1

The Planning and Development of Your Site

In the past few years, the design community has seen an explosion of sites powered by cascading style sheets (CSS). Highly visible brands such as Fast Company, ESPN.com, PGA, and Blogger have all adopted CSS for the layout of their sites, delivering their compelling content through this excellent Web technology. Their pages have become lighter and more accessible, while a few style sheet files provide them with global control over the user interface of their entire site. The potential of CSS has been well established by these mainstream sites, and the technology (which languished since its introduction in 1996) is quickly becoming the *de facto* means by which a site's design is built.

However, while CSS has been elevated to near-buzzword status, it's important to remember that style sheets are simply a tool to be used in the overall design and development of your Web site. Granted, that tool is an incredibly powerful one, but it can only *facilitate* high-powered, professional-looking Web sites. Although using style sheets can afford you an unprecedented level of control over your site's design, no technology is a silver bullet. Despite what technology evangelists might tell you, adopting CSS won't inherently make your site more usable, your design more compelling, or your breath more wintergreen-fresh.

So, if we put aside the buzz for a moment, we can see that although CSS is an incredibly important aspect of a Web site's development, it should be viewed in the context of that site's entire lifecycle. In this chapter, we'll discuss the following topics:

- Understanding your project's scope
- Establishing the goals for your project
- The fundamentals of information architecture
- How to begin your site's design

Establish the Scope

If we were asked to build a house, there are a few questions we'd want answered before agreeing to do the work. How large will the house be? How many rooms? What kind of budget is allocated for this project? All of these questions are meant to establish the *scope* of the construction project. It's a means of gathering information about the project, so that you can more intelligently assess its needs. By establishing the scope of the project, we can better understand exactly how involved the project is, how long it will take to complete, and how much it will cost — all items that are integral to any formal contract.

If you can't tell by now, our construction metaphors aren't exactly our strongest point. But while we might have been snoozing through those episodes of *This Old House,* the parallels to the start of a Web project are uncanny. Before firing up Photoshop or slinging one line of code, you and your client should work together to produce a well-reasoned *scope statement.* This aptly named document not only determines what work will be performed throughout the duration of the project, but also implicitly defines what work is *outside* the scope of the project. This is an incredibly important point. When the deadlines are tight and the expectations high, knowing exactly what is expected of you throughout the course of the project will keep your budget in check, and both you and your client focused.

Most frequently, the scope statement contains the following information:

- **Strategy.** This contains some information about the goals and business needs behind the project. Is the site you're designing supposed to increase advertising revenue, or shore up readership numbers? Is the project supposed to increase a site's accessibility, or its search engine ranking?
- **Deliverables.** These are the products that will be created over the course of the project. If your project involves some level of site planning, will you be building a site map, or providing wireframes? When the project is finished, will you provide the client with all the Photoshop files used in your designs?
- **Assumptions.** This is all of the conditions and constraints that were used to establish the scope and upon which all other timelines and goals are founded. Should any of these initial assumptions change, the scope of the project should be revised accordingly. For example, if initial client meetings uncover only 30 pages on the site that need to be built, then the landscape (and your estimated budget) can change rather drastically when the client informs you two weeks into the project that it has another 300 pages it would like you to redesign.
- **Scope management.** No matter how extensively you document your assumptions and timelines, someone will invariably request changes during the course of your project. Whether it's Murphy's Law or bad karma, it doesn't matter — you and your client need to acknowledge that this change will likely occur and mutually agree upon some process for handling it. Some developers will write a separate document detailing how these changes are managed, and how the client approves the resulting changes in schedule and budget.

This isn't an exhaustive list, nor should it be seen as definitive. As we'll no doubt harp at you throughout the remainder of this book, each project has its own needs, its own goals. Because of this, each scope statement should be tailored to reflect this.

However, it is important to remember that it is the project scope that forms the basis for whatever contract you and your client establish. Because of this, the gathering of the requirements should be a highly collaborative process. You and your client should work closely together to flesh out exactly what information should go into the scope statement. Otherwise, if you and your client have differing expectations as to what work falls under the auspices of the project scope, then problems may arise as deadlines approach.

Determining Roles and Responsibilities

No person is an island, and the same is especially true of Web contractors. Every project requires some level of coordination with additional people, whether they are members of your team, or stakeholders from the client's company. As such, a successful project becomes less about providing a set of deliverables at a specific time, and more about managing the different members of the project team. Clearly communicating the expectations placed on every member of the project will help ensure that deadlines are met on time, on budget, and above the client's expectations.

At some point in your professional career, you'll be staffed on a project where its requirements exceed your abilities as an individual. This isn't something to dread, however. Rather, discuss with your client that their project's scope requires additional resources to meet the deadlines, so that you can plan your budget accordingly.

In either scenario, it might be helpful to sketch out a table that outlines not only the various roles distributed throughout your team, but their responsibilities as well. In the following table, we've banged out a rough sketch of what the average project team might look like. Even if you're the sole resource working on a project, this exercise can be quite helpful. Acting in multiple roles can be quite a juggling act throughout the duration of a contract, and a table such as this can help you identify exactly what is required of you, and with whom you'll need to interact.

Role	Responsibilities	Deliverables
Project manager	The overall traffic manager, overseeing the project's progress from gathering requirements to delivery. Works with the client sponsor to define the scope and requirements of the project.	Scope statement (co-author); timelines.
Project sponsor	This primary point-of-contact at the client company defines business requirements, and provides sign-off at various stages of the project.	Scope statement (co-author); creative brief; any additional materials deemed necessary for gathering of requirements.
Information architect	Develops the site's infrastructure and establishes interface guidelines that are both intuitive and scalable.	Site map; wireframes.
Web designer	Establishes the site's visual design, or "look-and-feel."	Graphic mockups; static HTML templates (as well as necessary CSS/image assets); style guide.
Web developer	Responsible for any server-side programming when building a dynamic, database-driven Web site.	Functional specification; application server installation/configuration.
Database developer	Builds the database that will house the Web site's content, and drive other dynamic aspects of the site.	Data model; database installation/configuration.

In the first column, we've identified the different roles that are distributed throughout our project team. From there, we've outlined a brief overview of the expectations that will be placed upon them over the course of the project lifecycle. While it's nearly impossible to accurately capture everything that a Web designer is responsible for in fewer than 12 words, this brief synopsis should at least convey the most important aspects of the role. Additionally, we've outlined the specific items each member will deliver over the course of the project. These deliverables should be drawn directly from the scope of the project and matched up to the individual best equipped to deliver them.

Also, you may notice that we've included a "project sponsor" on this table, which is a member of the client organization that shepherds the project from inception to completion. While perhaps not as integrated into the day-to-day execution of project goals and requirements as the rest of your team, this person exercises veto power in critical decisions, defines the business needs that drive the project's goals, and ultimately facilitates communication between your team and other stakeholders from the client company. As a result, this person is integral to the success of your project. Any additional information for which they are responsible should be tracked closely, as though they were a part of your project team.

Budgeting Time and Expectations

After you've assessed the different needs of your project, you will be in a better position to determine how long it will take, and exactly how much it will cost. "Time is money," of course, but that's rarely more true than within the bounds of a consulting engagement. Granted, it's a fine thing to discuss strategy, scope, and staffing, but it's your project's budget that will determine whether or not any of those things actually come to fruition. With a properly established budget, you can begin to add people and technical resources to your project plan, as well as any other expenses that your project might require.

Furthermore, the amount of money allocated to a given project can limit the size of your team. Perhaps our project plan requires two developers and one designer but funds exist only for two full-time resources. As a result, something must go: either reduce the number of staff on our project, or reduce the scope of the project to the point at which two people can easily handle it.

Conversely, the budget could affect the quality of our team: if the funds are not available to hire an experienced application developer, then we might need to hire a less-expensive (and perhaps less-seasoned) resource instead. Or, if the budget is especially tight, *we* just might need to pick up that Perl book and start skimming. Therefore, a solid understanding of any budgetary constraints will help us understand the extent to which we'll need to bootstrap our own skills or those of our team members, and how that preparation will affect timelines.

Managing Change

Let's say that we're nervously eyeing the clock two hours from site launch. We're looking at the last few items on our to-do list and are feeling a bit pressed for time before the site's go-live. Just then, our client sponsor strolls up to say that *his* boss wants some holiday e-cards designed, and that they should go live with the newly rebranded site (this has never happened to us, we swear). Just like that, our priorities have been told to shift, but the project's timelines haven't budged an inch.

This is what is known as "scope creep" and is a part of almost every project. At some point, the requirements outlined at the beginning of the contract may need to be updated to reflect some new or updated business requirement. If our project can't adapt to our client's needs, then we're likely working on the

best product they'll never be able to use. So, while this kind of change is expected, it is very important to know how to manage it effectively. No designer wants his or her timelines in a constant state of flux, especially when the budget isn't. How, then, do we manage scope creep within a project? There is no easy answer to this question, but there are a few strategies that might be useful to keep in mind.

