> Production's goals are simple: No misinterpretation of user capabilities or project goals; create a website that looks and works the same for every user. No duplication of effort; code each HTML page only once.

Phase 4: Production and QA 06

Phase 4: Production and QA

With the legwork and planning of the site essentially completed, now is the time to create, implement, and integrate. Phase 4 is where the actual building happens. You have defined and structured the project and have developed a look and feel — here is where you put together all the pieces you have designed, planned, and gathered. If this were pie baking instead of web design, consider your fruit sliced, your ingredients measured, your oven preheated, and your crust shaped. After one more check of your recipe, you are ready to bake.

WHAT THIS CHAPTER COVERS		
PREPPING	**BUILDING**	**TESTING**
> Assessing Project Status	> Slicing and Optimization	> Understanding Quality Assurance Testing
> Establishing Guidelines	> Creating HTML Templates and Pages	> Creating a QA Plan
> Setting File Structure	> Implementing Light Scripting	> Prioritizing and Fixing Bugs
	> Populating Pages	> Conducting a Final Check
	> Integrating Backend Development (if Applicable)	

This phase is divided into three sections — **Prepping**, **Building**, and **Testing** — a production workflow aimed at keeping the project's HTML construction on track. Whether your budget is upwards of $100,000 or under $10,000, the steps delineated here work for all web projects — redesigns and initial designs alike. Either way, your goal is simple. No duplication of efforts. Code each HTML page only once.

ASSESSING PROJECT STATUS

Before production actually starts, take a moment to review the project's status. Did the scope increase? Is the project on budget? Has the all-important content arrived? And is your team ready for the production task ahead?

Right here at the beginning of the production phase we must address an important fact: The web is driven by HTML. We assume that you or someone on your team has an understanding of the

HTML process, either through pure hand-coding using BBEdit or Allaire's Homesite or the like, or by using a WYSIWYG editor such as Adobe GoLive, Macromedia Dreamweaver, or Microsoft FrontPage. Here's the burning question: What is the level of that understanding?

Reassess your HTML production team's capabilities now that you know the true extent of the design and technical requirements, and if you are not qualified to make the assessment, find someone who is. Coordinating web production takes both ability and experience. Depending on your team's level of expertise, you will need to determine the true level of complexity that the site production can handle. For example, if you are creating a 20- to 40-page brochure site with light JavaScript, you can probably get away with using a WYSIWYG editor. If the site is more complex — intricate tables and/or frames usage, additional scripting and/or DHTML implementation — you will need to have the knowledge to troubleshoot problems along the way, which usually means utilizing people with a fluid understanding of HTML. This tends to call for people who can code pages by hand or who at least can read and understand the code well enough to tweak HTML and troubleshoot during the production process.

And before any coding truly begins, a final just-before-production-starts review of audience needs (browsers, screen size, connection speed), technology (plug-ins, scripting, backend needs), and redesign goals (download size, user experience goals) can only help. You will have to address complex questions about servers, directory structure, and the HTML production specifics that may have been left until this phase. The Client Spec Sheet will help.

Your goal? No misinterpretation of user capabilities or project goals. No backtracking. Code each HTML page only once.

ESTABLISHING GUIDELINES

Establishing clear guidelines for HTML production during the initiation of a web redesign project helps to answer questions and avoid costly backtracking. The Client Spec Sheet sets parameters for audience capabilities and technical standards for the site. This is a worksheet. It is long and detailed and technical. The client may simply say, "I don't know. You're the expert; you tell me." Some discussion is likely necessary. For instance, the project manager or lead production designer might have to explain what effect choosing to support 3.x browsers might have on being able to support certain functionality, or what effect selecting Flash might have on wanting to support dial-up modem connections, and so on.

The Client Spec Sheet is available for download from www.web-redesign.com. Due to its length, we

< PREPPING
> Assessing Project Status
> Establishing Guidelines
> Setting File Structure

CHAD KASSIRER ON KNOWING YOUR CLIENT BEFORE YOU CODE

Clear communication with the client is the key to a successful web project. Before beginning the production process, it is important to have agreed on and signed off on two things: a composite of the target audience and the client's expectations concerning the site production details. To assist with this process, I rely on a Client Spec Sheet to document these items. Ideally, this document is administered shortly after the project has been kicked off. This way, there is one central document that can serve as a guideline for everyone contributing to the building process. Not only does this assist in all phases of the process from information architecture to design to production, it also establishes some necessary parameters for the site's requirements and identifies possible limitations early on.

It is every web designer's, programmer's, and production engineer's goal to create a website that looks and works the same for every user. However, with the numerous possible combinations of platforms, browsers, connection speeds, and monitor resolutions, this is nearly impossible to accomplish. To decide the best way to design and build the website, you need to identify the target audience. Once this is established, you can tailor the site to best suit this audience's needs before being concerned about other users. This is not to say that no one other than your target audience is important, but the client's priorities need to be established. These priorities will impact decisions made during the production process. A more realistic goal is to make the site as close to perfect as possible for the target audience while still being functional for everyone else.

By initiating a conversation at the start of the project, a dialog is created between production and the client. During this conversation, the client's expectations and preferences can be discussed before deciding the direction the client wishes to take and documenting this on the Client Spec Sheet. The production lead, as the integrator of design and engineering, uses this document as a reference for making decisions during the design and production phases. When used properly, the Client Spec Sheet is extremely useful and saves time and money by eliminating ambiguities, which cause unnecessary delays and frustration.

I recommend using the Client Spec Sheet early in the process. It documents and clarifies to everyone what the initial goals of the project are, even if changes are made during the process. In case the requirements or expectations of the client change, the Client Spec Sheet also serves as a contract to refer to when additional costs are required or disputed. By using the Client Spec Sheet as a reference to help guide your decisions throughout the project, you can build a site with the client in mind.

As director of production for web development shops Red Eye Digital Media and Idea Integration/San Francisco, Chad Kassirer (www.whatdesign.com) has played a key role in the production process for many award-winning websites such as Adobe's Splatterpunk, One World Journey's Georgia Revealed, and SFMOMA's Making Sense of Modern Art. Chad has never appeared at any web conferences or written any books, but he knows people who have.

could not show it in its entirety in this book, so we show only the first two parts: Target Specifications, and Functionality and Features (see worksheet on next page). All told, it is five parts long, as follows:

Part 1: Target Specifications

Part 2: Functionality and Features

Part 3: Design and Layout

Part 4: File Structure and Directory Preferences

Part 5: Server and Hosting Information

As tedious as filling out this worksheet may be, the information needs to be addressed and answered before any HTML production can begin, and that includes conferring with the visual designers at the onset of Phase 3: Visual Design and Testing. Encourage client feedback within a short timeframe. This information should be back to the team and analyzed before the visual designers start developing concepts and definitely before the production designers start building the Protosite.

The team's lead HTML production designer should be the team contact; the project manager may or may not be as technically savvy. Have the client — or the client's key tech lead — answer all questions as thoroughly as possible, adding additional comments as necessary. Encourage the client to write "N/A" next to nonrelevant items and to identify areas in which advice, suggestions, or clarification is needed. Filling it out should be taken seriously; the results from this analyzed worksheet serve as a set-in-stone guide for production.

The worksheet on the following pages will help you articulate and identify the technical parameters of your site redesign, including specific questions regarding target audience connectivity capabilities, browser versions, functionality, and actual file structure. When you are finished, please return all compiled information back to the project manager on the web development team.

Scope Expectations Meet Scope Reality

An estimate of 100 hours can easily turn into 300 hours if the complexity of the site has been underestimated. In Phase 1: Defining the Project, you estimated the project's budget based on the projected scope. Did you plan on 50 pages and now there are 120, or are you still on target? Assess. Has your scope grown, either through Scope Creep or as a result of client-requested changes and/or additions? If so, you will need to either increase the budget or downsize the allocation of hours… or take a loss. Regardless, if you haven't yet addressed potential budget changes with your client, do it now — *before* you start coding. And make sure you have included resources for QA along with the time necessary for fixes.

HTML Expertise

Although this chapter concerns HTML, it is not about how to code, the ins and outs of HTML, coding theory, nor advanced scripting implementation. We focus on the redesign workflow process and how it relates to the actual site production — keeping your project moving smoothly, staying on schedule and on budget. For guides to actual hands-on coding, seek alternative resources such as *HTML Artistry: More Than Code* (New Riders, 1998) by Ardith Ibanez and Natalie Zee or *Creative HTML Design.2* (New Riders, 2001) by Lynda Weinman.

TARGET SPECIFICATIONS | PART 1

Establishing clear audience specifications enables production to have a targeted goal. It is often difficult, if not impossible, to maintain consistency of experience from one browser or platform to the next. It is important for the HTML production team to understand not only the target end user but also who can be left behind.

