Boot Camp
Design Essentials for Beginners

Web Authoring Boot Camp

Get started as a kick-butt web author with must-have info on web project planning, workflow, and code

L.J. Bothell

Web Authoring Boot Camp

L.J. Bothell

kick-butt web authoring

studio bast
Seattle, WA
www.studiobast.com

Contents

Web Authoring **Boot Camp**

WHAT'S GOING ON?

section 1

Web authoring bootcamp, that's what!

In this section, we're going to find out what web authoring is, and how planning, preparation, and research can do to help you design effective and usable websites.

intro

Introduction to Web Authoring Boot Camp

Well, lookee here, it's another web design book. As if there aren't already enough of them in the bookstore and sold through Amazon. What can this book possibly offer that I want to know?

Why Another Web Design Book?

It's true, there are tons of web design, coding, authoring, production, styling, programming, analyzing, and ice-cream sundae-making books on the market. Many of them are really good, too, like anything from PeachPit Press, Sitepoint, and Sam's Teach Yourself. Get one or two of these – a how-to and a reference-oriented book, and you'll have a terrific start to educating yourself on the wonderful world of creating websites and web media. Excellent!

This book is a little different, for several reasons:

- It's free.
- It's open-source, so that it can be openly distributed and used for non-commercial education purposes.
- It's the first put-together open-source book available on web authoring (as of this writing).
- It covers not only the how-tos of building websites, but also treats the whole website project – from planning to completion.
- It looks at web authoring as a profession, a workflow, and a great career to get educated in.
- It's written by a web author who has education and experience in the field and who also gets to teach this information to students.

Who This Book is For

This book is for new web authors – those who are taking classes, teaching themselves through tutorials and videos, and anyone who wants a glimpse into the world of web authoring – as a profession as well as a workflow practice. It's for you if:

- You want to know what a web author is and does for a living.
- You want to understand the web project planning and preparation steps that happen before coding.

- You want to build usable, accessible, and intuitive HTML websites that anyone can use.
- You want to get step-by-step basics on coding practices and workflow.
- You want to launch well-designed, W3C standards-compliant websites with basic SEO practices.
- You are starting/taking classes in web design/production as career training.
- You want to build your own websites, freelance build for other clients, and/or work with creative talent agencies and employers who hire designers, authors, producers, and other web specialists.

This book is not really for seasoned web authors, although it's a terrific reference tool after you have used it to get your feet wet. Is covers the essentials of planning, preparation, designing, building, and launching of static HTML websites, and refers readers to online sources and more detailed books for advanced web authoring like stylish CSS, HTML5 and CSS3, Javascripting, and other web authoring languages and scripts. It's not for you if:

- You already have a several websites of experience under your belt.
- You already work in the web specialties field and are past beginner-stage.
- You are looking for dozens of specialized tricks, styles, and recommendations for intermediate/advanced CSS.
- You want to build dynamic websites that are server-side and database driven.
- You want detailed workflow and training on HTML5 or CSS3.

How to Use This Book

This book is pretty compact, and is more-or-less a process/workflow book. It combines a conversational tone with explanations, details, examples, and factoids. The goal is to give you what you really need to get started as a kick-butt web author, rather than every detail known to web geek – that's what the Web is for. Occasionally, I'll make a joke or ask you a question, and I expect you to chuckle and to answer. Heh – gotcha!

Each chapter introduces itself, then gets into the nitty gritty of the subject, requirements, tools, and workflow. You are offered examples of code and illustrations, plus occasional exercises to try out yourself. Have a willingness to try what you see and to explore more through other examples and tutorials on the Web.

You can start at the beginning to learn what web authoring is and how to plan web projects in Section 1. You can jump right into basic HTML coding and styling in Section 2.

You can get started creating a web page template or a basic Cascading Style Sheet. You can use the chapters and resources at the end to jumpstart problem-solving or adding bits of dynamic stuff to your sites.

This book also makes a pretty handy virtual paperweight.

Learning Tips

- Take your time. This book is meant to be used, not just read. Look at and try out the code.
- Slow down when you need to so that you don't overload your brain. Focus on a chapter section or three, take a breather, then go back and make sure what you are learning makes sense and that you can use it.
- Take the pieces of code you see and work to assemble them throughout the book as you learn the elements and workflow for HTML coding.
- Share what you learn. Talking about it, and showing someone else what you are planning and doing helps make stuff stick.
- Make sure to stay hydrated and have chocolate somewhere nearby. Learning sticks better with chocolate. Everything is better with chocolate.
- Create something! This book will help you best when you come up with a simple website idea of your own and apply the planning, preparation, design, coding, and validating steps for yourself.

