PART I

Planning Your Project and Setting Up Dreamweaver

Although it might *be more exhilarating to start designing web pages on the first day of your project, experience has taught us that a successful web development project starts with some strategy. First you want to take a step back and look at the big picture. What is your site or application going to do? Who's going to use it? How does it have to work? How should it be organized?*

Next, to make the most of Dreamweaver, take a little time to understand how the software creates, interprets, and displays web pages. Get the software set up in a way that suits your workflow, and then finally start developing your site.

Even then, before you get into the process of building actual pages, you're going to need to spend some time setting up your site in Dreamweaver, and creating the shared templates, library items, and other assets that will enable a rapid, efficient development process.

CHAPTER 1

Planning and Preparing for a Web Project

Dreamweaver and Fireworks can help your project run more smoothly whether you are building a website from scratch all by yourself for yourself, collaborating with a multidisciplinary team to deliver a web application for a client, or anything in between. Regardless of the scope of your project, take some time in advance to think through the *architecture* of your site (fundamentally, its site map), develop a look-and-feel (graphic design and interface); and gather the content. When this preliminary work is done, you can plunge into the development, staging, launching, and maintenance of the site.

This chapter assists you in sorting through the elements of your preferred methodology before you get down to brass tacks with Dreamweaver and Fireworks. Topics addressed in this chapter include:

- **When to use Dreamweaver and Fireworks**
- **Nailing down your process**
- **Gathering requirements for your web project**
- **Designing the project's information architecture**

When to Use Dreamweaver and Fireworks

When we're grandparents, the little kids are going to roll their eyes whenever we start reminiscing about "the dawn of the Web" when we had to hand-code our websites. We'll say, "That's right, kids, we typed all those little angle brackets into text processors with no validation or nothing! We didn't have these fancy-schmancy what-you-see-is-what-you-get, self-correcting, self-updating, automated whatchamacallits." And they'll be right to roll their eyes. Who cares about how these things used to be done? It was ridiculous that we ever had to work that way. "Why, if we moved one page, we had to find all the links referring to that page and then manually change them. Why, one time...."

OK, there we go again. Anyway, the point is, Dreamweaver is a really cool way to keep an entire web project in the palm of your hand (or at least on the hard drive of your laptop), and Fireworks is a unique tool expressly designed for the development and optimization of web graphics. Dreamweaver goes beyond enabling you to design web pages visually—it helps you manage your entire site from the top down. Best of all, it enables a team of people to work together on the same project without accidentally munging each other's work.

> *Munging* means over-writing or otherwise erasing or corrupting an existing file. Use this around your techy-er colleagues and they'll give you a jot more respect.

If you are working with collaborators, chances are not everyone is going to do their work in Dreamweaver. That's OK. Dreamweaver produces clean code that even the most hardcore code jockey can't complain about. From the point of view of programmers and technical architects, you're working on the front end of the site, or the presentation layer (not to be confused with other kinds of layers, which we'll get to later). Sure, if you've got UltraDev, you may be delving into all kinds of database calls and scripted routines, but chances are you'll be handing off your front-end design to someone doing the back-end work somewhere along the way.

With Dreamweaver, though, you can keep your part of the project all in one place, and there's no reason not to do all of your work inside the application (at least after you're done doodling on cocktail napkins).

Fireworks is where you'll develop and refine the graphical look-and-feel for the site—the site's logo, the graphical elements, the navigational hoo-ha's, and so on. The entire site won't live inside Fireworks the way it will in Dreamweaver, but because they're both Macromedia products designed to work together, any graphics you develop or import into Fireworks will flow easily into your site templates and pages over on the Dreamweaver side.

WHEN TO USE FLASH AND SHOCKWAVE

Other web-oriented Macromedia products that play well with Dreamweaver and Fireworks include Flash and Shockwave. Both of these products are descendents of Macromedia Director, an application used to design interactive, well, applications. Chances are, last time you stuck a CD in your drive and watched a little promo or clicked on a bulbous shiny set of interface buttons, they were developed in Director. When the Web came along, Macromedia rolled out Shockwave as a way of adapting Director-like material to the vicissitudes of the Web (bandwidth limitations, mainly, as well as the nasty habit of web users who click away when you want them to sit still—something people developing for CD-ROM never had to worry about).

But Flash is every designer's favorite tool for developing interactive movies, animations, and every other kind of beast that slithers, crawls, runs, or flies across your screen. Optimized for streaming over the Web, and widely accepted as a format, Flash is the first choice when you need that level of production values, or when you want your users to be able to, say, play a video game at your website. Artists love Flash too (see `http://www.snarg.net/` for a hypnotic example of what we're talking about).