	Existing Site Specs (Check One Below)	Priority/Target (Check One Below)	Others to Support (Specify One or More)
Resolution	☐ 378×544 (web tv) ☐ 1024×768 ☐ 64×480 ☐ Other (explain) ☐ 800×600	☐ 378×544 (web tv) ☐ 1024×768 ☐ 64×480 ☐ Other (explain) ☐ 800×600	☐ 378×544 (web tv) ☐ 1024×768 ☐ 64×480 ☐ Other (explain) ☐ 800×600
Browsers	☐ Netscape ☐ AOL ☐ Internet Explorer ☐ Other (explain)	☐ Netscape ☐ AOL ☐ Internet Explorer ☐ Other (explain)	☐ Netscape ☐ AOL ☐ Internet Explorer ☐ Other (explain)
Browser Versions	☐ 3.x ☐ 6.x ☐ 4.x ☐ Other (explain) ☐ 5.x	☐ 3.x ☐ 6.x ☐ 4.x ☐ Other (explain) ☐ 5.x	☐ 3.x ☐ 6.x ☐ 4.x ☐ Other (explain) ☐ 5.x
Platforms	☐ Macintosh ☐ Other (explain) ☐ Windows	☐ Macintosh ☐ Other (explain) ☐ Windows	☐ Macintosh ☐ Other (explain) ☐ Windows
Connection Speed	☐ Wireless/handheld ☐ DSL/cable ☐ 28.8/33.6 dial-up ☐ T1/T3 ☐ 56.6k dial-up	☐ Wireless/handheld ☐ DSL/cable ☐ 28.8/33.6 dial-up ☐ T1/T3 ☐ 56.6k dial-up	☐ Wireless/handheld ☐ DSL/cable ☐ 28.8/33.6 dial-up ☐ T1/T3 ☐ 56.6k dial-up
Page Download Size (typical page)	☐ 30K and under ☐ 80K (graphic heavy, animation) ☐ 30 to 80K (typical page) ☐ 100K+ (not recommended unless a high-bandwidth site)	☐ 30K and under ☐ 80K (graphic heavy, animation) ☐ 30 to 80K (typical page) ☐ 100K+ (not recommended unless a high-bandwidth site)	☐ 30K and under ☐ 80K (graphic heavy, animation) ☐ 30 to 80K (typical page) ☐ 100K+ (not recommended unless a high-bandwidth site)

<<< This worksheet is available in full (all five parts) for download at www.web-redesign.com >>>

FUNCTIONALITY AND FEATURES | PART 2

The addition of specific technologies that allow greater functionality can greatly enhance your site. These same features can exclude a percentage of your audience, however, and can cause production scope to increase, usually due to unforeseen technical errors and troubleshooting. Please identify which features you already have on your site and how they are currently being used. Please also indicate which features you are looking to add and how you foresee them being used.

	Preferences/Status (Current and New Site)	Issues	Comments and Usage Details (How It Is or Will Be Used)
Frames	☐ Used on current site ☐ Will not be using ☐ Yes (use on new site) ☐ Not sure (list comments)	Causes difficulty printing and navigating and may require additional scripting and quality-assurance testing. Causes difficulty for search engines. With a multiframe setup, could incur extra programming and QA costs.	
Forms	☐ Used on current site ☐ Will not be using ☐ Yes (use on new site) ☐ Not sure (list comments)	Requires additional programming and integration. Specific and detailed information is necessary to determine complexity.	
JavaScript	☐ Used on current site ☐ Will not be using ☐ Yes (use on new site) ☐ Not sure (list comments)	Does not require a plug-in, but is not supported by all 3.x browsers. Adds noticeable download time.	
Pop-Up Windows	☐ Used on current site ☐ Will not be using ☐ Yes (use on new site) ☐ Not sure (list comments)	May require use of JavaScript; may not be supported by 3.x browsers. Inconsistent size and placement depending on platform and browser.	
Cascading Style Sheets (CSS)	☐ Used on current site ☐ Will not be using ☐ Yes (use on new site) ☐ Not sure (list comments)	Does not require a plug-in. Allows for global updating of fonts, colors, and styles. Supported by most 4.x browsers and above.	
Dynamic HTML (DHTML)	☐ Used on current site ☐ Will not be using ☐ Yes (use on new site) ☐ Not sure (list comments)	Does not require a plug-in. Used to create special features such as dynamic menus. Supported by most 4.x browsers and above. May require additional testing, programming, and QA.	
Flash	☐ Used on current site ☐ Will not be using ☐ Yes (use on new site) ☐ Not sure (list comments)	Requires a plug-in. Sometimes causes accessibility/download issues; may require two versions of a site to be built (HTML only and Flash) or use of a browser sniffer.	
Media (Video/Audio)	☐ Used on current site ☐ Will not be using ☐ Yes (use on new site) ☐ Not sure (list comments)	Requires plug-ins. May involve download and processing time. If using any type of media, please list as much detail as possible, including type of media, format, and desired output.	

<<< This worksheet is available in full (all five parts) for download at www.web-redesign.com >>>

Do a project-wide time check. You should have been tracking your hours on a weekly basis, so this should be a relatively easy assessment. How much of your allocated time and resources have you used up? Are you on budget? Has the scope increased? Do you have the time left in your budget to comfortably complete the site? Knowing how many resources and hours are necessary to complete a project's production and QA is regularly one of the gray areas in project estimating. The hard truth? Most things that appear to be simple are not and will take much longer than you estimate. Coding an HTML page or template can take a few hours or a few days — it is one of the factors contributing to Scope Creep that is extremely difficult to gauge until actual production begins.

Readdress Audience Capabilities

Focus on your user. You are producing the site based on the capabilities of your target audience, and you cannot translate the visual design into HTML unless you know your parameters: target operating systems, browsers, monitors, and connection speeds. Use the results from the Client Spec Sheet as a guide.

Check Content Status

Content should be in — all of it. But chances are it won't be. You must be on top of content status. Your Content Delivery Plan was clear: Content must be in before production can commence in earnest. Alert your client that the time has come, that a content freeze is imminent.

Announce to the client early in the process, as early as with the initial proposal that accompanied the budget, that if/when content is late, production will be held up and cost overruns will commence. Billing for overruns is never painless, but it will be far more viable if you warn the client of consequences ahead of time.

SCOPE: ARE YOU ON TARGET?	
Sitemap	How big is the site? How many pages? Is it what you planned for?
Visual Complexity	Is the slicing a nightmare or fairly straightforward?
Light Scripting Needs	DHTML, JavaScript rollovers, forms, pop-up windows, frames, pull-down menus, and so on. What did you plan for when you initially budgeted/scheduled? What are you now slated to include? Do the two match up?
Backend Engineering	Are the engineers on budget/schedule? Have the requirements been adequately defined, and do they still match the scope/cost expectations?

Considering Accessibility

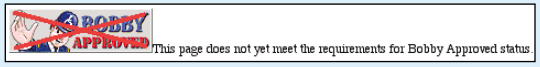
Imagine your only web access is through a browser that does not support images. Pick any site — if the navigational aids and buttons don't have descriptive ALT tags, you won't be able to differentiate between graphics.

Recent government backing and new Accessibility with Disabilities Act (ADA) standards are behind an ongoing push to support full-access web sites. This new set of standards, led by the World Wide Web Consortium (www.w3c.org), aims to connect all people to the web regardless of disability, including the handicap of older browsers or outdated technology. Understanding accessibility needs before you start coding — especially if your site is bound to comply to accessibility standards — will avert damage control later (for example, coding ALT tags into 100 pages instead of having done it once at the outset on the HTML template).

Here are two free tools that can help you test your site for accessibility after it's up: Bobby and Macromedia's Section 508 Accessibility Suite.

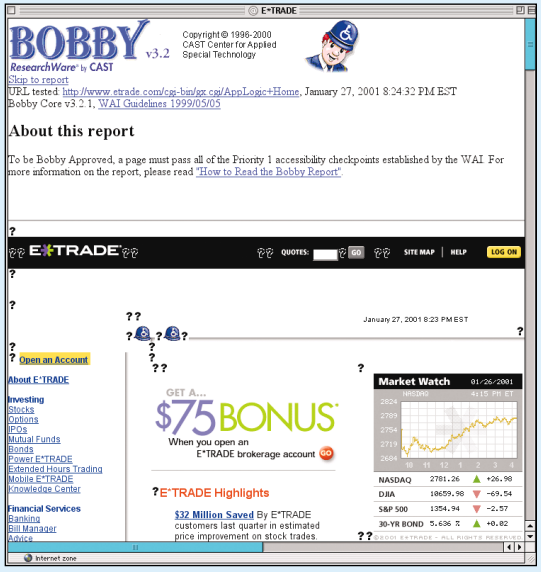
Bobby (www.cast.org/bobby) is an online tool that rates your web page immediately. Enter an URL, and Bobby identifies the areas that are not accessibility compliant and will let you know if your images have proper ALT tags. It's fast and impressive; the results may surprise you ([6.1] and [6.2]).

The Section 508 Accessibility Suite for Dreamweaver 4 and Dreamweaver UltraDev 4, created by UsableNet [6.3], enables web sites to be checked for accessibility in the same way that a document is spell-checked. The extension, available for free on the Macromedia Exchange (www.usablenet.com/macromedia/index.htm), helps ensure that web content meets Section 508 and Level 1 W3C/WAI guidelines. Reports can be run on one page, a complete site, selected sections, or any folder.



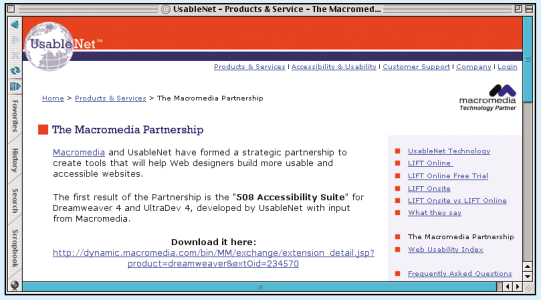
< 6.1

A Bobby Approved icon appears when a site meets all requirements for disability standards. When a site does not meet the standards, Bobby clearly does not approve and lists the site's errors as well as suggestions for improvement.