1 The Web, The Code, and Everything

What is the World Wide Web?

The World Wide Web (commonly abbreviated as the "Web") is a system of interlinked hypertext documents accessed via the Internet. With a Web browser, you can view web pages that contain text, images, videos, and other media, and navigate between them using hyperlinks.

History

The World Wide Web Consortium (W3C) is the main international standards organization for the Web. This consortium develops and upgrades the standards for the web. If you don't like something, it's their fault.

Seriously, though, Tim Berners-Lee published what is considered to be the first website in August 1991. Berners-Lee was the first to combine Internet communication (which had been carrying email and the Usenet for decades) with hypertext (which had also been around for decades, but limited to browsing information stored on a single computer, such as interactive CD-ROM design).

Then

Originally meant for the free sharing of knowledge, the Web has transitioned from basic documents to the use of URLs to get to addresses, and from simple text and basic image pages to highly styled websites. It's gone from scientific and information-based to edutainment, e-commerce and promotion-based sites. It has become a primary way for us to get research, news, products, social interaction, etc.

It wasn't always this way. Back in the day, websites were written in a markup language called HTML, and early versions were very basic, only giving a website's basic structure (headings and paragraphs), and the ability to link using hypertext. This was new and different from the then existing forms of communication, since visitors couldnow easily navigate to other pages by following hyperlinks from page to page.

Now

Right now the Web is becoming our primary source of information, entertainment, education, shopping, promotion, and more. There are applications, social media, multimedia, games, and just about anything you can imagine. If it's not there, it soon will be.

Basically, the Web is a worldwide storage basket of all things info, and likely more aggregation, easier access, more personalization, more "plugging in" etc. will happen.

HTML

HTML (Hypertext Markup Language) was the original simple way to place text and images in a web page, and browsers were created to recognize the language and render it for viewing. It is based on a series of tags, which could be used without closing or doctype declarations, and simple inline text formatting styles like bold and italic. It allowed for fairly simple report-style information formatting.

As the Web and browsers progressed, more visitors gained access to information and later product marketing. They needed attractive, more easy to read web pages. Web design progressed, and the markup language became more complex and flexible, giving the ability to add objects like images and tables to a page. Features like tables, which were originally intended to display tabular information, were soon subverted for use as invisible layout structures.

The potential for marketing, entertainment, and profit on the Web pushed the boundaries of simple HTML far past where its original standards were meant to go. Visitors to web pages wanted to see more excitement, so HTML styling commenced, with inline styling of widths, heights, text formatting, image and link modifications, and more. Plus, as browsers developed, web designers created hacks and workarounds to get HTML to do their bidding. Which, it turned out, different browsers and monitor resolutions interpreted differently. A web page on one browser, like Internet Explorer, could look pretty good, while on Firefox or Safari it would look equally poor. Or vice versa.

In the 1980's and 90's, Cascading Style Sheets (CSS) were developed to extend HTML styling and allow additional standardizing information to be developed to render more complex pages. The first officially adopted version was available in 1998. Web pages could now have complex styles created and stored in a separate file, which, when updated, would cascade those changes throughout all pages on the site that referenced the CSS file. The styles could look a lot more similar between browsers, which were also becoming more standardized as new versions came out. However, as marketing, sales, entertainment, and visitor needs for attractive, dynamic, and increasingly robust web content display grew, even more had to be packed on top of HTML and CSS.

Website Building Enhancements

For more interactive-seeming pages, Flash and JavaScript have been added to the mix.

They allow for style enhancements like galleries, slideshows, banner ads, rollover navigation, and more. These make pages look alive and refreshed, but don't usually allow real interaction between the visitor and the page/page creator. These, and some AJAX, are client-side tools (you and the browser being the client, or receiver).

Client Side

Client-side scripting refers to programs on the Web that are executed by the visitor's web browser, instead of on the website owner's web server. This dynamic HTML (DHTML) enables web pages to be scripted; that is, to have different and changing content depending on visitor input, environmental conditions (such as the time of day), or other variables.