Introducing Sign-Off

Once a particular deliverable has been finished and presented to a client, it is usually a good idea to ask the client to formally "sign off" on the work. This sign-off can take the form of an e-mail from the client, minutes from meeting notes, or preferably a physical signed document. No matter the form it takes, it should formally document the client's approval of what has just been delivered. By securing the client's sign-off on a given deliverable, the client confirms that our work meets the requirements that the client set before us. In effect, the client is telling us, "Yes, this is what we asked you to provide for us — let's move on."

Some designers might call this "blazing a paper trail." The rationale frequently is one of offloading accountability onto our client's shoulders: that if any future changes must occur, the fault — and incurred cost — lies not with us as consultants, but upon the client's revised requirements. And, on the face of it, this thinking has a lot of appeal. Whenever possible, we should toe a hard line with the established scope, and ensure that the agreed-upon requirements change as little as possible before the project's completion. Sign-off is one way to help ensure this, enabling us to point to completed work should we ever be asked to undertake time-consuming revisions.

Viewed in a more positive (and somewhat less mercenary) note, sign-off can be a valuable means to increase the level of collaboration between consultant and client. Sign-off provides a scheduled touchpoint for our clients, allowing them to check in on progress made to date. In this, the client almost becomes another member of the project team. Formalizing the approval process integrates the client's decisions into the project lifecycle, and increases the level of interest the client has vested in maintaining the project's momentum.

And on the subject of momentum, sign-off is in itself a valuable device for maintaining a sense of progress from project inception to final delivery. Over the course of a given project, you may find that most deliverables cannot be built unless another has been completed. For example, it isn't possible to begin building HTML templates of our designs if the mockups haven't been finalized — or rather, the template process becomes extremely lengthy and expensive if the design is still undergoing revision. By requiring sign-off on a particular phase of work before the next phase can begin, you can help ensure that your work is delivered on-time and on-budget. That should make parties on both sides of the negotiating table quite happy.

Refer Frequently to the Project Scope

While the scope statement enables us to define the requirements for our project, it also implicitly establishes what is *not* in the agreed-upon scope — a critical point when deadlines are tight and client expectations high. It's important to have established this baseline with your client.

This is where the collaboratively authored aspect of the scope document becomes most important. By working closely with the client at the outset of the project to define its scope, the client has a more concrete understanding of how that scope (and any changes to it) will affect both pricing and timelines. That's not to say that this will mitigate any and all potential scope changes. Rather, it will help facilitate any later reviews of the original proposal and allow both sides of the contract to more intelligently and openly discuss how the new changes will affect schedule and pricing.

Frame the Work Within Your Budget

From this, it's important to remember that the project's budget can be a valuable tool in mitigating scope creep. Just as our scope statement helps us understand what is out of our project's jurisdiction, so, too, can our budget help us mitigate unnecessary changes. If there are insufficient funds for a requested change, then the issue is quickly rendered moot.

Failing that, it's worth discussing with our client just exactly how this change will impact the budget and the project timelines. Frequently, the relationship between additional work and additional time or cost is forgotten in the heat of a fast-paced project. By demonstrating that *X* amount of work will require *Y* additional dollars at *Z* billable hours, we can work with our client to assess exactly how much of a priority the scope change actually is. (We were never especially good at algebra, but please bear with us.)

This might sound as though we're trying to get out of additional work — quite the opposite, in fact (after all, we must pay for those plane tickets to Bali). Rather, it's our responsibility as contractors to help our clients attach a quantitative measure of importance to that work; namely, does the amount of additional time and funds justify the importance of this new project? Weighed in this manner, this latest work can be assessed by our clients against the other parts of the project scope. If the scope change is ultimately decided to be a must-have, then we can work with our client to decide how to proceed — should the timeline and budget for the project be revised, or should some other aspect of the project be foregone to usher in this new task.

Moving Forward

Of course, if the client is willing to alter the scope and the budget to accommodate a vital change, then we need to be equally flexible. Much as we did before beginning the project, we need to establish the scope for the requested change. What kind of work will it entail? How long will it take? How many resources will it require? Once these questions have been answered, we can more accurately estimate exactly *how* this scope creep will affect the project as a whole. If designing those holiday cards will require three days of design and review, then that needs to be communicated to our client. While we might not be able to produce what they're asking for in the requested time, an open and frank discussion about how long this request will take — and how it will therefore impact the larger project — will often follow.

While discussions of pricing and process are no doubt difficult ones to have with your clients, it is extremely important to remain firm on these issues. If it helps, try not to see these discussions as a means to protect the bottom line. Rather, every client will try to test the limits of the project's scope; the more you capitulate to out-of-scope requests on short notice, the more they will anticipate and expect this behavior. This can quickly lead to projects that fly wildly off of the original specification and schedule, which will in turn push back delivery dates. Neither the designer nor the client will be pleased with this result. Therefore, maintaining a firm (but fair) line on these issues will help you meet the project goals — and your client's expectations — successfully.

Constant Communication

Not to sound too much like a greeting card, but communication is quite possibly *the* element that determines a project's success. Conversely, the lack of effective communication can run an otherwise well-planned project aground. As the project manager, you must remain in constant touch with your clients about the status of the project, potential shifts in scope, upcoming delivery dates, and milestones. In short, the more touchpoints you can maintain with your client about how the project is progressing, the better. We've never been on a project that ran off-track because of too much communication. However, we've definitely seen instances where insufficient contact met with disaster.

Designer, Know Thy Goals

There's an implied "All of Them" at the end of this section's heading because any project carries with it a small army of distinct (and, at times, competing) goals. The Rational Unified Process, a software project management methodology (`http://ibm.com/software/awdtools/rup/`), establishes the goals for a project by defining the *Critical Success Factors*, often referred to as CSFs. These factors are something like a laundry list that will help you determine when the project has completed. Some sample CSFs might include:

- Build for-pay subscription newsletter service into site.
- Increase traffic by 40 percent over 6 months.
- Redesign home page to allow for rotation of 728 × 60 banner ads above the company logo.

There's nothing especially surprising here, and it is, in fact, a rather modest list of business requirements that are key to the success of the project.

Of course, things become complicated when we try to establish *whose* factors define a successful site. In other words, who are the project's "stakeholders"? Who can benefit from the project's successful completion? Conversely, who would be affected by a less-than-successful Web site?

While a client might undertake a redesign to gain more space for advertising on a site (and therefore shore up the company's advertising revenue), this requirement could conflict with users that are just trying to find a particular article buried beneath the banner ads. So, while your redesign might meet the established business goals with flying colors, the site's users might consider the project an unmitigated failure.

So, if business and user needs are in competition, exactly to *whom* are we supposed to listen? As much as we'd like to, we can't give you an easy answer to that question. Obviously, we can't treat business and user needs as an "either/or" scenario. Rather, it is our responsibility to perform a rather delicate balancing act between business and user goals, and ensure (somehow) that both are represented in the work we ultimately produce.

Your Client's Goals

If your project is to be of any value to the client, it must advance the client's business objectives. Frequently, our clients are outside of our own industry. Whether the client comes from the print industry, the automotive industry, or has a small business looking to establish an online presence, they often have little experience with the *how* of Web design. After all, that's why they're talking to us. We have been tasked to take their particular business requirements and goals and realize them online. Of course, the lack of industry understanding works both ways. We often have as little experience with our clients' industries as they do with ours.

Therefore, it's important that both sides of the equation get to know each other. When gathering requirements for your project, find out everything you can about the client, and the context in which the client company operates. It's not possible for us to know too much about the client's industry, business, and goals (both short- and long-term). No question is too basic. We must find out as much as we can about the client's industry, potential sales markets, marketing strategies, and competitors. This knowledge will only help us as we plan and execute a project that will meet the client's needs.

Of course, as we're gathering this information, we should explain our own work as thoroughly and clearly as possible. We should tell our client a story about what this Web project might look like, from

beginning to end. We can explain what we'll be building, and what kinds of deliverables we will produce at different stages of the project. But more than explaining *what* we produce, we should explain *why* our projects are structured as they are — we can describe why a site map is important, or why design mockups must be finalized before any HTML can be coded. In doing so, we can demystify the project lifecycle for our clients, and help them better understand the sequence of events that lead them to a successful project end.

Your Audience's Needs

If you were building a house for yourself, you could immediately dive into the planning without taking anyone else's goals into account. Because you're the only person who will be living in your little shack *de résistance*, you can take wild liberties with the structure, layout, and aesthetic of your house. Go on, put the bathroom in the middle of the kitchen — we won't tell anyone, honest. Of course, if you ever have any guests over for dinner, you can bet that you'll get some puzzled glances, and more than a couple questions about what you were thinking.