< 6.2

This screenshot shows the results of running an URL through Bobby. The question marks show areas that either are noncompliant to ADA standards for accessibility or could be improved upon.



< 6.3

Usable.net and Macromedia team up to help check for accessibility.

Check Design Status

Have the graphic templates been finalized, approved, and turned over to production yet? If not, light a fire under the chairs of your visual designers. They are holding up production. Have a delivery schedule set up so that graphic files can be handed off in a phased process: the home page and a representative subpage first, and then let production figure out the HTML templates before the remaining pages are delivered.

During Phase 3, the visual design team met with production to ensure that the designer's vision could be feasibly carried out through Flash, DHTML, JavaScript, and/or straight HTML. By the first delivery of graphic templates from the visual designers to production, certain issues such as projected K-size download and potential optimization hiccups should be resolved.

Slippage and Consequences

Content will be late; this is predictable. Anticipate it. But what do you do when that magic date has passed, the content officially becomes late, and your production is compromised? After a few gentle reminders via phone and/or email, send a firm yet diplomatic email restating due dates, details, and the costs associated for each day the content is further delayed. What follows is an excerpt of a letter that addressed slippage as it was happening, and spelled out the consequences.

> "… for clarification, we have determined some associated costs for the addition of the animated product demo and also for additional production work if our content delivery deadlines slip further.
>
> "We realize you have tight budget constraints and do not wish to incur extra charges unless absolutely necessary. As explained earlier, we have allocated resources for a particular timeframe in order to produce and complete your project, and this time window is quickly evaporating… "

Furthermore, financial consequences were clearly outlined: For each day until the final content was delivered in full, a rate was applied for "holding" resources. The effect was dramatic. The first part of the content was delivered by the end of the week, and the rest of the project ran smoothly through launch. Should you find yourself seeing deadlines slip by, consider incorporating this slippage and consequences terminology into your workflow.

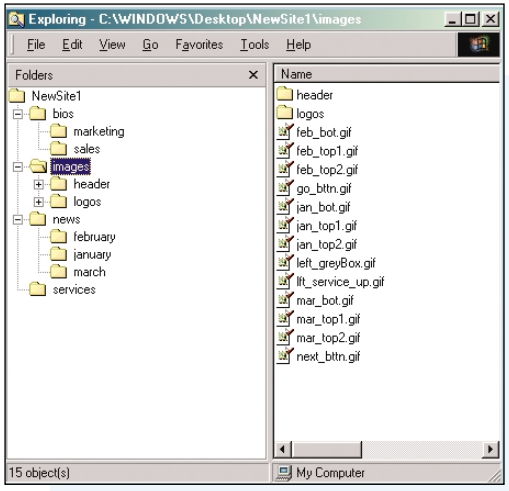
Confirm the Backend Integration Plan

Is your redesign a static site or a dynamic site? If static, and you are not involved with a backend engineering team, this section does not apply to you. If the site is dynamic, however, plan on having a meeting before production actually starts, a front-end and backend status update. Restate all technical specifications to all team members, review the technical requirements, confirm the integration plan, and clarify responsibilities.

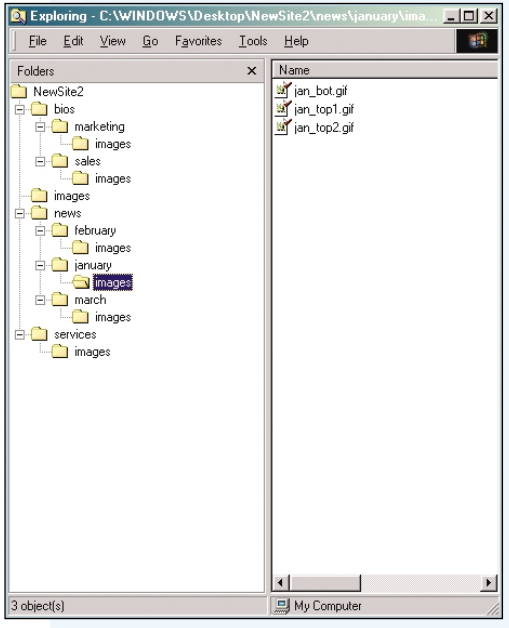
SETTING FILE STRUCTURE

Often confused with site architecture (Phase 2: Developing Site Structure) by newbies and clients with just enough knowledge to make them dangerous, file structure is, in fact, simple — but important — housekeeping. Starting out organized will help you stay organized, so make it a priority. (This is especially true for projects with multiple team members.) Although there is no best way to organize a site's file structure, different strategies support different goals ([6.4] and [6.5]).

The Client Spec Sheet asks about redesign specifics regarding existing HTML page-naming conventions and the existing file structure. Does the client want to leave things as they are, and if so, why? Whichever method is eventually decided on, it should be aligned with redesign and maintenance goals (such as how the site plans to add and archive post-launch content).



6.4 >



6.5 >

Two structures, different strategies. [6.4] has images listed at the root level, and [6.5] lists images within the current month folder. When to use which strategy is totally dependent upon preference.

Three File Structure Questions

1. How is the folder structure currently set up, and is there a client-desired reason for this method?
2. Does the folder structure follow the content structure in organization of the content?
3. Will the images be at the root level or separated into individual folders?

Redesigning offers an opportunity to start over clean. Chances are, the HTML structure of the old site is a mess: files duplicated, images scattered among folders, old versions of files still up on the server… Establish a logical, maintainable file structure for the redesign site. The goal? To start out as clean, organized, and scalable as possible.

File Structure and Scalability

How much growth (increased traffic, added content, new products) is anticipated in the 12 months following launch? Are you planning to add additional sections? How do you see them growing? By date? By topic?

When determining the file structure, know that it depends largely on how the client envisions the redesigned site growing and evolving. The plan you adopt for your file structure must be aligned with the anticipated maintenance, including logical archival of outdated content. Create subdirectories that will make sense to the maintenance team after launch and include file directory instructions as they pertain to archived or added pages in the Style Guide. Disorganization and clutter is a regular post-launch occurrence in situations in which maintenance has been handed over to a new team. An organized file structure that anticipates growth and regular updating can help counter the almost-inevitable degradation of site organization. For setting the file structure, a few pieces of information that are essentially based on client preference should be known. For instance, will the redesign repurpose existing files and the existing file structure, or will it start from scratch? How often will updates be made? Daily? Quarterly? The Client Spec Sheet asks for this information.

The big question here is this: Does the client care? Possibly, but not likely. Does the client even understand? Maybe, but probably not. But whether dictated by the client or established by your team, the file structure should respond to and fit with the answers of the preceding questions. The goal? Be scalable. Stay organized.

QUICK REFERENCE: STATIC VS. DYNAMIC	
Static Site: Front-End Only	Pages are prebuilt in their entirety and are viewed when referenced by a browser, usually using the .html or .htm extension.
Dynamic Site: Front-End and Backend Teams	Pages are created "on the fly," usually by pulling content-specific information from various places such as from a database. The site usually contains standard HTML pages as well. Additional code (ASP, JAVA, PERL) can be added to the HTML pages to allow for dynamic content population.

SLICING AND OPTIMIZATION

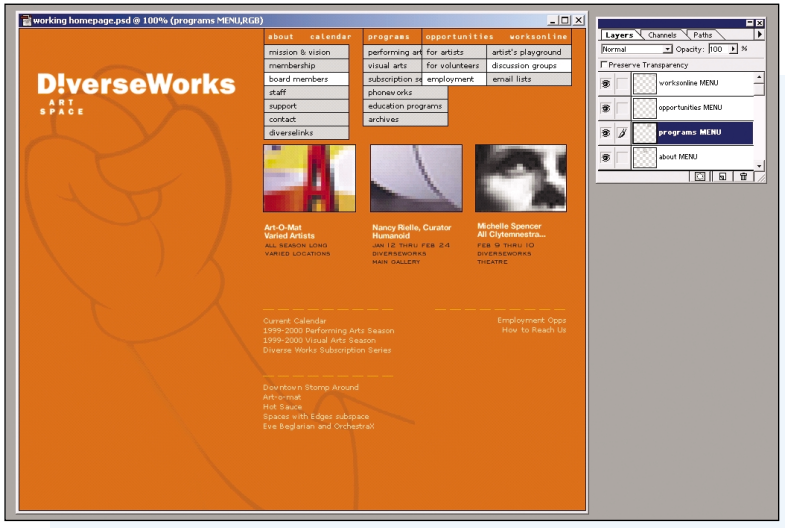
After reviewing your information (the **Prepping** part of this phase) and making sure your redesign project is on track, you are ready to start HTML production in earnest and begin **Building**. At this point in production, the graphic templates [6.6] are processed (sliced and optimized) into HTML elements (graphics) so that they can be put back together (spliced and coded).

Prior to production, during the design phase, before any visual design directions are finalized and approved, they must get checked by the production team to ensure that the files are, in fact, sliceable and optimizable under target-audience download requirements. Visual designers need to work closely with the HTML production team to determine the best way to slice the graphic templates so that the HTML tables — the rudimentary basis for HTML layout — can be constructed.