Web authors write client-side scripts in languages such as JavaScript, which are often embedded within an HTML document (embedded script), They may also be contained in a separate file, like JQuery, which is referenced by the document (or documents) that use it (external script) by links in the web page's header section.

Server Side

Server-side scripts are executed by the web server when a visitor requests something (like a login approval). They produce output in a format understandable by web browsers (usually HTML), which is then sent to the website visitor's computer. The visitor cannot see the script's source code (unless the author publishes the code separately), and may not even be aware that a script was executed. Documents produced by server-side scripts may, in turn, contain client-side scripts. The visitor's request is then fulfilled by running a script directly on the web server to generate dynamic web pages. It is usually used to provide interactive websites that interface to databases or other data stores.

Server side tools/languages have made web pages almost completely dynamic/interactive. Languages like ASP, Python, PHP, Java, C#, and Visual Basic work with information databases to build and drive dynamic websites. The visitor can input information to get specific pages rendered especially for him/her, and to store data so purchases can be made, etc. Look 'em up online to learn more.

XHTML

A lot of websites do not need to be fully dynamic, and HTML limitations for non-dynamic sites have long since been blown. Intermediate web production is all about moving past HTML to the more precise, manageable, and World Wide Consortium (W3C) validated

current standards for basic web coding. This means that with increased standardization, most browsers show the same results from code, and the code has been expanded to allow for more general style and basic tag use.

Original HTML coding was very forgiving (it didn't require a doctype, charset, namespace, closing tags, or avoidance of deprecated tags to still show the web page). HTML does require more tweaking as a website grows to work with multiple browsers and quirks. This makes web production unwieldy and not very cost or resource-effective.

As a result, in February 1999 XHTML 1.0: The Extensible HyperText Markup Language was developed. In January 2000 it was officially adopted as a W3C recommendation. It has been designed by the W3C to merge the best of HTML and XML (Extensible Mark-up Language), to work fluidly with the easy-to-organize CSS, and to be able to be parsed by every browser with as much uniformity in interpretation as possible. CSS had already been adopted for enforcement and validation in 1996.

XHTML is stricter than HTML, because it absolutely requires:

- A doctype declaration.
- Use of HTML, head, and body tags.
- All tags to be closed.
- Use of quotes around values.
- Use of all lower case elements, attributes, and predefined values.
- Use of the ALT attribute and value for all images.
- Avoidance of deprecated tags that are being phased out by the W3C.

While this can sound pretty tough, it actually is easy to learn. It also results in organized and consistent code, allows clean validation, and keeps you away from deprecated tags and quirky practices that give browsers (and you) huge headaches. HTML5, which is currently being implemented but which is not well supported by many browsers yet, keeps these expectations intact.

Web Addresses

Anyone who has searched for something on the Internet using Google or another search engine has seen the results, with hyperlinks to the related website. The address of these websites are known as URLs. The URL (Uniform Resource Locator) is the full web address: **http://www.studiobast.com.**

HTTP (Hypertext Transfer Protocol) is a protocol for distributed, collaborative, hypermedia information systems. It is used for retrieving inter-linked resources, and is integral to the universal web address: **http**://www.studiobast.com. Sometimes you'll see HTTPS, which is a secure version of the URL, such as when you have logged into a website. This tells you that you are looking at/working with hyperlinked pages.

WWW is the part of the URL address that reaches out to the World Wide Web: http://**www**.studiobast.com.

The **Domain** name refers to your domain, or the domain the Web host you use has assigned to you: http://www.**studiobast**.com, http://www.**freehostia**.yoursite.com.

The **Subdomain** name is your contribution to the URL of a site you have hosted on someone else's web host domain: http://www.freehostia.**yoursite**.com.

The **Host** name refers to where the website is hosted: http://www.studiobast.**com**, where .**com** indicates this is a commercial address. You will also see .net, .info, .edu, .gov, .ca (for canada), etc.

Web Work

Web pages have gone from a few dozen to millions, with more exploding daily. Many of these are part of larger websites – the collection of web pages making up a theme and collection of information. A whole industry in Internet Technology is made up of those who plan, design, produce, populate, test, and distribute web pages/sites. What have we really got out there in the Web specialties, besides increasingly high-fallutin' titles?

Front-end

The observable content (like page layout, visitor interface, graphics, text, audio) is known as the front-end.