And clients love Flash too, or at least they usually do when you demo the little bugger for them running on your laptop. They don't necessarily love it quite so much when users decide that the site is too slow or too Flash-y and fail to stick around to register for the great bargain or stock their shopping cart with whatever widgets your client is trying to e-commercially sell.

So, having sung the praises of Flash, let us now warn you to use it sparingly and when it is called for by the project's requirements, and not just because you finished a course on interactive design and need something "rilly kewl" for your portfolio.

Needless to say, Shockwave and Flash are sold separately, but also play well with Dreamweaver. See Chapter 20 for more on such dynamic, animatronic, interactive magic.

Getting Your Process Squared Away

Before you fire up the software and start cranking out web pages, take a step back to sort out your process (ur, methodology). Nowadays, most web design and development projects are collaborative and require a lot of coordination among team members. Yes, if you're running a one-person project or shop, you don't have to answer to anybody, you don't have to use anyone else's lingo, you don't have to adhere to anyone's deadlines, and no one is going to second-guess your work. But even then you're going to have to figure out what to do first, what part of the project depends on other parts being completed first (sometimes referred to, for short, as *dependencies*), and what your timeline and milestones are going to need to be...unless maybe you're building a website for your cat and there's no deadline.

In most situations, you've got a "someone" to answer to, whether it's your boss, your client, or simply your audience. That's right, web design requires you to anticipate and meet your audience's needs; that is, if you expect them to come to your site, use your interactive application, register with your enterprise, or come back again after the first visit. As the bard once said, "You're gonna have to serve somebody." Furthermore, in most commercial projects, you're going to have to collaborate with somebody, or with a whole team of somebodies. There might be a branding expert, a writer (perhaps called a content developer), some developers (technical architects, front-end scripters, back-end coders, middleware specialists, and so on), and possibly a project manager. Oh, yes, and a visual designer or graphic designer. But maybe that's you?

You may be working with people who cut their teeth in the field of professional services, interactive or advertising agencies, publishing, and software development. You're going to discover that everyone has a different name for the same thing (is it a *storyboard* or a *wireframe*, a *site map* or *thumbnail series*, *use cases* or *process flows*?), and most people see the project revolving around their discipline. In any collaborative project, some time—at least an hour—should be spent up front hashing out the division of labor, the dependencies (such as, "I can't develop the content inventory until you finish the site map"), the points of handoff or turnover, and the milestones and deliverables expected by the client (even if the client is just your boss).

For more ideas and discussion about various web-development methodologies (and there are a number of equally valid approaches), check out the author-created website for this book, at `http://dreamweaversavvy.com/`.

Gathering Requirements

How are you going to know what to put into your site or application unless you spend some time and effort learning the needs of your website's or application's eventual users? (Consultants call this stage of a project *discovery*—not to be confused with lawyers pawing through your files.) This discovery phase should involve interviewing representatives of every audience type or anyone else with a stake in the usefulness and success of the site. This means not just your boss or client, your client's boss or team members, and other obvious stakeholders, but also, if at all possible, some potential users of the site—often customers, partners, or vendors. Find out what they want. Your client may not always know what their users want as well as they think they do. Also, it makes a killer argument when the client has gotten attached to some horrible idea to be able to say, "But your site's users don't want that. See, here in these interview notes, they say they'd never come back if you had *that* as part of your site."

This leads to the first commandment of web design (perhaps the only commandment, we're not sure).

Know Thy Audience

What if they built a web site and nobody came? They did. And nobody did. It was called the dotcom bubble. Maybe you missed it? Just because you can sell your boss, or your client, or a venture capitalist (VC) on an idea doesn't mean that people are going to come and pay you to keep executing that idea. Understand your audience. Go meet them if possible. Interview them, but also watch them as they work. Study what they like and dislike. Learn as much as you can about usability. If your ultimate product isn't usable, guess what? People won't use it. If your site doesn't meet a need, then no one will need it. This sounds simple, but a lot of VC money went down the drain because people wearing the right shade of blue shirts who knew consultantspeak put together some really hep-looking Powerpoints and 10-page business plans with no revenue model.

OK, that's not really fair. When tulips are the rage, everybody buys tulips. But don't end up like the dotcommers taking their one-way U-Hauls back out of the San Francisco Bay Area even as we type this. Think about the needs and desires of your audience. Understand them. What are they reaching out for? How can you satisfy those needs? Get that straight and the rest of the project will practically take care of itself.