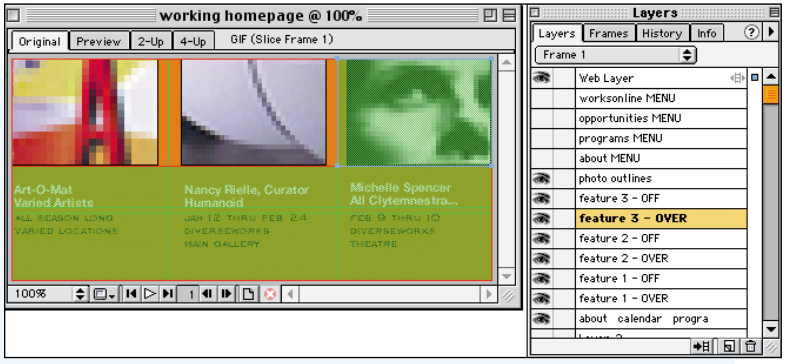
< **BUILDING**

- Slicing and Optimization
- Creating HTML Templates and Pages
- Implementing Light Scripting
- Populating Pages
- Integrating Backend Development (If Applicable)

Expert optimization should be a high priority. For an excellent how-to resource, we recommend Lynda Weinman's *Designing Web Graphics.3* (New Riders, 1999).

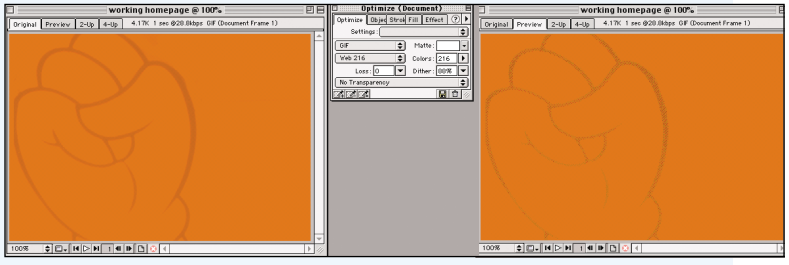


6.6 > *The graphic template for www.diverseworks.org is delivered to production from design as a layered Photoshop or Fireworks file. This file contains all the elements of the page, including all rollover states, each in their own layers. Shown here are the pull-down menu bar graphics shown in their "on" state.*



< 6.7

Graphic templates are divided into sections and sliced in either Fireworks or Photoshop. Clearly identified layers indicate on/off/over states or DHTML callouts.



< 6.8

Before and after shots of a background image being optimized in Fireworks. The file size is reduced to 16K by reducing colors in GIF format.

After the Photoshop/Fireworks files are actually handed off in a state that is producible, production does the actual slicing [6.7] and optimization [6.8] of the pieces. Note that sometimes, when budget and resources dictate, one designer may fulfill both visual design and production roles.

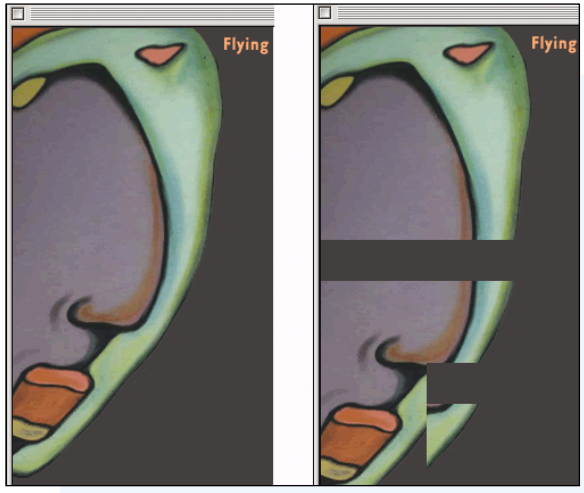
CREATING HTML TEMPLATES AND PAGES

If building a website is akin to building a house, you are currently at the point at which you have your graphics and content (your building materials) and your file directory (your house frame). Now you can build your HTML templates and the contentless pages that get saved from them (drywalling your rooms). As the templates are being created, you will want to incorporate light scripting (wiring, plumbing, and other functionality). After that step, you will be able to populate your pages (furnish your house) as you get close to launch (your housewarming party).

The first HTML template sets the standard for globals such as navigation; table structure; HTML font usage; ALT, COMMENT, and TITLE tag treatments; and so on. Take the optimized graphics that were sliced from the graphic template, add any other elements that need to be included (including any light scripting that should be incorporated; light scripting is discussed next in this chapter), and build in HTML. Test the initial HTML file(s) on various

browsers and platforms. Make sure the graphic template translates and the HTML tables don't break [6.9]. This file will be your base. If it is faulty and you don't fix it here, its errors will propagate in every page saved from it. Note that this testing is not considered QA per se; it is simply standard procedure for the production designer to check for errors.

Save from this initial template to create a page — the first of many. This new page (no longer a template) becomes a designated page within the site and is ready to be populated with content, whether static or dynamic. These pages now can be linked and tested.



6.9 >
Large graphic elements often get sliced into pieces for easier download. Beware of tables breaking. Shown here: www.flyingsparkfurniture.com with a workable table (on the left) and during pre-QA troubleshooting.

A Few Definitions for the Uninitiated

graphic template *n.* A layered, digital file (usually Photoshop or Fireworks) containing unrendered, editable text, built by a visual designer, that clearly indicates all information necessary for producing the design in HTML. Once a graphic template is sliced, optimized, and coded, it becomes an HTML page.

HTML template (also called **HTML shell**) *n.* An HTML page containing no page-specific content, built by the production designer by splicing together all the elements that were sliced and optimized from the graphic template. Visually matches the graphic template. (Utilized by production to create further files using the Save As command.)

optimize *v.* 1. To compress an image or code into as small a file size as possible to minimize download time. Usually saved in GIF or JPG format. 2. To webify, to make web ready.

slice *v.* To separate a graphic template (or a portion of a template) into two or more images (usually either GIF or JPG). *n.* A sectioned-off area of the Photoshop or Fireworks file designated to be a single image (usually either GIF or JPG).

splice *v.* To reassemble GIF and/or JPG images in a seamless manner using HTML so that the file, when viewed on a browser, looks like the original graphic template.

Version Control

Make sure it is very clear to anyone with access to the active HTML files when a file is being worked on. This is, obviously, meant to prevent two or more team members from working on a file at the same time. Such miscommunication usually results in wasted time, overwritten files, and lost work. If you have two or more people working on the HTML, having an established method of version control in place facilitates efficiency. Dreamweaver 4 includes a handy feature that allows people to check in and check out files. Third-party programs such as SourceSafe, Perforce, and WebDAV might be appropriate for your workflow as well.

As you build the HTML templates on which your pages will be based, the production designers should take care to adhere to visual standards established during the design phase and written into the Style Guide (see the end of Phase 3). In anticipation of the QA testing you will need to conduct later in this phase (the **Testing** part of this phase), keep checking your work against all browsers and on both the Macs you might be working on and the PCs most of your audience will use.

Including Includes

Is one of the reasons your site is being redesigned that it became cumbersome to upkeep? Sometimes a simple maintenance need like updating a copyright footer turns into such a mammoth job (as it would be on a site with hundreds of pages) that the updating task often goes undone. When building, you sometimes find yourself repeating things: bits of code, headers and footers, and so on across the entire site or at least on a majority of pages. Using the common example previously mentioned of a copyright footer, how do you change the year on every page? You can do one of the following:

1. Hand open each page (time consuming) and edit each. Reupload each page.
2. Do a global search-and-replace using an HTML editor. (This assumes that there are no variations and that your original text is all the same.) Reupload each page.
3. Build an include. Reupload a single page.

An include (noun, not verb) is a chunk of text coded and stored separately but applied globally so it can be edited in one place. A JavaScript include is a repeating functionality. Rather than plugging the repeating code into every page, simply reference an external file that is saved on the server separately from the HTML page. No complicated nested frames necessary. An include is an src property (source indicator), which is not so different from an IMG tag (image indicator). It is even almost dynamic in this way, except that includes don't require a backend database. Note that using includes somewhat slows loading. Assess your priorities: Does ease of updating outweigh the quarter-second of loading? Probably yes, but it's worth the evaluation. Regardless, including an include streamlines future production and is a great feature for redesign projects for which upkeep was a challenge on the old site.

IMPLEMENTING LIGHT SCRIPTING

Rollovers, forms, pull-down menus, pop-up windows, image swaps, frames… all are the result of light scripting. By "light," we mean essentially do-it-yourself or basic JavaScript that requires very little understanding of complex programming. Light scripting should not be confused with anything like JAVA, ASP, or CGI. Rather, it is standard functionality that appears in site after site, such as rollovers **[6.10]**. It is code that can be "lifted," or shared, and slightly modified to fit to your site's needs.