The front-end of web work includes the visitor-facing stuff: the web pages, forms, graphics, general interaction, content, and so on. These things all take different skills, and people can specialize in, or mix it up in, several areas:

- **Graphic Design/Concept**: The layout/look/palette/type and CSS of websites.
- **Visitor Interface/Experience Design**: Design based on the focus of the visitor's experience and interaction, including experience testing, pre design surveys and interviews, focus group creation, and all things that make the visitor's interaction as simple and efficient as possible.

- **Interactive Design**: JavaScript, Flash, streaming media, and other multimedia and scripting client-side expertise.
- **Prototyping**: The build-out of the web concepts in a working prototype before full coding and programming happens. In designer/developer teams, this is considered the pre-build.
- **Content Management**: Providing, placing, and/or maintaining the information that makes up the content of the website, like text, images, links, animations, etc.
- **Usability**: The practice of making sure a website is accessible, has usable visitor interaction (like navigation), accessibility with other mediums like print and mobile devices, and search engine optimization.
- **Production**: The update and maintenance of websites, such as new pages, code cleanup, adding and removing content, and checking links.
- **Marketing/Promotion**: Email marketing, site listing and promotion, SEO, etc.
- **Webmaster**: Multi-skilled individuals who carries out many of/all of these tasks for a whole website.

Back-end

The back-end comprises the organization and efficiency of a website's source code, invisible scripted functions, and the server-side components that process the output from the website visitors. This requires the more programming-oriented work that makes complex interactive websites work. Does not usually include any visual design, but rather has designers and other front-end folks as part of the collaborative team. Back-end work includes:

- **Developer/Programmer**: Main non-design aspects of building websites, such as writing markup and language coding. Web development can range from developing the simplest static single page of plain text to the most complex web-based internet applications, electronic businesses, or social network services.
- **Database Administrator**: A database administrator (DBA) is a person responsible for the design, implementation, maintenance and repair of an organization's database, which drives fully interactive sites like Amazon. The role includes the development and design of database strategies, monitoring and improving database performance and capacity, and planning for future expansion requirements.
- **Analyst**: Web analytics is the measurement, collection, analysis and reporting of internet data for purposes of understanding and optimizing web usage. This provides data on the number of visitors, page views, etc to gauge the traffic and popularity trends which helps doing market research.

- **Security**: Web development takes into account many security considerations, such as data entry error checking through forms, filtering output, encryption, and security patches.
- **Project Manager**: Oversees collaborative design between group members with specialized skills.

What is Web Authoring?

For the purposes of this book, we are going to describe **web authoring** as the work involved from the beginning to the end of a web project. Web authors don't just code/build cool sites; the really employable and valued web specialists train and gain experience in the whole range of skills that are involved in designing the website concept, producing it, and testing/releasing the website for launch.

Planning

A web author has to start at the beginning, even if s/he inherits a previous web project. This includes getting to know the website client and the requirements of the project, often through client meetings, and/or a client survey. You need to set up your administrative information for organizing and billing the project, like contracts, charter, forms, communication methods, etc. Planning also includes getting clear on who the website visitors are going to be, and what they are supposed to accomplish/ Web aothors use planning tools like an AIA table, wireframes, project calendars, interviews, and various research methods. Finally, planning includes setting the timeline and budget, researching the website needs, and gathering the resources and tools needed to do the work, like the content, images, web host, etc. You can't design a website if you don't know why and for whom.

Design

The design part of web authoring goes hand-in-hand with understanding the expected website visitors, accessibility, SEO, and usable layout. Design isn't just about being cool, but is mostly about crafting a website design that the visitors can actually easily use and react to. This also includes doing the anticipated sitemap, wireframes, and approval comps. You can't produce a website until you have a clear, and client-approved design plan.

Coding

Coding the website is part of the production, and means writing the actual HTML and CSS (and related client-side script) of the website. Some web builders like to use WYSI-

WYG tools for this, where they click on icons and assemble pieces in a visual way, rather than writing the code itself. Some programs are very good with being clean code translators, while others are not.

Production

Production refers to the assembling of the website, including the coding, placing of images and media, linking, and creating clean navigation elements. It can also include the creation and/or editing of content, like received text or stuff you write, preparing images for web readiness, updating existing pages, and merging with tools and resources that the website might provide to visitors.

Pre-launch Preparation

Before you can launch a website, you have a whole bunch of considerations on your checklist: revisions, finding/contacting a web host, revisions, uploading, page testing, client-side testing, server-side testing, form testing, link testing, cross-browser comparisons, revisions, validation, revisions, and more revisions. Coffee??