However, when designing a Web site, our own needs and preferences are the last that we should consider. Rather, we design for others, for our users. If we build a site supplied with world-class content, but the user can't figure out how to navigate beyond the home page, then we've failed not only our users, but in our design as well. A successful, user-centered design can yield high traffic, a flourishing community of satisfied users; an unusable site nets you a high degree of dissatisfaction, the size of which will likely be inversely proportionate to the size of your audience.

Of course, unlike the guests at that ill-fated dinner party, it's a bit more difficult to figure out what your users want. As a result, it's far too easy to leave them out of the equation entirely when we make plans for our sites. Instead, we discuss our pages as a collection of features, areas of functionality, or disparate areas of content. That can easily be a rather cold way of assessing your site — and you can bet that your users will give you the cold shoulder, hurrying off to find a site that helps them achieve their goals, rather than hindering them.

Creating Personas: Putting a Face to Your Audience

So, how do we make a site more usable when we've never met a single one of our users? Given just how virtual our little medium is, our users are often invisible to us. So, instead of thinking of them as a faceless mass of surfers clicking through page after page of our site, we can create *personas* (or user profiles) that give our users a face. Personas are model users who can help you better understand the needs, behavior, and goals of your users. In creating these fictitious profiles, you can better understand and anticipate the behavior patterns of the people who will actually use your site.

Figure 1-1 shows a sample persona.

While Frank is a fabricated user, his usefulness derives from the fact that he is *strategically* fictitious: his biography, aspirations, and professional goals are all drawn from trends sampled from your site's users. By doing so, a persona becomes a valuable guide through the planning, design, and development process. It allows you to put a face to the otherwise faceless people who will be visiting your site, and allows you to avoid the pitfall of basing design decisions on technical or personal biases. Rather than asking yourself how *you* might navigate a certain page, you can ask yourself how your persona might do so. It's a tactic meant to humanize the design process, yes — but ultimately, the quality of your site will improve, as will your users' satisfaction with it.

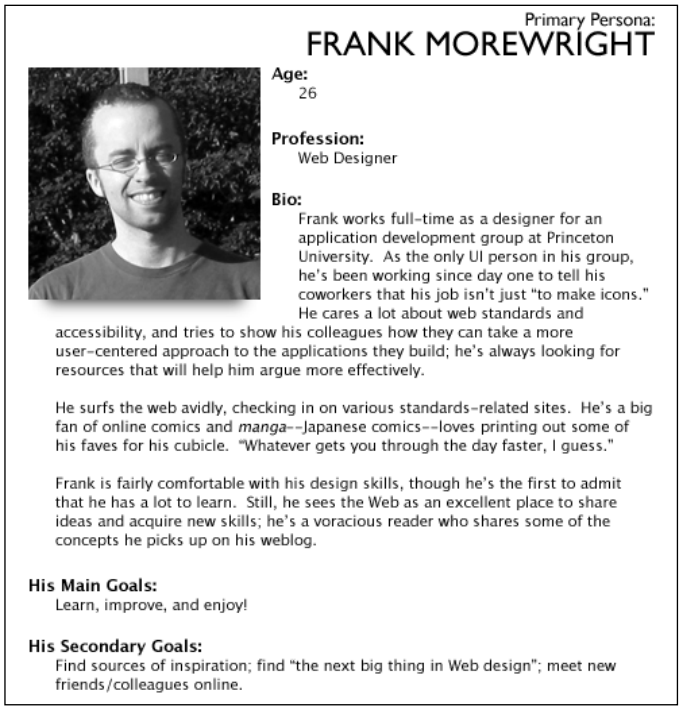


Figure 1-1: A sample persona. (We didn't say it was going to be pretty.)

So, how do we actually create a persona? There are a number of ways to begin this process. The one you pick will ultimately depend on the scope of the project, and the amount of energy you're able to commit to it.

The first (and least "time-expensive") option is to assess what you know of your audience from various internal sources. Examining your site's server logs isn't a bad place to start. These files can give you valuable technical information about your users. At the base of it, this research will yield some important technical demographics: you'll be able to assess what kind of browser landscape exists in your audience and on what operating systems they view the Web. As you'll see in later chapters, each browser has a number of CSS bugs and rendering idiosyncrasies. Knowing what browsers you must support will play a critical role in the development and testing of your site.

Furthermore, you might be able to glean some valuable geographic data as well. As they troll through your site's pages, each visitor will leave their IP address in their wake. From this bit of information, various log analysis tools can tell you from what parts of the world your users originate. Why is this important? If you're building a site for an international audience, your design should be able to speak to people of multiple languages and cultures. For example, will your site's icons convey the same meaning to an American audience as they would to a German one, or even to a user from Singapore? Knowing from *where* your site's audience comes is as important as knowing *what* your audience wants.

> *There are a number of log analysis tools available to you in conducting this research. AW Stats (*`http://awstats.sourceforge.net`*) is a freely available log analyzer that can analyze log formats for such popular Web servers as Apache's httpd and Microsoft's IIS. Webalizer*

(`www.mrunix.net/webalizer/`) is a similar package, but works only with Apache log formats. ShortStat (`www.shauninman.com/mentary/past/shortstat_maintenance.php`) is a PHP application that can track various kinds of user data. These are only three such packages, and there are dozens, if not hundreds, of alternatives available. Each has its own strengths and weaknesses, which should be assessed according to your needs and technical requirements.

In addition to analyzing server logs, you should interview the site's stakeholders. These are the decision makers, the people who drive the direction of not only the site, but also the business behind it. These people will have a strong bead on the site's audience, and ideally have had close contact with them. As such, they can provide valuable insight into your users' needs, and into which areas of the site would be most relevant to them.

Of course, the best method of creating effective personas is setting aside internal statistics and assumptions, and actually *meeting* some of your users — after all, facts and figures can go only so far in identifying just what it is that your users value. At some point, you need to set aside quantitative data for qualitative interviews; there is no substitute for sitting down with people who will (or currently do) use the pages you're designing. Talking with them about their needs and goals not only creates a vital feedback loop for you as the site's designer, but also helps you put real-life anecdotes and experiences behind the design decisions you'll be making. For example, you might poll your users on any of the following points:

- **Technical information.** What kind of browser do they use, and on what kind of computer? How do they use the Web? *Why* do they use the Web?
- **Customer information.** How do they view the site of your company or client? Do they use it? What do they think of the site's competitors?
- **Personal information.** This might include such information as age, gender, and location (for example, urban or rural).
- **Design preferences.** How would they define a "good" site design? While you might not ask them to leap into an art school–esque dissection of a given site's design, you might ask them to tell you some of their favorite sites. Try to find out why those sites are their favorites. Additionally, you could try to uncover what sites they like least, and why.

Of course, this isn't a comprehensive list — the goal of any user sampling is to get as much data as possible that will be helpful to you in your design effort. But rather than just seeing this as a data-mining initiative, think of it as a series of conversations with real people. While demographic information and technical statistics are vital to planning your site's development, collecting anecdotes and quotes from actual people will help to create a more effective persona. Remember that these are ostensibly the individuals that will be visiting your site, and whose needs your design will need to address. When you think of them as *users*, your design suffers; when you think of them as *people*, your site will be all the more successful.

Once you've collected as much data as possible, the analysis begins. By sifting through your notes, trends should emerge. An overwhelming percentage of your users might use the Macintosh version of Internet Explorer; a large minority of your users might be color blind, or suffer from poor vision; a high number might be working mothers, or perhaps teachers looking to acquire professional development credit. More than likely, your audience will be multifaceted, and could contain any or all of the above. But no matter the spectrum that your users cover, it's important to keep these seemingly disparate characteristics in mind as you sit down to write your personas — because just like your "real" users, your persona may not be able to be easily categorized.

In fact, the best place to start working on your persona is probably the narrative, or the persona's biography. This is the meat of the persona, and is where the credibility of this virtual user is established. If you have a gift for embellishment, this isn't a bad time to flex those creative muscles. While you might want to communicate your persona's lack of technical expertise (and, in doing so, remind the developers that your users aren't especially Web-savvy), try to think up a few fictional details about his or her life:

- Is your persona married?
- Does your persona have any children?
- What about hobbies? Perhaps your persona is a cigar aficionado, reads avidly, or was president of his or her high school chess club.

All of these details might seem superfluous when your project deadlines are looming, but this extra level of creativity can help your persona leap off the page and into the forefront of your mind while you're planning your site.

Once you've completed the narrative, add the finishing touches that will complete the picture:

- **Name.** A real name, such as Nathan, Molly, Jon, and so on.
- **Age.** The age of your persona.
- **Work environment.** What kind of computer does your persona use in the office? How fast is his or her connection to the Internet?
- **Technical frustrations.** What does your persona find difficult about working online? What makes him or her want to close her browser window in sheer frustration?
- **Photo.** A photo.