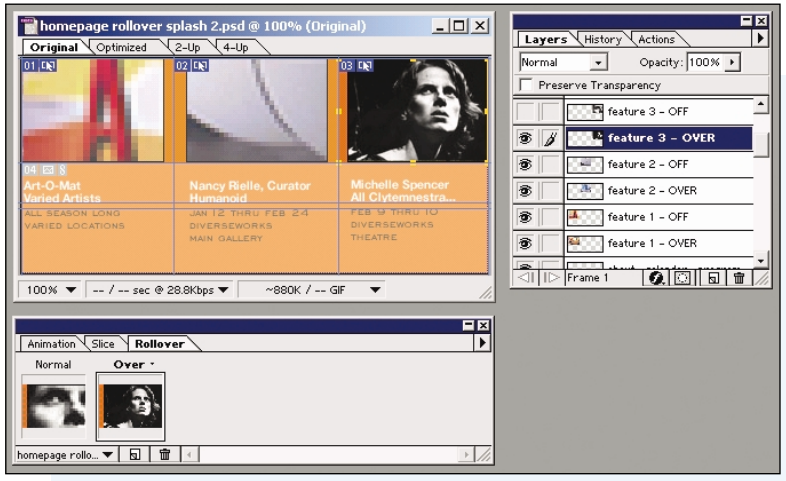
As software improves, implementing light scripting and special features (such as media requiring plug-ins) gets easier and easier. This should come as no surprise. If you used Fireworks to slice out and optimize your graphics templates, you were able to attach simple behaviors such as MouseOvers and SwapImages as you optimized and exported. If you exported an HTML file of the graphics template, much of your light scripting is already done.

Include all light scripting at this point in your workflow. Add browser sniffers and redirects. Drop in QuickTime or Flash files. Test all added functionality against specified browsers and against your audience's capabilities. Test, test, test. Look for errors. Yes, there is still QA coming up, but don't wait until then to catch bugs.

Resources abound for code sharing. "Lift" JavaScript from any of these sites: www.javascript.com, www.builder.com, developer.netscape.com, www.scriptworld.com.

Smaller Is Better

Yes, you are responsible for maintaining the K-size for the file. This includes more than just the K-size of the graphics involved; make sure to take note of HTML K-size (source code) and any outside programming. You have a target, and now you have a "finished" piece. Too big? Go back and reoptimize. See if you can shave off a few bytes here and there. Make some decisions. Adjust.



6.10 > *On the home page of www.diverseworks.org, the rollover is a dithered close-up shot of artwork that rolls over to become sharp. ON and OFF states for the rollover are specified in the layers palette of the graphic template.*

JEFFREY ZELDMAN ON WEB STANDARDS

Write once, publish everywhere. That's the goal.

To achieve it, the Web Standards Project (www.webstandards.org) has called on browser makers to support a core group of standards. These standards have several names (CSS, HTML 4, XML, and so on), but they all support a very basic idea: the separation of style from content.

What does this mean? It means your design lives in one place (for instance, in a Cascading Style Sheet [CSS]), and your content lives elsewhere (for instance, in HTML or XHTML documents, or in a database of XML-formatted text entries).

Why would web designers want this? Why would we want to separate our design from our data? For one thing, if the template for an entire site (or a section of a site) lives in a single CSS document, redesigns are a piece of cake. Need to change your background image, color scheme, margin widths, text size, fonts, and/or leading? Edit one CSS document, and an entire site (or section) instantly changes to reflect the new design. Try doing that with traditional "HTML-as-design-tool" markup. You can't. Even with sophisticated HTML editors, you're looking at hours or days of monkeywork, not to mention additional hours of browser-specific testing and debugging.

For another thing, if you can separate your design from your data, then people using non-traditional browsers will no longer be barred from your site. Whether they're on web-enabled cell phones, Palm Pilots, nongraphical browsers like Lynx, or special browsers to accommodate a physical disability, they will now enjoy full access to the site's content. With the separation of style from content, you don't have to create alternate versions of entire sites to support these folks (an expensive and time-consuming process in its own right); you simply add a rule or two to your style sheet.

With full support for web standards that facilitate the true separation of style from content, our jobs will get easier, mindless and repetitive tasks will be greatly lessened, and larger audiences will be able to access our sites with fewer problems. Instead of wasting our time and our clients' money on alternate

versions and cumbersome hacks and workarounds, we can spend it on richer content, enhanced design, and additional functionality.

I creative-direct A List Apart, a weekly online magazine for people who make websites (www.alistapart.com). Because I control the site — it's not something I handed off to a client and forgot about — and because I update the content each week, I am also constantly upgrading the site's design and user flow in small and large ways. It's an ordeal because my style and content are tragically yoked together in a manner that is practically unsustainable. I use CSS to control typography, but because of current browser limitations (especially in 4.0 browsers), I still abuse HTML tables to control the layout. So any design change, even the smallest, takes hours.

When I finally get that page where I want it, can I automatically upgrade the rest of the site to work like that page works? No, because each page is a nest of painfully interdependent, hand-coded table cells. It's too complex for global search-and-replace. And I have no production budget (the site is independent and noncommercial). So there's an archeological effect as you delve back into older issues: The design is always subtly worse than the issue you've just read. A redesign downward, as it were. That may be interesting as a historical curiosity, but site-wide, consistent branding and user flow go right out the window.

Within the next 18 months, if standards compliance improves across all browsers and users upgrade, I'll redo the layout entirely in CSS, which will enable future redesigns to take minutes instead of hours — giving me more time to spend developing site features and cultivating guest authors. I'm currently trying to make changes like that, but with 10 percent of our readers using Netscape 4 and another 25 percent sporting IE4, I can't really move.

I've described a simple, content-based site. Imagine bigger and more interactive sites, liberated by standards like XML, CSS, and the DOM. Imagine one team redesigning while another implements new functions without either team worrying that they're canceling each other's work. It's going to be amazing, but we're not there yet.

Jeffrey Zeldman, (www.zeldman.com), the author of Taking Your Talent to the Web *(New Riders, 2001), is also the publisher and creative director of A List Apart, a weekly magazine "For People Who Make Websites"; co-founder and current group leader of The Web Standards Project, a grassroots coalition fighting for standards on the web; and founder of Happy Cog, the New York City web agency least likely to go public. In his free time, Zeldman, a popular speaker at web conferences, writes columns for Adobe Web Center, PDN-Pix Magazine, and Creativity.*

Cascading Style Sheets and DHTML

DHTML is JavaScript combined with Cascading Style Sheets (CSS) to manipulate HTML. It allows for multiple levels of HTML that can be put into layers and independently controlled. When using CSS, you define a set of attributes, apply a name, and then reference that name. If you want to change every header to a different color, CSS makes the task much quicker. Unfortunately, 3.0 browsers don't support CSS, and some 4.0s have difficulty with it, too. For the time being, if you want to use CSS, you need to do twice the work because you will usually need to create two side-by-side sites or suffer degradation of design as users view your site on browsers that don't support CSS.

Hand Coding vs. WYSIWYG (What You See Is What You Get)

They say hand coding is a lost art… or is it? Many projects require the knowledge and flexibility that comes with an advanced level of HTML expertise. For many of these projects, the HTML production designers create code one tag at a time — called "hand coding" — using programs such as BBEdit or a hybrid such as Homesite. Hand coding almost always results in "cleaner" code than WYSIWYG editors generate. And as HTML purists tend to be adamant about the crispness of their code, many coders avoid WYSIWYG editors not only because they tend to add extra and sometimes cumbersome proprietary code, but also because WYSIWYGs often don't allow tweaking to as fine a level as can be achieved by hand.

With recent versions, WYSIWYG editors have enabled individuals who are not HTML savvy (designers and non-technical team members) to create HTML pages with drag-and-drop ease. Adobe GoLive and Macromedia Dreamweaver, the two industry-standard WYSIWYG editors, are each making huge strides to offer more than just an easy-to-use interface. One of the biggest advantages of WYSIWYG editors is saving time. Hand coding can be a tedious and lengthy process.

Even though WYSIWYGs have their downsides — most notably the extra source code — these applications are excellent for getting started in web design and are definitely appropriate for a large percentage of projects. But learn the HTML, too. You will be better able to tackle any development challenge.

POPULATING PAGES

Your content is due. Chances are, some content is in, some is late, and some is still being changed. But with your HTML templates completed and your pages built out, if you don't have content to work with at this point, your production designers will be idling… on the clock.

Anticipate this moment. Before the deadline actually comes, email the person who is responsible for content delivery and let that person know that the content-submission deadline is imminent. Make it clear that as of a certain date, content will be frozen, period. Frozen means no longer changeable. Final. If you do not do this, content will continue to trickle in, and content that comes in after the freeze constitutes Scope Creep; you can charge for it (see the "Slippage and Consequences" sidebar earlier in this chapter). Be aware that content will still come in even after you officially freeze it. Trickle happens. Build a cushion into your freeze date if you can.

Once you begin to populate your pages, make sure the content goes into the correct places. But who can remember (or intuitively know) where content goes? Someone, probably the project manager, has been receiving the content from the client. With this person as development-side content coordinator, use the Content Delivery Plan as a checklist, rely on the naming conventions of the web-ready files that were delivered, or develop another method of ensuring proper content placement. Whatever the plan, communicate the content-tracking plan to all production designers involved in populating the pages. Make certain nothing gets missed or placed in the wrong spot.

As you place the content, pay attention to both the layout and the HTML text style standards set by the visual designers. Be on the lookout for content that was not anticipated and therefore has no standard. Contact the visual designers and ask them to define the standard right away. Likewise, if you come across headers that need to be graphic images, make sure to alert the visual designers. Sometimes there is a template for the headers. If so, production can simply create what is needed without involving the visual designers.