Launch/Maintenance

Whew! Then you are done. Or are you? Part of your relationship with your client/web project is getting it launched and closing the project properly. Launching includes making sure the site will be indexed by search engines and that it is reachable on the Web by a URL and a visitors' search. Closing the project means knowing who will maintain the site (if you will not) and making sure you leave clean and accessible code for your replacement, as well as any necessary documentation about quirks and concerns. Finally, you need to close your administrative functions on the web project: invoice, turning over files to the client, etc.

Coding vs. WYSIWYG

Code Authoring

As mentioned above, code authoring is where you hand-code in HTML, CSS, JavaScript, and any other languages or scripts you choose. You have complete control over the output of the website when you code from scratch. WYSIWYG programs can reinterpret your visual design choices with poor code that won't be up to current W3C standards and won't validate or look right on necessary browsers. When you know your code you can correct anything, create your own templates and snippets, and generally offer more professional value to employers and clients.

WYSIWYG Building

WYSIWYG is an acronym for *What You See Is What You Get*. The term is used in computing to describe a system in which content displayed during editing appears very similar to the final output. It allows people to edit web pages without knowing HTML. This is good on the fly for quick edits, but if you plan to do this as a regular web authoring practice, you won't be a serious contender for the web specialties because employers and clients often do not have or allow use of WYSIWYG programs for their professional websites.

This book will focus on the basics, next steps, and some special tips for web authors who will hand-code. You will learn how to accurately and cleanly code original web pages, create streamlined workflow process, and leave happy website clients who will want you back for more. That's the goal, anyway. Let's get going!

2 Web Projects

First, understand that a website is a collection of web pages that hyperlink to each other, use a consistent theme, strive to serve a basic purpose, and offer cohesive content that communicates to website visitors. It could be YouTube, Amazon, your local used bookstore's website, your blog, the local community college, etc. These sites share one thing in common – they are a collection of hyperlinked pages that are mostly related to each other in subject coverage and theming.

Axiom #1: A website is a method to communicate key information to and prompt action from an audience, and all design, content, and navigation options must serve this axiom.

How do you get from here (total noob and/or developing web author) to there? We have a LOT to consider. Check it out!

Website Planning

What do you need to consider in developing a website plan? For typical website construction, the basic aspects you need to consider include:

- **The usability**: The site should be visitor-friendly, with the interface and navigation made simple and reliable.
- **The content:** The substance and information on the site should be relevant to the site and should target the audience(s) that the website is focused upon.
- **The appearance**: The graphics and text should include a consistent and clean style that flows throughout the website. The style should be professional, appealing and relevant, primarily to the target audience.
- **The visibility**: The site must also be easy to find with major search engines and advertising media.

Usability

Imagine a website where the visitor knows exactly what page s/he is on (and why), can navigate really easily and fast to other pages, and gets exactly the information / completes exactly the right task with absolutely no fuss. Sound easy? It can be.

Focusing immediately on web usability is an approach that makes websites easy to use for a visitor, without requiring them to have any specialized training. The visitor should

be able to intuitively relate the actions s/he needs to perform on the web page. The broad goal of usability is:

- Present the information to the visitor in a clear and concise way.
- Give the correct choices to the visitors, in a very obvious way.
- Remove any ambiguity regarding the consequences of an action.
- Put the most important thing in the right place so the visitor can interact as you want.

Usability is often confused with accessibility. However, it comes at a more general approach- ensuring everyone (disability, disadvantages, or totally able) is able to use the website in the most straightforward manner. This includes having things that are easy to find in prominent spots, ensuring common or repeated tasks on a website can be shortcutted, and most especially designing the GUI (graphical user interface) so that anyone can easily and intuitively "get it".

Usability requires planning for layout and design concepts, and also for the actual content organization itself. For instance, the logical flow of information, in terms of important to useful to nice to filler, helps you determine what goes above the fold on a page, how to divide with subheaders, and how much content text to include on a page.

Website Priorities

Knowing the different priorities a website may have, and remembering who the site is centered around, is important to making the site usable and ultimately successful for the client's purpose.