Ideally, your personas should be around one to two pages long, chock-full of important details about your users' needs, as well as those personal details that make the personas come alive. That said, there is no hard-and-fast rule to determine when you're finished working on your personas. Most projects will benefit from a relatively small number of personas; popular opinion ranges anywhere from two to seven personas. Ultimately, *you* are the best judge of your project's specific needs. By better understanding your audience, you can then begin to assess the second most important part of your site: its content, and how your users should access it.

Information Architecture

For a moment, take a look at this book that you're holding. It comprises several hundred pages, each of which might contain a few dozen (we hope worthwhile) concepts. Now, were you presented with all of those ideas printed on one huge piece of paper in a random order, you'd likely ask the bookseller for your money back. There is no way that you could easily find any information on CSS, and this book would be utterly useless to you. Thankfully, you're spared that experience. At every stage of the book's development, there has been a constant attempt to bring order out of this conceptual jumble, on levels both large and small. The goals of the book were established, and then the topics that the book needed to cover were outlined. From those general topics came chapters, and within each chapter came section outlines and headings.

But there's more at play here than simply creating an outline and then tossing in a few paragraphs (or at least, we hope it looks that way). Rather, thought has been given to how the order of these content areas must make sense to you, the reader. Without that consideration for how *you* will read the book — how you will *interface* with the content therein — we might as well have saved everyone some time and printed everything out on that long sheet of paper.

In contemporary Web-speak, *information architecture* (IA) is the term most used to refer to this process. On the face of it, information architecture is a means through which you can define the internal organization of a Web site: to take all of the content for your site, divide it up into easily digestible chunks, and subsequently create a logical navigation structure that makes those chunks easily accessible by your users.

But taken simply in those terms, IA could be seen as a glorified job for a librarian: write down some discrete piece of content on an index card, and file it away in a well-labeled drawer so that someone else might access it. At the heart of it, IA isn't simply about content categorization and site structure. Instead, think of it as imposing order on an otherwise chaotic set of information, in a method that will allow others to more easily interface with it. In fact, the word "interface" implies this two-way street between user and site. Your site might have the most compelling content available online, but what good is it doing your users if they can't locate it? It's our responsibility as designers — and yes, as information architects — to impose order on a seeming jumble of pages, so that others might more easily browse through them.

Of course, figuring out exactly what that jumble contains is the first step to making sense out of it.

Putting It into Practice

Let's say that we've been asked to redesign a small Web magazine named *WebMag 5000* (we never said we were especially strong at branding, but bear with us). In meeting with the site's stakeholders, we learn that *WebMag 5000* is a publication focused on writing articles for online professionals, primarily those working in the Web design industry. Their writers are culled predominantly from their readership, and all contribute on a volunteer basis. *WebMag 5000* has accrued quite the reputation over the past few years, and there is a considerable amount of prestige associated with publishing an article on the site. As a result, the submission rate of prospective articles has been gratifyingly high for the past year or so.

Because it's positively swimming in a sea of great content, *WebMag 5000* features a diverse array of content: articles that analyze new industry trends; tutorials for designers and developers; and reviews of software and books in which the audience would be professionally interested. Unfortunately, readership has dipped a bit. The one common thread in all the users' feedback is that the site is getting harder and harder to navigate. With all of the new content coming in, the original site's design is starting to show its age. Originally built to handle editions posted (sometimes) twice a month, the original design was never meant to handle the amount of content from the current site's tri-weekly editions. The home page is so cluttered with new, featured content that users are having trouble locking onto areas of interest. So, as part of your redesign, *WebMag 5000* has asked you to give the site a facelift that's not only more visually appealing, but is also more useable.

Taking Stock of Your Content

If you're a freelance Web professional, the word "audit" likely conjures up stressful images of unpleasant discussions with various tax authorities. But, in the context of *any* Web project (large or small), a content audit is the first step to bringing order out of Web chaos. It's a method through which you can create an inventory of your content, identify the strengths and weaknesses of what content your site contains, and begin to organize that information into discrete, user-digestible chunks.

For example, let's say that we wanted to organize the books in a bookstore. To do so, we would first take an inventory of all the books currently sitting on the store's shelves, and figure out the best way to organize them. Should we sort the inventory by author, title, genre, or some combination of the three? Should we put all paperback books on one shelf, with all of the hardcover titles on another? Other than the funny looks we're likely garnering from you, these questions are actually quite important. We could settle on an arbitrary order for our books, but it's more important to organize them in a way that will enable us to find our books more easily in the future.

Furthermore, as you sift through books in the store, you're sure to uncover a few that you could bear to part with. Perhaps you've not read one book in years, or you realize that you don't really need all eight copies of *Goodnight Moon*. If you should come across some of these less-valued books, you can either part with them to make your collection more lightweight, or relegate them to a lower shelf so that other books might be featured more prominently. Similarly, any online content audit you perform will probably enable you to identify and remove cruft from your Web site, reducing the number of extraneous pages your users will have to sift through to achieve their goals.

So, turning to *WebMag 5000*, let's examine what we've already learned. In our early requirements-gathering discussion, we've learned that articles, features, and tutorials are the meat of the site. In further discussions, and in clicking around the old site, we discover that there is a significant amount of other information not accounted for:

- ❑ A contact page
- ❑ Rates and information for advertisers
- ❑ The site's copyright information
- ❑ An accessibility statement (for users with special browsing needs)
- ❑ Profile pages for all of the articles' authors
- ❑ Information and tips for prospective writers
- ❑ Pages on the site's history and development
- ❑ A subscription page (which details various for-pay services into which readers might opt)

After creating this content inventory, we can evaluate the merits of each item with the site's stakeholders. Each piece of content should be thoroughly reviewed, and its relevance to the site considered carefully. Do users really need to know about how the site was constructed? How many users currently convert from non-paying readers to paid subscribers? Is the "subscribe" feature even worth maintaining?

Once we've determined which pieces of content are to be culled from *WebMag 5000,* we can sketch out a diagram of the content inventoried thus far. It might look something like Figure 1-2.

Granted, this sketch is purely informational. It lets us quickly see all of the information contained in the site. There's no thought given to how this content is to be ordered, or to how a user might navigate from one area of the site to another. But this is only the first step in our IA planning. Think of this inventory as having taken all of the books off of the shelves. Now we must organize our pages in an intelligent fashion, so that our users might be able to access them more easily.

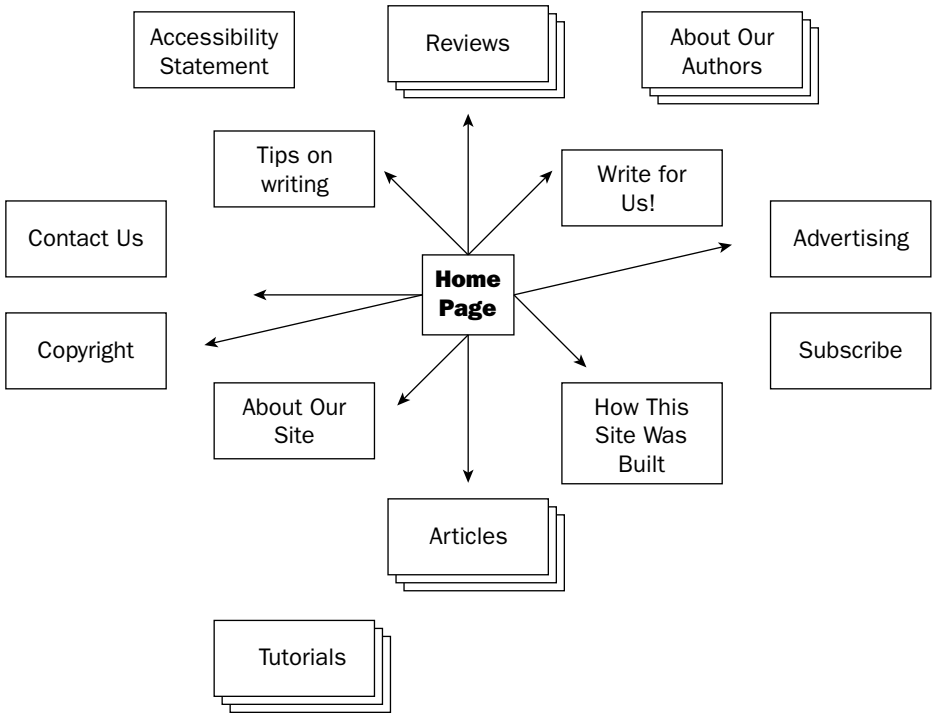


Figure 1-2: A rough content inventory for our site. How do we make sense of it all?

From Inventory to Hierarchy

So, how do we begin to make sense of our Web site so that others might do the same? To begin, it helps to consider your site not as a mass of pages, but as a hierarchy of information. In fact, it's no accident that most site maps look something like a tree: the home page sits at the top (or *root*) of the tree, with the various areas of the site branching out from it. As your users traverse through the different levels of the tree, each branch becomes more specialized than the one that precedes it. Traveling down from the root of the site down to the individual pages, the user is drilling down through these branches to find the content that matters most to them. Because of this, it's your responsibility to make the order of those branches as intuitive as possible.

