Java, and Windows Media Player. Also, as computer users continue to migrate toward high-speed Internet connections, bandwidth issues will become less and less significant over time.

Flash also may not be the best medium if your web application needs to be scannable to search engines. If your site's content must be easily found by search engines, you may want to think about using traditional HTML programming for your site (or, perhaps, providing an HTML version of a Flash site that can then redirect itself to the Flash version).

Of course, this is a book advocating Flash usability, so we should point out that a great many traditional Flash usability issues can be resolved with a little creativity. For instance, we'll show you how to enable your browser's Back button for Flash sites (in Chapter 12), as well as how to create effective full-browser-width liquid layouts in Flash (in Chapter 13).

Ultimately, we would like to convince you that there are certain inherent advantages Flash brings to the table over traditional HTML design, regardless of the kind of project you're working on. As we discuss in this section, these advantages include Flash's flexibility, its excellent cross-browser and cross-platform compliance, its asynchronous processing and state management capabilities, and its various design features that help developers achieve elegant visual effects. Moreover, with each version of Flash that appears on the scene, the advantages of HTML diminish even further.

Flexibility

Of course, you know about Flash's flexibility by now. Flash has always been the delinquent, trouble-seeking child of the web development family, unbound from the web constraints imposed on HTML. But while Flash's almost boundless design flexibility can allow for annoyingly long-loading intros, obtrusive ad blasters, and confusing site layout, it can also enable us to build tools and widgets in more beautiful and elegant ways than traditional HTML programmers could ever hope for.

In HTML, there's really only a finite set of tools at your disposal. If you want to build in any sort of interactivity, you typically create a `<form />` block with form elements that we've all used repeatedly by now: text fields, radio buttons, drop-down lists, and check boxes. Any kind of usability design we create with HTML interactivity usually translates to decorating these form elements, but how you interact with them remains largely the same.

In Flash, you're presented with a blank canvas. Want to build an online store? You don't necessarily need to have drop-downs to select the quantities of an item you want. Instead, you can decide that a slide meter will do a more efficient job. Or, you might allow a user to drag and drop items into a basket area in your store. These choices, along with the drop-down design, are all possible options in Flash, while not all are necessarily feasible in HTML.

[1] See `www.macromedia.com/software/flash/survey`.

This isn't to say that you can't build some pretty clever and beautiful applications in HTML. In fact, you could probably build an on-the-fly calculator using JavaScript events and something similar to a slide meter using HTML and JavaScript or a third-party source. However, such an element isn't going to be nearly as easy and reusable as it would be if you built it in Flash. Flash is simply a better tool for building innovative solutions that aren't going to be bound to the traditional constructs HTML brings with it.

Cross-browser and cross-platform compliance

Design once and you're done! Flash has the distinct advantage of not having to rely on how a browser interprets its code to render. HTML and CSS are certainly making progress in developing a series of web standards that most browsers are adopting, but there are still many years of work ahead before all major browsers fully adhere to strict XHTML/CSS standards.

From a development perspective, browser dependence can wreak havoc on your design goals and timelines. You have to consider not only what your target browsers are, but also how your application should look and feel on browsers that don't fully support your code. Also, it forces you to waste time considering whether a particular bug is really a bug on your end or an incorrect interpretation of your code on the browser's end. If you've had experience with browser compatibility issues, you're probably familiar with **code forking**, or building multiple sets of code to achieve the same desired functionality and appearance on different browsers.

With Flash, you have the luxury of knowing that the resulting application will function and look just about the same on all browsers and operating platforms, as long as users have the latest Flash plug-in installed. In this way, Flash is a lifesaver when it comes to having to comply with users' many different browser types and versions.

Asynchronous processing and state management

HTML applications are typically synchronous processes. When you fill out an HTML form, it's only when you click the Submit button that all your information is sent. Moreover, while the form submits, there's little else you can do but twiddle your thumbs and wait for the screen to refresh. When you want to complete a task in an HTML application, you're generally led through a linear workflow process that adheres to whatever the underlying data model is. As users, we spend a significant portion of time *waiting* for things to happen—time that could be better spent doing other things.

AS2 has the unique ability to manage the state of the application on the client. This allows you even greater flexibility when it comes to creating seamless Flash applications. While JavaScript can be used to manage an application's state to some degree, it's certainly not a common practice. It's far more efficient to handle application state on the server side, and code maintainability becomes a bigger issue the more you intertwine JavaScript with HTML.

Flash offers a far more robust set of tools that allow for asynchronous activity. If you ask Flash to, say, give you information on three new products in a store's inventory, Flash can

spit back the information as the items are processed. In HTML, your options are to either wait for all three processes to return or ask for one piece of information at a time.

Things are rapidly changing, however. The benefits of asynchronous processing in Flash are now being ported over to the HTML world. For example, you may be familiar with emerging web technologies such as Asynchronous JavaScript and XML (Ajax), which can produce rich, seamless web applications similar to those created in Flash. (We'll talk more about Ajax shortly.)

Robust design capabilities

By now, we've all become accustomed to the typical effects interlaced within most mainstream Flash applications. We can make objects fade out and fade in. We can make items grow, shrink, bounce, speed up, slow down, drag, hover, stick, slide—you get the idea.

Just as you can use effects to create captivating visual presentations in Flash, you can also use them to enhance the user-friendliness of your applications. Good usability design is often a matter of deciding when these effects are appropriate and how best to create them. Subtle, elegant visual changes to a button rollover can be pleasing to the eye. The way in which a text window responds as you scroll through it can be made to look jerky, or smooth and forgiving. With Flash, you have a laundry list of available effects that you can implement to provide that extra "wow" factor and seamless design when a user interacts with your applications.

Flash can achieve these elegant effects better than HTML-based technologies because, first and foremost, Flash is a design and animation tool. It's very easy to increase frame rates to give Flash movies a smoother look and feel. It's also easy to change the way objects move or transition from one visual state to another in ActionScript because Flash objects inherently contain those properties.

There isn't a really good parallel to Flash design in the HTML world. A combination of HTML, JavaScript, and CSS can partially get us there. But these languages aren't suited for creating elegant tweening techniques. We can create some elegant visual designs with CSS, but concepts like tweening and animation aren't a native component to these languages.

Flash 8 now comes with an even more robust framework to enhance the visual appeal of your projects. Here are just a few key improvements:

- Bitmap filters let you add effects to your objects that once were achievable only by importing images created in a graphics program like Adobe Photoshop. Filters like drop shadows, glows, blurs, and bevels are available at the click of a button!
- Text rendering capabilities far superior to those in any previous release of Flash. Flash 8 provides customizations for anti-aliased text so that it appears much sharper and is optimized for animation and readability.
- The Custom Easing feature, which makes it much simpler to tween objects in Flash in more complex mathematical ways than ever before. You can adjust tweens with a graphical interface rather than relying on complex math calculations.