- **Visitor-centered**: Engaging, successful, and marketable web design comes from remembering who really uses the website, and who puts the bread and butter on the client's table – the visitor. The visitor will make the decision to stay or leave, to bookmark or forget the site, to order or move to a competitor, to come once or return again and again.
- **Client-centered**: The client naturally has important communication goals – marketing, promotion, sales, influence. These purposes do drive the nature of the site. At the same time, clients often need to be tamed to help their website achieve their purpose - *Axiom #1*: A website is a method to communicate key information to and prompt action from an audience. A web author's job often includes keeping the client's desires from getting in the way of the visitor's needs.
- **Designer-centered**: In other words – You. Designers get into this business because they are creative and need, in one way or another, to create attractive order

out of chaos. We can also get excited about trying new layouts, techniques, and content, and that's great. It can also be a problem if our design sensitivities and desire to stand out overwhelm *Axiom #1*. The website urges visitor action, and the visitor comes first.

Website Purpose

It is essential to clearly define the purpose of the web project. You can/should create a purpose statement focused on what the website will accomplish and what the visitors will get from it. A clearly defined purpose will help the rest of your planning process as you identify the audience(s) and put together the design and content of the site.

You also want to work with your client to understand both short and long term goals for the website. The short term includes version 1.0, so to speak, where you plan for the main launch and usage for the present. Longer-term planning is also important since you and the client need to account for the future when expansion, modification, and improvement will need to happen. Could be six months, could be two years, but you need to plan for a website that can progress and expand cleanly for visitors and the client's business plan needs.

Ever seen a sprawling, disjointed website with lost links and different theme interpretations? You want to discourage this by planning for the future of the website as well as the current launch.

You also need to determine what the primary aim of the website will be. What is it supposed to do for visitors? Essentially, it should aim, as defined by the client's business plan, to do one or more of the following:

- Communicate by providing information for the visitors to take away and potentially use. This could be news, class listings, a café menu, Wikipedia, etc.
- Influence by providing content that influences the visitor to act in a certain way. This can include product reviews, repair instructions, tips and tricks, an opinion blog, interior 360 view of a house on the market, etc.
- Prompt by providing content and activities to get the visitor to do something related to the website itself. This includes registration for newsletters, ordering products, paying bills, making travel reservations, going to a free band show, etc.

Website Audience

Defining the website audience(s) is perhaps the most important step in your website planning. The audience is the group of people who are expected to visit your website – the

market you are targeting. These people will be viewing the website for a specific reason, and you need to know exactly what they are looking for when they visit. You may also have more than one key audience, such as if the website targets more than one age group, or customers for different product lines.

Like the purpose, you need to clearly define your understanding of what visitors want to do or feel when they come to your site in order to help them fulfill its purpose. While there is a certain amount of guesswork, research and your discussions with your client will help you identify the target audience. Consider:

- **Audience Characteristics**: Age, location, income range, gender, education, language, and other visitor discernments.
- **Information Preferences**: Reading bulleted lists, show-and-tell, video, instructional, newsy, quick & dirty, and so on.
- **Computer Specifications**: The likely computer speed, monitor sizes, various internet connections, accessibility tools, and other things that impact how fast the visitor can get the information download and use the information.
- **Web Experience**: The likely experience level of the audience with items like navigation, forms, logins, and other ways they may interact with a website.

Information Flow

You need to plan on all the types of information you need to include in the website. While the client should provide the content (text, images, etc.), you need to know what this will be so you can plan how to organize the website. This impacts how you:

- Create the information flow for the visitor, such as the layout and types of pages.
- Create the navigation and prioritize what the visitor sees first.
- Organize the information the visitor needs to act upon.
- Place text content, navigation, images, and overall communication items to fulfill the website's purpose.

Key information points you need to consider include:

- **Inform or educate:** Break down by presented information, where to get more information, resources/links, etc.
- **Product sales:** Break down by products, features/benefits, how to buy, testimonials/how to use, and so on.
- **Marketing and promotion**: Break down by services offered, reputation points/

testimonials, how to contact, why the service is different from similar services, etc.

- **Edutainment**: A combination of information and resources, networking/contact information, interacting with information (like media pieces), games, etc.

Website Content

You need to collect a list of the necessary content, then organize it according to the anticipated audience's needs. This requires attention to detail, since you need to also consider any items that do not support the defined purpose or accomplish target audience objectives.Then you need to remove these unuseful items.

Your client may allow you or wish you to interact with potential visitors at this point. This can be done through creating focus groups, doing "potential content" interviews, content and or navigation surveys, and using other marketing and usability tools. For instance, it can be very useful to test the content and purpose of a site on a focus group and compare the offerings to what the audience actually needs.