So, let's say that after an exhaustive amount of research and user interviews, we discover that the audience for *WebMag 5000* can be broken into three types of users:

- **Readers.** These people are most interested in the reviews, articles, and tutorials published on our site. They're mainly consumers of information, so it's important to get them the information they need as quickly as possible.
- **Writers.** In one sense, this group is the lifeblood of our 'zine. Because all of our writers work on a volunteer basis, we must create a prestigious home on our site for the features they produce. Their name — and their work — should be featured prominently, in the hopes that they'll provide additional content in the future.

- **Prospective writers.** Of course, since we're looking for more volunteers to round out our queue of authors, we should have information for them on how to contribute, and what we're looking for. These people will (ideally) be drawn from our regular readership, and they should already be familiar with the topics we discuss.

From this, we can start to flesh out our site's hierarchy. To address the needs of these three groups of users, the highest level of our site might look like Figure 1-3.

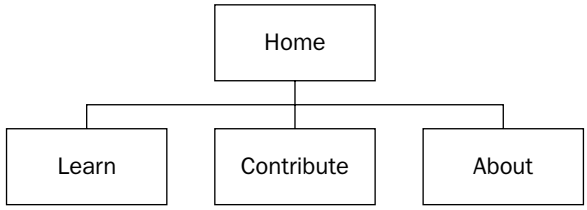


Figure 1-3: Identifying the main areas of our site

Because our primary audience is our readership, a section entitled Learn will house all of the articles, tutorials, reviews, and featured content for our webzine. The other main content area is the Contribute branch, which we hope will pique readers' interest in writing for our site and, therefore, increase our library of high-caliber articles. And finally, the About section contains information . . . well, *about* our little site that all three groups of users might need. Regular readers will get a better sense of our site's aims and philosophies. Writers will get a sense of how our site can give their work more exposure. Prospective writers can use this information to decide if our site would make a good home for their work.

> *There are very few "correct" answers in IA. The approach we've taken here to organizing our hypothetical site could easily be replaced with a number of alternatives. For example, we could have easily placed the site's reviews, articles, and tutorials in their own top-level sections. Whatever structure you settle upon for your own site, it's important to gather feedback from your users whenever possible. There is always the chance that assumptions you make about your users' behavior — no matter how well researched — might not meet with real-world user approval. If your project plan allows it, allotting some time for user testing and feedback can only help improve the result.*

Of course, the structuring effort doesn't end with these three branches. As we said earlier, each level of our site represents a higher, more fine-grained level of detail. Each of these three sections can contain pages, that's true — but they could also contain subsections, which in turn could contain pages and other subsections, and so on, as shown in Figure 1-4.

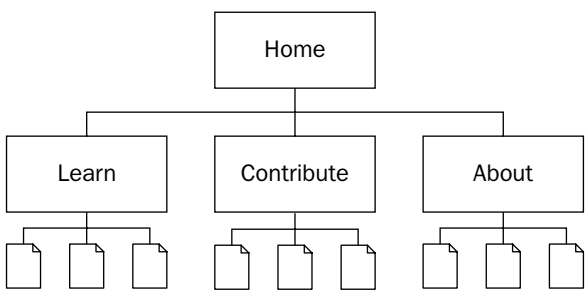


Figure 1-4: Our content hierarchy begins to fill out.

This hierarchical categorization can easily be taken too far. Many sites fall into the trap of over-classifying their content, creating so many different tiers that users are unable to quickly access the content that's most vital to them. More often than not, these too-complex navigation structures arise from a lack of planning when thinking about a site's IA. The impulse to put the book somewhere, *anywhere* on the shelf becomes more important than placing it in an intelligent location, where others might easily locate it.

As tempting as it might be to organize your content beneath 12 levels of deeply nested navigation, that kind of information architecture is more a hindrance than a help to your users. This is another area in which our personas can help us, and enable us to reign in our impulse to over-classify. If we consider what language would be helpful to "Frank" to find a specific section of our site, or which categories make the most sense to "Natalie," then we can frame our IA work around our users' needs and goals. While planning your site's information architecture (and, in fact, throughout every stage of your project), it's important to keep your users' needs at the forefront of your mind. Paying attention to them now will only ensure their satisfaction later.

Building Our Site Map

Now that we've established the different high-level categories of our site, we can finally complete the organization process with a site map. A site map is a graphical, high-level overview that shows the title of each page or section in a Web site in a visually digestible, logical hierarchy (and if you can say that five times fast, we'll have to give you a gold star). The site map is the culmination of our IA work to date. It takes the content inventory we assembled, and organizes the content therein into a logical, navigable hierarchy.

As shown in Figure 1-5, every page and section of the site is listed and named in the site map, with "stacks" of pages to represent a group of related pages. We don't need to display every page in our Articles section, so we can easily group them through this notation. If we remember the content hierarchy diagram from before, then we can see exactly how this high-level tree structure drives the navigation for our site. There are three main navigation sections (Learn, About, or Contribute) and each page of our site has been placed within one of these sections. This means that from the home page, our users will most likely need to browse into one of these content areas *before* being able to navigate to one of the pages below.

Furthermore, we've relegated some pages to an area of "global navigation," which is accessible from every page of the site. These pages don't specifically fit within the bounds of our site's three main content areas, but are considered to be more universally relevant (that is, no matter which page of the site the user is currently reading, this information could potentially be relevant). If we were reading a *WebMag 5000* article and wanted to know about what options (if any) that we would have in reprinting it, we could easily select Copyright from the global navigation and find licensing info therein. Alternatively, while we may not be able to navigate directly from the site's colophon to Writing Tips in one click, both pages would allow users to access the accessibility statement directly.

> *There are a number of applications with which you can build your site maps. Designers often rely heavily on such applications as Adobe Photoshop or Illustrator; others might rely on more IA-specific graphing tools, such as Microsoft Visio (*`http://office.microsoft.com/en-us/FX010857981033.aspx`*) or OmniGroup's OmniGraffle (*`www.omnigroup.com/applications/omnigraffle/`*). Some IA professionals build all their deliverables in HTML, while others swear by their trusty pencil and paper. Try out a number of different tools, and settle on one that best meets your needs.*

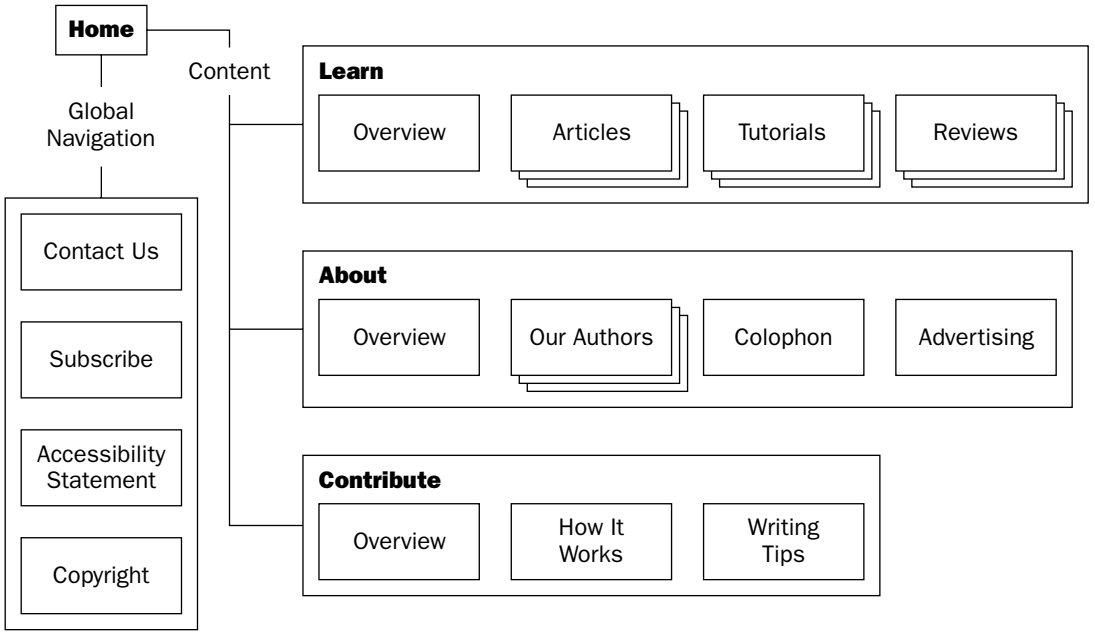


Figure 1-5: A site map for *WebMag 5000*, our imaginary Web site

But the boundaries aren't (and shouldn't be) as distinct as they might seem. Presumably, there will be content on the home page that will feature articles and other internal content. Because of this, it's also likely that there will be some facility on that front page allowing users to move directly into an individual article, two levels down in the navigation. Should our site map reflect this? If so, how should it?