Getting the Information Architecture Right

Information architecture is a $10 word that means how your site's information is organized. What do people see first when they come in the front door? How many levels down is certain information buried? How many clicks does it take to get to crucial information? What is the structure of the navigation? Dreamweaver won't figure any of this out for you. (And Fireworks? Fuggedaboudit). Sure, once you've sorted it out, Dreamweaver is an awesome tool for maintaining the site map, navigation links, and so on. But you have to do the hard thinking first.

ENSURING THAT YOUR SITE IS ACCESSIBLE

One of the most important changes to the specifications for HTML in Version 4.0 is the inclusion of requirements that your site be accessible to people with disabilities. This means that your work can be used and appreciated by the estimated 10 percent of the world's population with some sort of physical disability that makes it difficult to access most websites. The World Wide Web Consortium (W3C) has set up a special organization and website that provides helpful checklists, guidelines, and ideas in support of the Web Accessibility Initiative (WAI). Check out `http://www.w3.org/WAI/` to find out how to insert accessibility into your website relatively painlessly. Another very useful site that serves as both an example of a well designed, accessible website and as a tutorial is the Web Accessibility In Mind (WebAIM) site (`http://www.webaim.org/`).

Fortunately, because Dreamweaver is so flexible and easy to use, you can make mistakes when you start and still correct them later on in the project. That's right, no matter how carefully you gather your requirements, know thy audience, or massage your client, guess what? New requirements will emerge at the 11th hour. Projects you've never heard of will suddenly demand to be integrated into your pristine site map. Entire divisions will be defunded and no longer entitled to that valuable real estate on the home page. Never fear, with a few points and clicks, subsections can be promoted to top-level categories, and entire site areas can be snipped out and placed in limbo. I'd like to see a "real" architect try to rearrange a real building once the contractors are in the house!

Still, just because things are inevitably going to change, that's no excuse for not trying to get it right at first. (That's what our editor told us when we said we didn't need to do an outline for this book. "It's all going to change!" we whined. He was having none of it....) In fact, getting your architecture clarified at the beginning makes it that much easier to *track the changes* as they emerge. Think about it. It's a lot easier to see what's changed if you know what it changed from!

Also, remember those other people on the team (or your cat). They need to know where you're planning on putting stuff. They may be writing scripts and have to know what directory (a.k.a. folder) a certain piece of content is going to live in. They may need to know how many levels down their funky little application is going to be running. Sorting out the site's architecture is step one of designing any substantial web project.

Developing a Site Map (or Thumbnails, or Process Flows)

A great way to get an overview of a website at a glance is to create and maintain a site map. Ultimately, Dreamweaver can generate or show you a site-map type view of your site, but that's once you've actually created all the pages. When you're just getting started, you can literally just draw it by hand or use any illustration software to put it together. A site map looks something like a family tree (except without the male and female figures). Pages are represented as boxes and labeled. You can indicate the name of a page or—if you really want to get into it at this stage—what the page will contain. Navigational links are represented as lines between boxes. Then, like a family tree, child and sibling pages all stem from a parent page, and you can take this organization down as many levels as you like. Some people make the boxes smaller as they go down in levels. Often the third level of links is represented as a simple text list. The level of detail depends on your own needs or those of your client.

Figure 1.1 shows a scrawny little site map we just whipped up for an imaginary vanity site for a cat named Fraidy.

Site maps are also sometimes referred to as *thumbnails* because they represent each page of the site (or each major page) as a tiny thumbnail version of the actual page. (If your bosses are fans of *Spinal Tap,* they may wonder if you are actually going to deliver teeny-Stonehenge-sized pages. Reassure them that this will not be so.)

When you are developing an interactive application (as opposed to a series of static, linked pages), the site map might be referred to as a *process flow*, because it shows how the user might flow through the various pages or screens to accomplish some task. (For example, let's say you developed a new search engine for a site, and they already had a site map. You could create a process-flow diagram showing how users can go from the basic search box to the advanced search page or to the search tips, and then ultimately to the search results.)

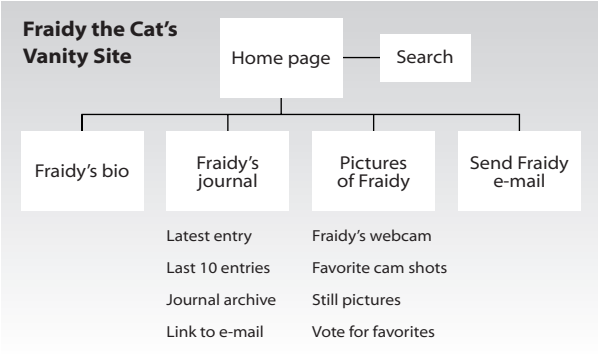

Figure 1.1

Fraidy's site has a main page with a search link (off to the side as part of the persistent navigation at the site) and four subpages reached from the home page's side navigation. Two of those subpages have subsubpages of their own.