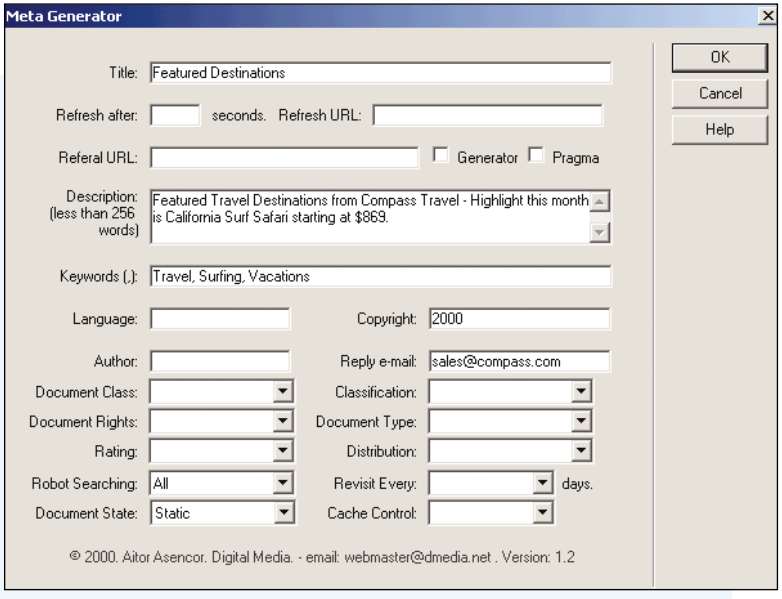
Invisible Content

Populating your pages involves *all* content, including the frequently forgotten, production-specific "invisible content" — ALT, META, and TITLE tags. Some invisible content, like ALT tags on graphic globals such as navigational elements, should be added at the HTML template creation stage so that it only needs to be done once. Others, like TITLES, should be included when pages are built from those templates. Invisible content is regularly left until the very end or is flat out forgotten. Keep it in your workflow.

Content Buckets

Dynamic sites often have designated areas, or "content buckets," where dynamically generated content (for example, "Today's Top News" or a database-built shopping list) gets placed. This is usually an HTML placeholder built into your page that will get filled by dynamic content. Content buckets need separate consideration because they are points of integration between backend and front-end.

If your site is not dynamically driven but you have an area where content regularly changes, make certain you specify clearly in your HTML Style Guide how to properly update that area.



< 6.11

Dreamweaver 4's Meta Generator screen helps make the creation and implementation of META tags streamlined and part of the workflow.

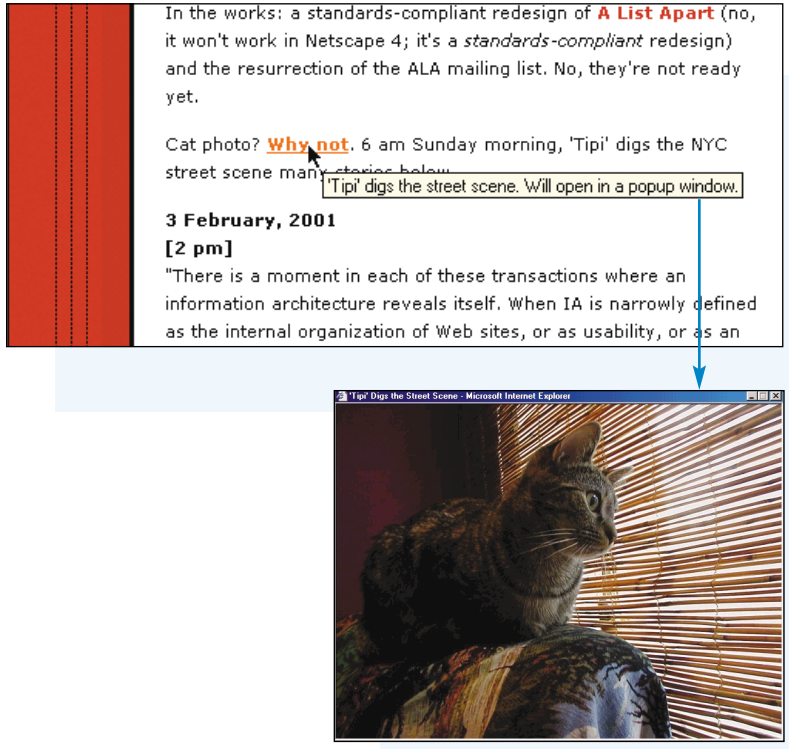
Make sure the invisible content is ready to go before the coding process begins. Few things are more frustrating than starting to work with a page or section and then having to go back and back again to fill in blanks. Know what the blanks are and how to fill them *before* you begin. Dreamweaver 4 has a handy form that asks you questions up front before you start each page. Copy this form [6.11] for your client to fill out as part of content delivery.

The client may not have every last item or image ALT tagged, and that is okay. But ALT tags can go a long way to add clarity, further define a word, or work with light functionality [6.12]. Regardless, as long as a naming convention or style is established, the production designers can move forward.

INTEGRATING BACKEND DEVELOPMENT

Communication between the backend development team and the front-end design and production teams has always been important, but at this point in the web development process, it becomes absolutely crucial. Suffering a lack of consistent communication is an exceptionally easy trap for any project to fall into, especially because some backend engineering can take months while the front-end is usually measured in weeks.

The logical place in the workflow for backend and front-end to integrate is during or right after all of the HTML pages are complete. At the beginning of the production phase, however, gather all front-end production and backend engineers, and work out a plan for integration and communication. What is the best way to create the HTML templates so that they can be handed off to the backend team for dynamic content population and programming? How much programming should be done in the HTML stage? How much experience with programming do the HTML production designers have? Which team will be responsible for inserting the actual backend code into the HTML pages? What is the timeline for integration? Who will be doing what to the templates from each team? A typical meeting will require the project managers or leads from both teams to meet. Key members from the development team should also be present, including the information designer and the art director. The technical specifications document and the Client Spec Sheet should be pulled out and reviewed by everyone.



6.12 >
ALT tags can further define a particular link, eliminate guessing, and/or help the user make a decision to click or not to click. Here, www.zeldman.com's simple text link uses the ALT tag to quirkily identify what clicking will open: a pop-up window of Tipi the cat.

UNDERSTANDING QUALITY ASSURANCE TESTING

You've built your site; now make sure it works. Quality assurance (QA) is one of the most often skipped steps (besides usability testing) in the development process. Not surprisingly, we highly recommend *against* skimping on QA. Broadway productions wouldn't go live without a full dress rehearsal with sound and lighting in place; you shouldn't launch a site without a comprehensive run-through, either.

We recommend shooting for a QA budget of approximately 10 percent of your total time and resources. You need this time to track and fix mistakes such as spelling errors, orphaned and rogue links, misplaced content, and so on [6.13]. But the even bigger job is bug tracking and fixing: broken tables, functional errors, browser crashes, everything that is not up to specifications. You then need the time to fix those bugs and then to crosscheck once again before the site goes live. And, if you have access to the client server, you will want to QA immediately post-launch as well.

However, in many projects, there is seldom time left in the budget for QA testing. More often than not, the testing and acceptance of production are slammed right up against launch. All too often, production deadlines have been pushed (usually due to late-arriving content and technical snafus), and the time allotment for QA is compromised. The extent

The QA Lead

In Phase 1, we outlined various roles and responsibilities, one of which was that of the QA lead. Depending on the size of the project or the extent of your development team, you may not have the luxury of having a dedicated individual assigned to overseeing and managing quality assurance. If this is the case, chances are the project manager will have to fill this role. For project managers new to this role, we recommend a crash course in QA — some expertise in the testing and launch of any product is far more valuable than "winging it." For an excellent overview of QA principles and philosophy, go to www.philosophe.com.

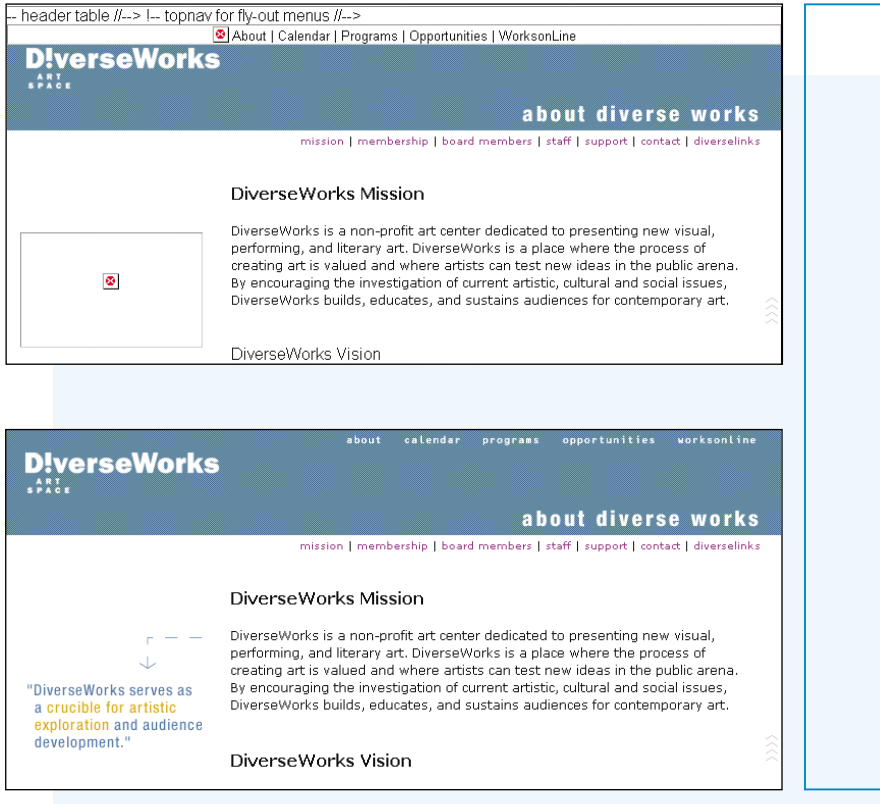
If you are managing this task, make sure to keep client expectations in line. Educate your client as to the value of comprehensive QA and to the extent and the cost that QA can take. Make sure the client understands that "comprehensive" calls for more than one day and more than just a few thousand dollars.

to which you will actually be able to conduct QA will depend largely on three things: 1) how close you are to your launch date — usually a result of how well you were able to adhere to your schedule, 2) acceptance criteria, or how perfect the site needs to be prior to launch, and 3) how flexible, if at all, the launch date is.