In all honesty, it's your decision as to how exhaustively your site map charts the relationships between different sections of the site. If we were to graph every point in which users could move between each area, our site map might start to look like a map of downtown London, riddled with arrows and lines and making very little sense. Rather, we can outline some of the more important relationships between the different sections. For example, let's say that every article, tutorial, and review published on our site features a link to the author's profile page in the About section. Conversely, each author's profile will contain links to any articles, tutorials, and reviews that he or she has published on our site. If our site map needed to reflect this relationship, we might update the document to look like Figure 1-6.

This kind of cross-linking helps promote deeper relationships between different sections of your site, which allow users more than one way to access the information they're looking for. A user might come to your site looking for information on a particular piece of software, and could happen upon additional work by the author on his or her profile page. By creating relationships across the different content categories, we create a richer interface for our users, as well as a more scalable foundation for any future content we add to our site.

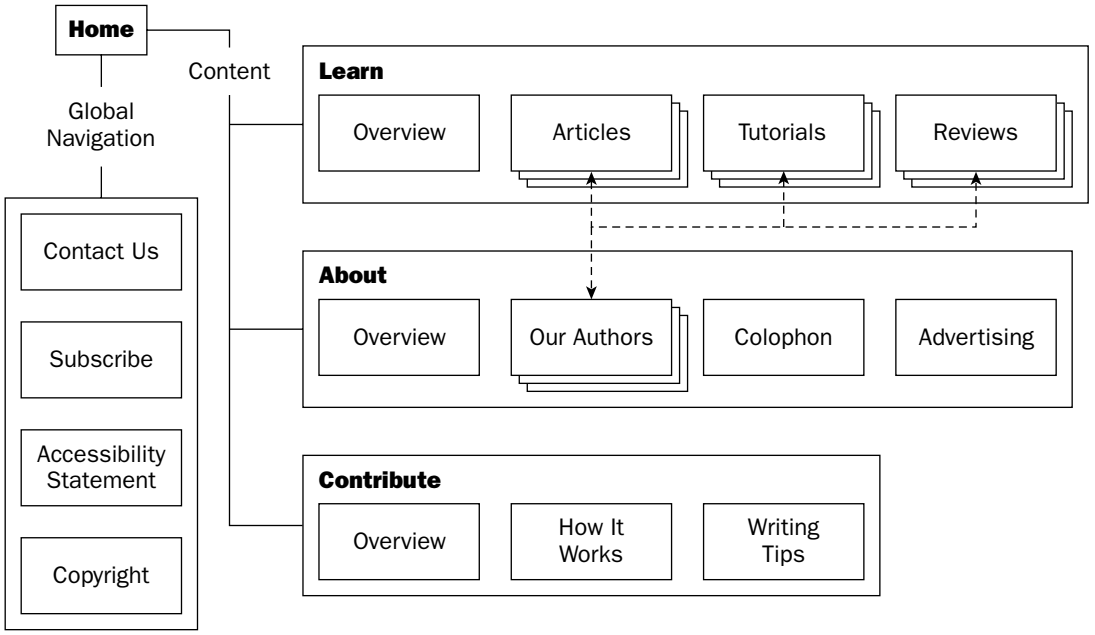


Figure 1-6: Annotating the relationship between sections

But once we've built our site map, our work is only halfway done. Now that we understand the "big picture" of our entire site's IA, how do we organize content on the individual pages?

Wireframes: Blueprints for Your Pages

A *wireframe* is a line drawing that represents the different areas of content on a page. Much in the same way that a site map gives us a sense of how different areas of our site relate to one another, a wireframe enables us to see how the different pieces of content on a given page will interact. These black-and-white line drawings imply structure *only.* Bereft of color, typography, or any other design elements, wireframes force us to establish a layout for our pages.

Typically, it's not necessary to build a wireframe for every single page of our site. Rather, we can sketch out a wireframe for each unique page layout in our site (such as your home page, search results, blog entry, and so on). By starting with these plain-looking documents, we can identify any possible usability issues before we've begun fleshing out our site's design.

Just as we started drafting our site map by taking inventory of the types of content our site would contain, we must identify the different areas of information that our page's layout will house. If we are building a wireframe for a *WebMag 5000* article, then let's assume that the page will contain the following information:

- **Content.** The article's content.
- **Related articles.** A listing of articles related to one the user is currently reading.
- **Logo.** The *WebMag 5000* logo.

- **Global navigation links.** From our site map, these are Contact Us, Subscribe, Accessibility Statement, and Copyright.
- **Primary navigation.** These are links to the Learn, About, and Contribute sections, which we defined in our site map.
- **Secondary navigation.** These are the navigation options within each section. If we were in the About section of our site map, the subnav options would be Our Site, Our Authors, Colophon, and Advertising.
- **Author's name.** This should link to the article's author profile.
- **Category listings.** These should list under which content categories the current article has been filed, so that users might view additional articles in these categories.
- **Advertising banner.** The client tells us it needs to be 280 pixels wide and 120 pixels tall, but we don't need to be concerned with layout at this stage.

Equipped with this list, we should now begin to group these chunks of content into different categories, which should speak to how the different content areas relate to each other. For example, with the content we've listed, we might create the following content hierarchy:

- Content:
  - The article's content
- Related content:
  - The author's name
  - Category listings
  - Related articles
- Branding:
  - The *WebMag 5000* logo
- Navigation:
  - Primary navigation
  - Secondary navigation
  - Global navigation
- Advertising:
  - The banner advertisement

These content groups are broken down largely into conceptual chunks (that is to say, that ancillary bits of information have been dumped into Related Content, the banner advertising has been oh-so-cleverly labeled Advertising, and so on). Were you working on a page that had a higher degree of interactivity (such a complex form that requires the user to enter a sizeable amount of data), you might use headers that describe the data contained therein (My Personal Information, My Company Information, Shipping Address, and so on).

Armed with this high-level list, we can turn these content groupings into a rough wireframe that demonstrates how these relationships might play out on a page.

In Figure 1-7, we've settled upon a rather traditional two-column layout for the wireframe. Most of our content categories have translated into discrete areas on the page, placed according to the amount of visual weight we need to allot them in our layout. For example, the client's logo has been placed at the top of the wireframe, in order to maximize the visibility of the *WebMag 5000* brand. The one content category that "straddles" the page layout is the site's navigation. While primary and secondary navigation have been placed immediately below the logo to facilitate easy navigation, the "global" navigation has been relegated to the footer of the page. While this might indicate to some users that it is perhaps the least important area of the page, we decided to position it at the bottom in the hopes that its consistent placement would enable users to find it more easily.

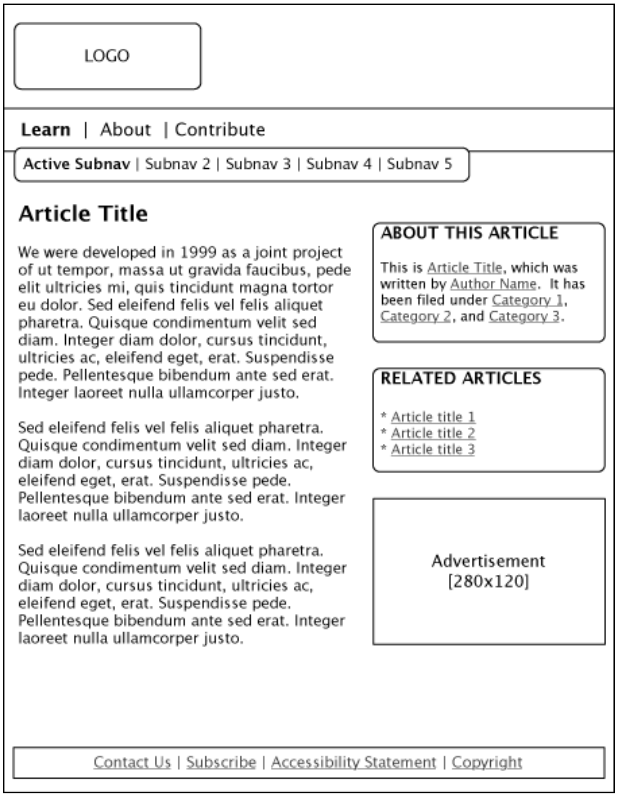


Figure 1-7: The finished wireframe, ready for a design to be hung upon it

The rest of the groupings play out largely as you might expect. The article's content area is of primary interest to the site's users, so that has been given the most weight on the page. In the right-hand column, many of the "related content" information has been stored to give it the most visibility, while remaining subordinate to the primary content area. The author's name and the article categories have been placed in a block named "About This Article." Other related articles have been moved into a separate area immediately below. The advertising banner is placed in the sidebar to give it maximum visibility without detracting from the content as a whole.

So, ultimately, the architecture dictated by the wireframe should reflect the needs of both your client and your users; the decisions we've made in this example wireframe might not be appropriate for your site

or its requirements, but may at least present an idea of how to move forward. However you decide to perform this ever-delicate balancing act between client and user requirements, keep the goals of the page at the forefront of your mind. Build a clear, usable interface, and the rest will fall into place.