You can create whatever codes or symbols you'd like to indicate different types of pages. Use dotted lines for dynamic pages (pages that are created from a database), or use rounded edges for new pages. Have fun with it. This is information design without that tedious "making it work" part.

Creating Wireframes (or Storyboards, or Process Flows)

At least as useful as a site map is a set of *wireframes* for your project. Wireframes are analogous to what people in the movie, television, and advertising business call *storyboards*. Except storyboards usually show pictures of people saying stuff and indicate a flow of action. Wireframes indicate a desired progression through the website's pages. You can think of them as being drilled down to one further level of detail from the site map.

> Wireframes are usually done without color. They don't show the actual design of the pages, and in fact, they are presented denuded of as many design elements as possible so that your client doesn't get the idea that you are presenting the actual, final design. Instead, you are just showing the functional elements and content areas for each page and trying to get some signoff on that so that you can go ahead and develop a real look-and-feel.

Each page in your site map can be represented as a wireframe, usually a full page showing roughly where the navigation, content areas, and any interactive elements (such as forms or image maps) will go. The exact placement of these elements is not the point. The point is that the wireframes indicate a list of the elements that will eventually populate each page. They should also indicate where any of the links will lead so that anyone reviewing the wireframes can easily see how the user would step through the site. This is much easier to do with barebones wireframes than with a complete site mockup—if you use one of these, you risk having to listen to the marketing department representative ask, "Why did you put that purple color under the logo?"

Figure 1.2 shows a mocked up wireframe page showing one step in a content-management process (gleefully adapted from the brilliant work of designer and information architect Dan Shearer of New York, New York).

Ready, Set, Rumble!

This chapter explained some of the planning and organizational work you'll need to do before you launch a serious web-development project. This includes determining when it's appropriate to use Dreamweaver, Fireworks, and other Macromedia products during this development process; hashing out a process or methodology for developing your site or application; figuring out how to collaborate with other team members (if necessary); gathering requirements for your site; and developing an information architecture.

This information is useful no matter what tools you were going to use to assist you as a web designer. Don't ignore these steps because they involve thought processes and decisions that your software applications can't do for you. Once you know how you're going to design your project, what you're going to do it with, how and when you are going to use your tools, who the product is for, and how the information at the site will be organized, you're ready to start cranking away in Dreamweaver and Fireworks. The next chapter will take you through the elements of a web page and a website and will even show you some actual Dreamweaver screens! Remember, have fun.

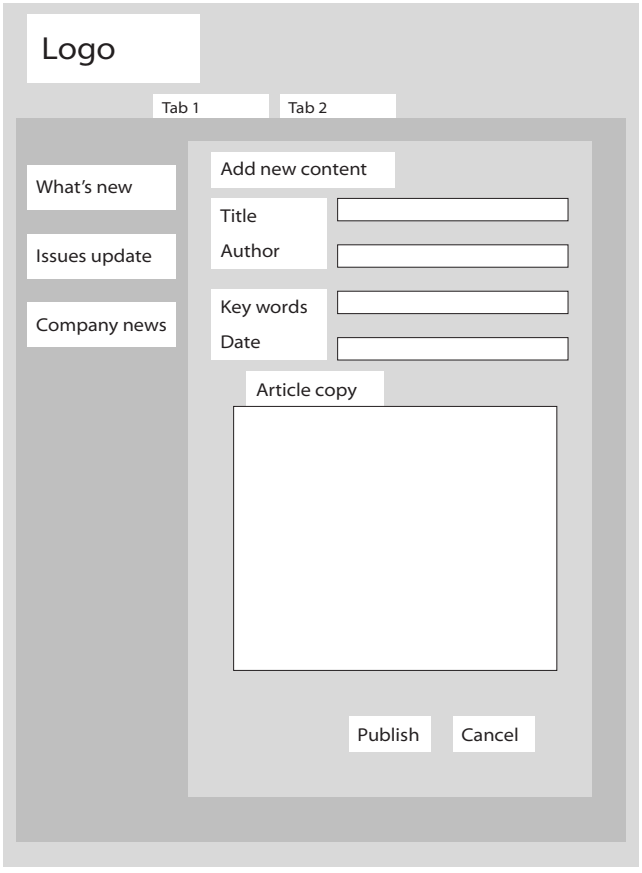

Figure 1.2
Without looking like an actual web page, this wireframe shows that the interface will include a logo; tabs at the top; options along the left side; text boxes; and buttons for publishing the content or canceling the process.