This critical testing process can take place informally with just a few team members, or it can be a larger undertaking, done either in-house or by hiring an outside company or team. The real-world tendency is to approach this process haphazardly, but be forewarned: Without a cohesive QA plan, you are taking a big chance. And chance is never a good step to stand on, not when there is a budget at risk. Have a plan.

CREATING A QA PLAN

You have known since the beginning of your redesign project that you would need to QA your site and that you would need a plan for it. Chances are, however, the extent of your QA plan is a budgetary/scheduling line that looks something like this: QA = 12 hours. Or 5 hours. Or 20+ hours. That budgetary line depends on the scope of your project, client expectations, and the expertise of your team. Reassess your QA plan. Keep in mind that complicated frame sets, intricate HTML templates, light scripting, and links all need to be QA tested. There



6.13 >

A typical, simple bug: The image isn't loading (top). A quick directory check and reupload of the image solved the problem (bottom). An example of a bigger bug would be a DHTML pull-down menu that crashes certain browsers (this is harder to get screenshots of).

Quality assurance testing can employ several procedures, most of which typically are used both in software development and for testing websites and web applications. In all testing situations, the extent of testing varies widely depending on technical complexity and the detail of the test plan.

BASIC/STANDARD TESTING PROCEDURES	
Smoke Testing	Testing without a formal test plan, smoke testing is also called "ad hoc" or "guerilla testing." Often, due to time and resources, this is the only type of testing conducted prior to launch.
Alpha Testing	Also referred to as "internal testing," alpha testing is the initial testing of a site after the production and functionality are in place but prior to public display.
Usability Testing	An analysis of a user interacting with the site's interface through task-oriented actions, usability testing determines a site's ease of use through observation. (For more on this topic, see Chapter 8: Testing for Usability.)
User Acceptance	Usually performed through a number of specific tests, user acceptance is dependent on scope, budget, and expertise. User acceptance verifies customer requirements (platform, browser, operating system, connection speed, and so on).
Content Check	The content check confirms content placement (not just copy — check also for image utilization and positioning), spelling, and syntax.
Beta Testing	A final check to confirm that all is functioning as intended prior to the actual launch of a site, beta testing is generally performed on the client staging site or in a subdirectory on the live server.
ADVANCED/FORMAL TESTING PROCEDURES	
Load Testing	Also called "stress testing," load testing utilizes software programs that simulate multiple users hitting the site simultaneously to determine a server's breaking point. (Costs vary widely; research is necessary to determine needs.)
Functional Testing	Also known as "black box" testing, functional testing confirms actual functionality against the specification document. Specific setup involves the person testing the functionality having knowledge of the intended outcome, but not the programming details.
Unit Tests	A test of individual components on a web page to make sure they function as specified, unit tests are verifications conducted before the code is submitted for integration of intended versus actual functionality and response.
Regression Testing	Also known by the simple name "retesting," regression testing confirms that all tracked bugs have been fixed, that the old code is still working as intended, and that no new problems were created due to said fixes. Note: The level of regression testing and confirmation varies widely.
Security Testing	A check that confirms that database and transactional information is secure from unauthorized users or hackers; a security test usually involves inside understanding the server setup.

are essentially three levels of QA: light/informal, semiformal, and formal. Make the decision as to the level of QA your project requires.

A core plan for running quality assurance shows resources, time allotted, the extent of QA expectations, who is involved, criteria for acceptance, and what the development team and the client are each responsible for prior to site launch. Running QA should involve, at the very least, two complete run-throughs: first to generate a comprehensive bug list and second to go back over that bug list and make certain that the cited bugs have been fixed. For informal QA, this basic plan should suffice. For semiformal and formal plans, this core plan is expanded on accordingly.

Every test plan or testing situation will contain different criteria for acceptance. Each site will need to check functionality against requirements and across browsers, platforms, and operating systems, from simple pop-up windows and submission of forms to complex login procedures and e-commerce ordering systems. As the web continues to evolve from basic HTML to a functional, application-driven environment, more and more attention needs to be allocated to ensuring integration success.

Test Usability During QA

QA testing and usability testing are similar in approach and scope but different in expertise and goal. At times, however, the two overlap, especially when technical errors and complications (checked for during QA) affect a user's ability to move successfully through a site (checked for through usability testing). In fact, usability testing can sometimes be considered a type of QA.

While you are QA testing your site for errors, technical glitches, and cross-browser compatibility, we strongly suggest you also conduct one-on-one usability testing (also called "verification testing" at this stage). Why? To ensure that your site works from the user's point of view. Naming and labeling must be clear. Navigation must be intuitive and easy to follow. Your site might be clean and free from bugs, but if it isn't easy to use, the chances are it won't get used and will fail.

Conversely, the redesign might be easy to use (congratulations!), but if you have broken links and spelling errors, users won't get very far. Moreover, they will have a poor impression of the site and the company. Make a bug-free and user-friendly site your prelaunch goal. We recommend *both* QA and usability testing prior to launch. For more on usability testing, see Chapter 8.

QA & Servers

Prior to a site going live, the production team should test on both the staging/development server and then again when the site is moved over to the actual server environment where eventually it will be live. When the site is moved over, the testing environment needs to be exactly the same as the live environment. This means that the folders, file structure, and server-side scripts must be correctly in place; otherwise, many of the scripts and CGI elements may not work properly.

The Problem With Frames

If your site contains frames, expect QA to take at least twice as long. Nested frames? Even longer. As a rule, the more frames you have, the more QA is needed. Moreover, frames thwart search engines (see Phase 5: Launch and Beyond). Frames, while appropriate and good for some situations (for example, portfolios, maintaining several levels of navigation, and so on), are so problematic that most often they are simply not worth it. We recommend no frames unless absolutely necessary.

Include the Client

For informal testing, clients should also participate in the QA process in the same fashion as the team members: checking the site and submitting a sheaf of printouts with errors clearly indicated as well as browser and platform types noted. For any level of testing, the client should proof the content. Only the client will be able to truly know if content is in the wrong place or is incorrect. The client should be treated as (and should hopefully act as) a partner and not a finger pointer.

Informal testing is very basic and is doable by the development team. Formal usually entails hiring an outside, trained team. Semiformal is, logically, in between the two. Most sites with a development budget under $30,000 can usually get away with informal testing. Sites with complex functionality and an application layer normally include formal or at least semiformal testing in their workflow.

Light/Informal QA

For informal QA processes, the QA lead or the project manager coordinates and tracks all planned tests and assigns team members to sections of the site, individual browsers, browser versions, and platforms. The assigned team member then goes through the site and compiles and lists all bugs for the HTML production team to fix. An easy way of doing this involves printing out pages that have errors and clearly indicating each error on the printout. Note that these printouts are only complete and

A Core QA Plan

- Summary of overall goals for QA including methodology, schedule, and resource allocation.
- List of specific browsers, platforms, and operating systems being tested.
- List of desired connection speeds being tested.
- List of any specific paths or functions that need to be tested.
- A plan for bug tracking (using a web-based program or Excel spreadsheet or printouts).
- A plan for confirming that fixes have been made prior to launch.
- Any stated assumptions (known risks) to protect the team if all fixes cannot be caught prior to launch. These should be listed in the Details and Assumptions section (in Phase 1) of the project plan or contract and be signed off on prior to the final site being delivered or launched.
- A plan for fixing bugs that cannot be resolved prior to launch. Who is to handle them, how will any additional costs be identified, and so on.

helpful if the browser and platform is noted on the printout that notes the bug. Without knowing the browser and platform, it is difficult to re-create the error and therefore fix it.

The project manager also tracks the "bug list," which, in informal testing, is really no more than a stack of printouts with bugs noted. A big red checkmark through the noted bug indicates that it has been addressed, and an accompanying initial indicating "Fixed" or "Deferred" with a date helps track the fixes.

Usually, for small- to medium-size sites (under $30,000 budgets) with very little technical complexity, this informal process is a perfectly adequate method. Informal testing is also referred to as "ad hoc" or "guerilla testing" in that it has no formal test plan or approach. Testers are just "banging" on the site, looking for bugs to slay.