Sounds almost easy, doesn't it?

> *Jason Santa Maria's "Grey Box Methodology" (*`www.jasonsantamaria.com/archive/2004/05/24/grey_box_method.php`*) describes one graphic designer's approach to planning a design. His approach is a compelling read, and tackles the challenge of defining a page's architecture from a slightly different perspective. Whereas we took stock of the page's content before considering the layout, Santa Maria takes a more visual approach. The two methods achieve the same goals, however — both force the designer to think in terms of how the layout can help the user before planning the more aesthetic details of a design.*

Beginning the Design

So, the planning has been finished, the site map has been approved, and the wireframes are set. We can finally set aside all of this theoretical nonsense about "interface," "requirements," and "change management," and stop worrying about taking "notes" in "meetings" with "stakeholders." At long last, we can finally get into actually *designing* our site . . . can't we?

As much as we'd like to tell you otherwise, the actual construction of a site requires no less planning than the rest of the project. Sitting down to design and build a site is absolutely one of the most rewarding parts of any Web project, that's true. But this is the phase of the project in which all of your careful planning and analysis pay off. That's not to say that imposing a process upon the design process should remove any of the fun from it. Rather, some of the most compelling Web designs are a direct result of the constraints placed upon them.

That's not to say that there aren't many talented designers who can create a stunning personal site with little or no outside direction. However, it's another challenge altogether to create a design that simultaneously furthers a company's brand, ensures that the site is supremely user-friendly, *and* remains aesthetically appealing. From these competing needs, the true beauty of Web design originates.

Setting the Tone for Your Site

With a firm understanding of the scope of our project, we can then craft a design that speaks to the needs of both groups. In order to do so, however, we should establish the tone of our site. After all, we might settle upon a different design direction for a small law firm than we might for an online publication. Each company would have its own target audience, its own brand identity, and its own business requirements. As such, an understanding of what those goals are is integral to building a site that achieves them. Let's take a brief look at one site that exemplifies this attention to goal-driven design: Cinnamon Interactive (`www.cinnamon.nl/`).

In the early days of CSS adoption, far too many style sheet–driven sites were doomed to "look like Web standards" — their layouts were boxy, their palettes flat, and they made far from a compelling case for abandoning table-driven design techniques. Cinnamon Interactive was one of the earliest sites to showcase the true power of CSS as a tool for bringing gorgeous designs to the Web. Launched in January 2003, it featured an incredibly nuanced design (see Figure 1-8). The site's layered typography, beautiful use of color, and well-implemented access enhancements are facilitated by a foundation of CSS and valid XHTML, which ensures that the site is as impressive under the hood as it is above it.

Figure 1-8: Cinnamon Interactive. Fashionable and functional, this beautiful design floors random surfers while driving prospective clients into their portfolio.

But it's not just the jaw-dropping aesthetics (or the code behind it) that make Cinnamon a success. Rather, the goals of the site are beautifully reflected in its design. The cinnamon shaker logo is subtly mirrored in the photography on the home page, which creates a heightened sense of the company brand. This attention to the corporate identity is sure to impress not only random Web surfers, but also corporate clients looking for the same level of brand savvy for their own company.

Furthermore, the bulk of the home page is dedicated to links to the company portfolio. Not only is it the first link in the site's primary navigation (after the link back to the "*Voorpagina*," or home page), but the splash graphic on the home page is one big link directly into their body of work. This emphasis on advertising their portfolio is intended to not only draw interested users into some of their past projects, but ultimately move over to the "contact" page so that they might ultimately contract with Cinnamon for their own design initiatives.

Finding Inspiration

Of course, translating these goals into an attractive design isn't always the most elementary process. Just like a writer under deadline, it's easy for designers to get blocked. When presented with a creative brief, client requirements, and a mountain of documentation detailing your users' needs, a blank canvas in Photoshop can seem an almost impassable roadblock. So, how do we get started? How do we go from requirements to sleek, professional-looking design in nothing flat?

Simple — by seeing how others do it.

Don't worry, you bought the right book. This isn't *Professional Plagiarism* we're writing here. However, establishing and maintaining an awareness of current Web design trends is one of the keys to improving your own craft. the most rewarding part of working and designing online is that there is always something new to learn, another excellent resource you've not discovered yet. If we're feeling stumped by how to style a particular page element, uninspired by a corporate color palette, or just generally strapped for ideas, then seeing how other designers work in the face of adversity is always a welcome aid. So, whenever we start to feel as though we're pushing up against the constraints of our own design skills, we spend a bit of time browsing for inspiration.

This browsing could occur on-line or off-line. Whether walking through a museum, browsing through a magazine stand, or surfing through some well-designed Web sites, the medium you pick is less important than exposing yourself to well-reasoned, well-executed designs for an hour or two. Thankfully, we're able to keep that hour or two from turning into eight or nine by visiting a number of well-known design portals, or online communities dedicated to evangelizing some of Web design's brighter spots.

Over the past year, a number of sites evangelizing CSS-based Web designs have cropped up. The most recent addition is Stylegala (`www.stylegala.com/`). It features a design as compelling as the sites it covers. Each featured site in the gallery is accompanied by some thoughtful critiques by Stylegala's creator and designer-in-residence, David Hellsing. The site's users are allowed to vote for favorites within the gallery, as well as provide additional commentary. The discussions are always interesting, and the sites are all fine examples of what the Web (and CSS) can do.

Kaliber10000 (`www.k10k.net/`) isn't so much a community as an *authority*, providing a near-dizzying array of design-related news, tips, and tutorials on an even more dizzying basis. Founded in 1998 and staffed by some of the brightest designers in the industry, Kaliber10000 provides everything from screenshots of user desktops, to links to Flash-heavy portfolio sites. K10K won't just rid you of your design block — rather, it will help you obliterate it.

Selecting a Layout: Fixed or Liquid

Often characterized as one of the unholy wars of Web design, the debate between fixed or liquid layouts is the designer's version of "Chevy versus Ford" — each side has its fierce loyalists, with very little middle ground between the two poles. As we begin building our site, it's worth acknowledging this heated debate, so that you might choose a method that best fits your needs and those of your site.

Fixed Width

A fixed (or "ice") layout is one whose width is constant and unchanging. No matter how large or small the user's browser window becomes, the actual width of the page's content area will remain, well, fixed. The benefit to this approach is that it lends ultimate control to the designer, who always knows the dimensions of the canvas he or she is designing upon. For professionals coming from a print background, this is approach is a rather intuitive one. If the width of the content area is a fixed constant, then a designer can hang perfectly sized graphics upon that canvas with pixel-perfect precision.

However, the success of a fixed-width design is contingent upon an assumption: namely, the designer's assumption of the width of the user's browser window. Should the browser's width ever become narrower than that of the page, then a horizontal scroll bar will appear, forcing the user to manually scroll the page from left to right to see the site as its designer intended (see Figure 1-9).

800x600

1024x768

1280x1024

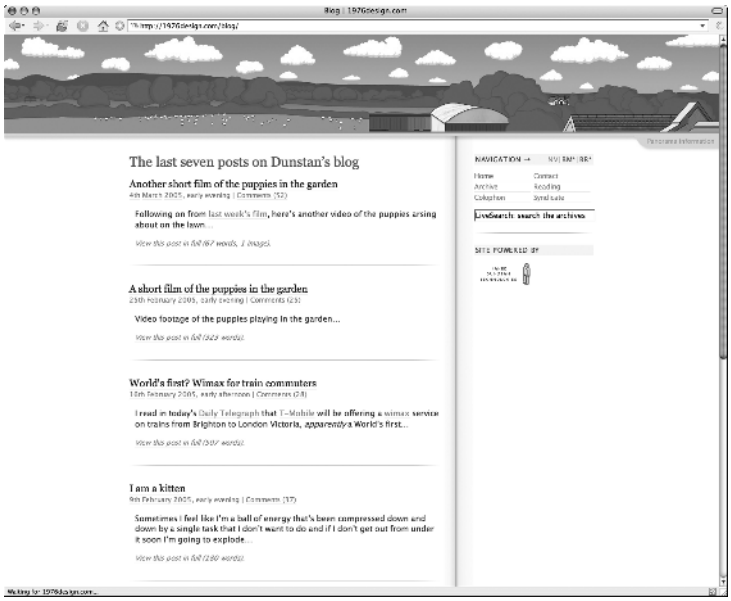


Figure 1-9: 1976design.com is the weblog of Dunstan Orchard, one of this book's authors. It is built with a fixed layout, which causes a horizontal scroll bar to appear at smaller screen resolutions.

Conversely, fixed-width designs are often unkind to larger screen resolutions. When browsing at higher resolutions and with a fully maximized window, a design optimized for smaller window sizes can be drowned out by unused white space. Our users' displays are constantly increasing, which makes fixed-width designs less than future-proof — a site designed for the resolutions of today will likely need to be revisited in a year or two, as its users clamor for a more intelligent use of the window widths available to them.

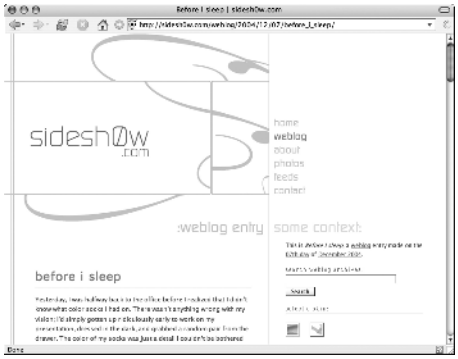
Liquid Layouts

Proponents of liquid (or "fluid") layouts are quick to counter that fixed-width designs set an unfair technical assumption on our audience: there is no guarantee that our user's browser window is large enough to accommodate the width specified in our design, and users with smaller windows should not be penalized by the sites they visit. So, to that end, liquid layouts are designed to be entirely flexible. As the browser window expands or contracts, the page's layout will follow suit, ensuring that the content fills the display at any window size (see Figure 1-10).

Most apologists for fixed-width techniques will say that their users' screen resolutions have been increasing over recent years, and that designing for smaller screen resolutions is no longer necessary. Their server logs might tell them that an overwhelming majority of their users are running resolutions such as 1024×768 or 800×600, and that building a design that caters to smaller screens is less vital than it was in the early days of the Web. Designers on the other side of the debate will quickly counter with the fact that *screen resolution does not determine the size of the browser window*. If a user's screen resolution is set to a specific width, the browser isn't automatically maximized to occupy all of that space. By making that assumption (and serving up a fixed-width design to match), designers can potentially exclude users who browse at smaller window sizes, forcing them to resize their pages to meet the designer's design criteria. Instead, fluid layouts are agnostic when it comes to the size of the browser window. By definition, they attempt to subvert the design to the preferences of the user, rather than the other way around.

Of course, liquid layouts are not without their flaws — no design technique is a silver bullet, and this is no exception. Just as it is easier to read text that has been divided into separate paragraphs, it is also easier to read text that has been set to an optimal width. This is where most fluid layouts break down. By allowing their content to reflow as the size of the browser window increases, that content can be increasingly difficult to read at larger window sizes. Studies have shown that the user's ability to comprehend onscreen text suffers slightly when the length of the line being read exceeds 4 inches (`http://psychology.wichita.edu/surl/usabilitynews/42/text_length.htm`); so while a fluid layout might be ideal for small-to-medium browser window widths (see Figure 1-11), legibility can suffer quite a bit when the window expands beyond that threshold (see Figure 1-12).

800x600



1024x768

1280x1024

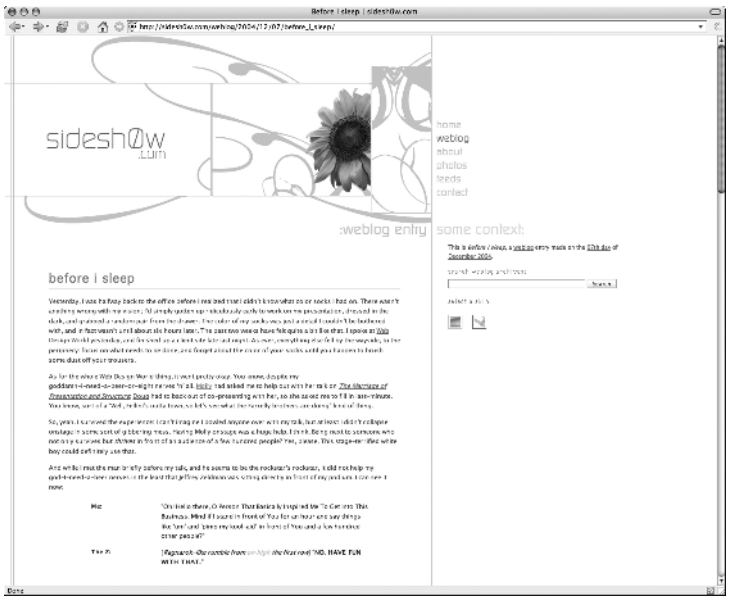


Figure 1-10: sidesh0w.com is the weblog of Ethan Marcotte, another of this book's authors. It features a liquid, resolution-independent layout, which some might find difficult to read at larger screen resolutions.

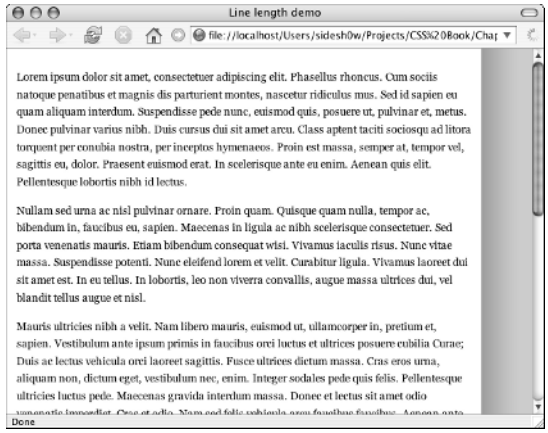


Figure 1-11: A page of text with a fluid width. Seems quite manageable at a narrow width, doesn't it?

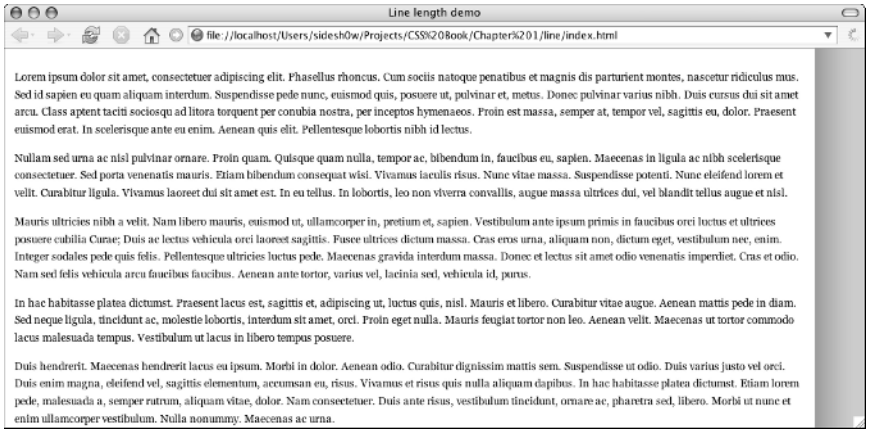


Figure 1-12: The same fluid block of text, but at a larger screen resolution. The line lengths have become unmanageable and are difficult to scan.

Furthermore, the lack of control over the content area's width can be difficult to balance when placing fixed-width elements within it. When a Web designer knows the physical dimensions of the page, then graphics can be designed specifically to fit within that space. Even the staunchest fluid-width proponents are envious of their fixed-width comrades' ability to place a perfectly sized graphic across the top of a content area. Most liquid designers are forced to use graphics of an arbitrary width, which could break the layout as the window becomes infinitely small.

> *Richard Rutter, Web designer and proponent of liquid layouts, has published a number of experiments with placing wide images within a fluid-width container (*`http://clagnut.com/blog/268/`*). He proposes a number of interesting CSS workarounds that may be of interest to those pursuing liquid layouts.*

Gaining Some Perspective

The truth is that both methods have their strengths and weaknesses. Neither is inherently superior to the other. In the hands of the right designer, either can be equally effective (or, in the hands of a less-talented designer, ineffective) in a site's design. Of course, the relative strengths of one approach can be debated *ad nauseam* at the expense of the other's faults — and as in most design communities, it is debated endlessly.

Rather than seeing either layout model as "The One True Path" of Web design, we would suggest that the two models are *tools*. The true key to success in either method is applying it intelligently to your work, with an overall sensitivity to the context of each element on your page. It's not the relative width of a design's content area that makes for a successful site, but rather the thinking before and behind the design that distinguishes it.

Summary

With this chapter, we've completed an overview of some of the highlights of a typical Web project. This overview is far from exhaustive and, in fact, could be called a bit of a gloss. Entire tomes have been written about project and client management, and we've dedicated only a few pages to it. However, we hope that this sketch of an entire project lifecycle demonstrates that there is far more than well-coded style sheets that make a site successful. In our experience, a successful project is rarely determined by process, documentation, and deliverables as much as it is by good communication. If the designer and the client make a concerted effort to work closely together throughout the project, then each will be more aware of the other's needs and expectations as both move toward meeting (and exceeding) the project's goals.

So with a better understanding of a project lifecycle, we can now turn to two of the more critical components of its design: XHTML and CSS. We'll examine the relationship between these two important technologies, as well as discuss some tips for more effectively integrating them into your design workflow.