Semiformal QA

If your project requires more than "guerilla testing," yet your budget will not accommodate formal testing with an outside company, the perfect middle ground is semiformal testing. Stepping up from informal to semiformal testing involves more time, expertise, and planning — and if possible, the addition of a trained QA lead and a test bed setup. A semiformal test plan should contain a one- to two-page overview that highlights the scope, timing, and goals of the QA testing process.

Test Beds

The bank of computers (set up in the testing area) that reflects the target browsers, platforms, and connection speeds of the audience is often called a "test bed." It is difficult to list every combination of browser and platform; at least use the main ones [6.14]. Even testing a smaller, representative group will result in catching many errors on the site. Test beds are common for semiformal and formal QA. Often for informal testing, the various browsers and platforms are not in the same location.

	NET 6.x	NET 4.x	NET 3.x	IE 5.0	IE 4.0	IE 3.0	AOL 4.0	AOL 3.0
MAC OS9	X	X		X	X		X	
WIN 2000	X	X		X	X		X	
WIN NT	X	X		X	X		X	
UNIX								

6.14 > *A chart like this one will help track all of the platform/browser configurations of the target audience. It can reflect the test bed setup. This sample audience does not include users on 3.0 browsers or UNIX platforms.*

Bug Tracking Tools

Although you cannot substitute automated software systems for actual QA testing with humans, there are many available tools that can aid in the process. For complete HTML validator testing, links, spelling, load time, and more, try www.netmechanic.com. Fees range from $35 to $200 for testing up to 400 HTML pages.

Other online tools? They are plentiful. Try www.scrubtheweb.com to help check your META information. www.w3.org/People/Raggett/tidy will help you clean up your HTML. For an excellent bug-tracking tool, visit www.alumni.caltech.edu/~dank/gnats.html. Want to learn more about bugs? Go to www.mozilla.org/bugs. Mozilla itself is handy for QA as well.

Formal QA

Planning for formal QA testing requires experience, time, budget, and most of all, attention to detail — minute detail. The biggest difference between semi-formal and formal QA is the level of test planning, the cost, the generation of documentation, and the degree of expertise.

Formal QA uses a comprehensive bug-tracking system and a fully trained QA staff (yes, staff) to test requirements and pages against specified browsers and platforms. It includes test plans, tools, use cases, a test bed, and reports. To illustrate the extensiveness of the formal testing process, consider this example of a typical formal QA plan: Identify at least 10 different paths through a site and test each path on three platforms (MAC, WIN, UNIX), with each platform hosting three browsers (IE, Netscape, AOL), with each browser having several versions (3.0 through 6.0, note that Netscape skipped 5.0) — all needing to be tested. This example now has approximately 450 different tests ($10 \times 3 \times 3 \times 5$) for the defined paths. Overwhelming? Yes. Impossible? No. Impossible in an informal setting? Yes. Recommended for large sites with a significant backend engineering and extensive functionality? Absolutely.

Bug Reporting

Reporting a bug is easy. Reporting bugs in a way that is meaningful, reproducible, detailed, and solution oriented is a challenge. Here's the old, serviceable, good-for-informal-testing way: Print the page out, note the browser/platform, circle the error, fix the bug, and then check the error as fixed (or deferred if the bug can't be fixed yet). Here's another (and maybe better) way: Use some kind of tracking device — even an Excel spreadsheet will suffice — although you can only have one person working with the file at a time. Whatever your tracking method, make certain to note the following information:

- Browser type/platform type.
- Operating system.
- Description of problem (one line).
- Detailed description.
- URL of page.
- Severity of problem.
- Can the error be reproduced?

PRIORITIZING AND FIXING BUGS

Decide what needs to be fixed immediately. These are the showstoppers — the glaring errors. Continue your list and prioritize the remainder of the fixes with headings such as showstoppers, high priority, medium priority, and low priority. Understand that some bugs may be unfixable because they are end-user dependent. If you can't re-create the bug, mark it as such. They might be due to end-user browser settings [6.15]. Depending on the time left before launch and the level of perfection that is necessary at launch, plan for prelaunch fixes and post-launch fixes alike. Postlaunch fixes should happen in an iterative fashion.

After addressing bugs, test your fixes. Try to re-create the error. Some fixes may require several tries.

6.15 >

An image glitch (top) shows up on AOL 4.0 browsers during the QA process, but randomly — only on some laptops. End-user preferences setting were to blame: Unchecking "Use compressed images" and then dumping the browser cache remedied the glitch and displayed the image correctly (bottom). End-user-caused bugs are largely out of QA control.

QA BUDGET COMPARISONS BY OVERALL PROJECT BUDGET RANGE AND TECHNICAL LEVEL

Light/Informal QA	Semiformal QA	Formal QA
For projects with budgets under $30,000, estimate the QA budget at 1% to 3% of project cost. **Technical level: light.**	For projects with budgets ranging between $30,000 and $70,000, estimate the QA budget at 5% of project cost. **Technical level: moderate.**	For projects with budgets over $70,000, estimate the QA budget at 10% to 20% of project cost. **Technical level: moderate to complex.**

NOTE: For projects at or under the $10,000 budget mark, a mere $100 worth of time and resources will not work. You do need to go through the entire site.

Showstoppers

There are bugs and then there are big bugs (sort of the difference between a harmless little earwig and a thumb-sized Palo Verde beetle). Big bugs are showstoppers — errors that simply cannot go live. These errors *have* to get fixed before launch (for example, the home page loads incorrectly, a pull-down menu crashes IE, the frame sets are mis-targeted, and so on). As you track bugs, prioritize. What are showstoppers? What can get fixed in an iterative approach in the first week of launch? Sometimes the launch date is set in stone, and you do not have the time to fix all bugs. Prioritize and slay the showstoppers first. The rest can wait a few days.

CONDUCTING A FINAL CHECK

Conduct a final check with all teams involved. Make sure all systems are go. Here are the key five items to confirm:

- **Design check.** Designers have a keen eye for detail; they might catch misalignments and incorrect graphics that a good QA team might never notice. HTML text might be placed incorrectly; a photo treatment may have been misapplied. Have the art director or designer give the site a thorough look on both Mac and PC to ensure quality control.
- **HTML check.** Confirm that all tables, cells, and graphics are lining up properly. Your team may not have had enough time for ample tweaking. After QA is in full gear and fixes are being implemented, let the HTML team check once more that the site is visually working on both MAC and PCs. Sometimes QA fixes alter/wreck code.
- **Functionality/engineering check (if applicable).** Confirm that all functionality is working in accordance with the technical specifications. Make sure that database integration is complete and that all transactions can be accomplished on the live server.
- **Content check.** Confirm that the headlines are reading as headlines, that body copy reads like body copy… you get the picture. Make sure that the content formatting was appropriately applied by the production team and that everything is lining up as expected.
- **Client approval.** Making sure the client sees and approves the entire site prior to launch might seem like an obvious check-off item. Surprisingly, it is often the case that, although the redesigned site has been signed off on by the marketing department the entire time, the CEO or advertisers who need to approve the site before it goes live may never have seen the final site. Sometimes it is appropriate to wait until the last possible moment to get the approval from the highest level; sometimes this delay causes chaos.

PHASE 4 SUMMARY

The production phase is probably the most straightforward phase in the Core Process. You actually produce and build the site. There is little room for improvisation. Production is a straight shot from start to finish: Query your client, compose the Client Spec Sheet, consult on feasibility with the visual designers and the information designer (Phases 2 and 3), build the Protosite and test functionality (Phase 3), receive the graphic templates from the visual designers, slice and optimize graphics, build HTML templates and integrate light scripting, build and populate individual pages, integrate complex functionality and backend applications and/or engineering, and test. Then breathe. Then build the HTML Style Guide and prep for handoff (Phase 5).

Why involve the production team throughout the entire process? Quite simply, without advance checking, testing, and confirming, the building phase can be risky. You may as well get off a ski lift and ride down any random run without checking its skill level. Think of the possible crises: finding out as you build the HTML templates that your pull-down menus block the contracted advertising space, or trying to slice and optimize a layout that simply does not translate to HTML. Either of these scenarios would involve chalking up as a loss the numerous hours spent coding. The team would have to backtrack, and if the launch date is firm, you might not have enough time.

Not to worry, however. The Core Process sets you up so that production can get pulled off with minimal hitches. Sure, you encountered bugs — every site has them. Be thrilled that production is done! Well, almost. Your site is built. It is logically organized and will be easy to maintain thanks to well-thought-through HTML. Your site is bug free. It is user friendly. It looks exactly as your visual designers intended. You are ready to take care of launch and what follows. If this were that pie we mentioned at the beginning of this chapter, it would be baked.

Now get ready to serve.

PHASE 4 CHECK-OFF LIST

Prepping

- ☐ Compose the Client Spec Sheet
- ☐ Assess project status

Building

- ☐ Set file structure
- ☐ Receive graphic templates from the visual designers
- ☐ Slice and optimize graphics
- ☐ Create HTML templates
- ☐ Implement light scripting
- ☐ Build individual pages
- ☐ Populate individual pages
- ☐ Include invisible content
- ☐ Integrate complex functionality and/or backend engineering
- ☐ Freeze production

Testing

- ☐ Create QA plan
- ☐ Conduct QA testing
- ☐ Prioritize and fix bugs
- ☐ Conduct final check