Once you have a clear idea of the content list, you need to organize the basic information structure by categorizing the content and organizing it according to visitor needs. For instance, for a product site, you could organize by

- Product information,
- Product support and contact info,
- Testimonials, and
- Purchasing of the product.

You could further organize the product information into product lines, featured products, discounts, and a gallery of product images.

Once you define what the categories will likely be, you should give each one a concise and descriptive title that will become a link on the website.

Website Page Types

Websites offer two types of web pages; static and dynamic. Your website may have primarily static pages, such as for a portfolio website, a brochure site for projects, or business listing sites that provide information but don't actually sell from the site. You can also design dynamic pages, especially if your client wants to sell from the website, you need an updating blog, are including contact forms, or displaying lists of information you want people to be able to sort and filter. Let's look at both types of pages.

Static pages

These look and act the same whenever a visitor enters them. They are like pages in a brochure, presenting information and prompting but not actually expecting the visitor to do anything but read on or visit a linked page that might be dynamic. While you can add little semi-dynamic touches, like embedded media, a rotating phrase of the day, or a slide-show, the actual web page doesn't change content and layout with every request unless a human (web master/programmer) manually updates the page. A simple HTML page is an example of static content.

Dynamic pages

These adapt their content and/or appearance depending on the visitor's interaction or changes in the computing environment (visitor, time, database modifications, etc.) For instance, ever notice how Amazon.com offers you reading material based on your preferences once you log in? Submitting a form, logging in, answering a poll, sorting a book list, and purchasing a product are dynamic page activities. These kind of pages need client-side scripting like JavaScript, and/or server-side programming, like PHP, Perl, ASP.net, etc. Simple forms can use HTML and a CGI form handler without higher programming languages.

Website Page Categories

Based on your information and category organization above, you can start planning what the actual pages and navigation will look like. This is where you consider how to easily break down information and activities into small chunks for easy reading and action. There are many kinds of pages that websites include, and here are examples.

First, you will always have some basic pages, regardless of the information flow you have decided to follow:

- Home page – also known as the welcome or landing page
- Disclaimer page
- Contact information
- Sitemap

Then, based on the information flow you have chosen, you can mix and match the pages as they relate to the categories you decided upon:

- About Us
- Products

- Services
- Testimonials
- Instructions/How to do
- Purchasing
- Networking Info
- Gallery
- Privacy Policy
- Contact Us
- Directions
- News and Updates
- FAQs
- And more

A good rule of thumb is to, based on your category needs, choose 5-7 pages for your primary/main navigation, including the Home page. More than that amount of primary navigation will confuse the average visitor, which is why you can consider sub pages for important but not high-priority information.

Website Page Levels

Main Level

Main level pages are the ones we just discussed – the priority categories your website requires to reach the audience. These are the pages which will be your primary navigation links. Your home page is one of them, and you would choose the others based on the website's purpose. For instance, for a products website, you might include the products, testimonials, FAQs, purchasing, and Contact Us.

Second Level

Second level pages also include important information, but are mostly sub categories of your main pages. You would not list them in your main navigation, although if you design drop-down navigation, the first sub level, or this second level, would be listed there. For example, for your products website, your Testimonials section might include an industry reviews page, a customer satisfaction page, and a page of news article links where your product has been covered.

Third Level

You may find, depending on the size and complexity of your website, that you have a third (fourth, fifth, etc.) level of pages, like tree branches with smaller twigs. These you

would definitely not place in your main or drop-down second level navigation. These can be accessed from your website's sitemap, from occasional inline links, and from the first and/or second level pages themselves. These third level pages, and ones at even lower levels, would branch out of the secondary level pages from your main categories.

Utility Pages

Utility pages are, simply put, necessary pages on your website that do not relate to the main categories. These include your Legal/Disclaimer page, your Sitemap page, and can include your Contact Us page if you want to link that somewhere other than the main navigation.

Navigation Order

The navigation order is also critical. You want to draw the visitor in with clear choices so s/he can move to the needed category/page link, then from there move to the next level and/or action to be taken.

- **Main navigation** (top level) takes visitors to the main categories (pages) of the website where they can learn more about the categories of information and move to the acting or decision-making step. On a book products website, Mystery could be a top level.
- **Sub navigation** (second-level) is for sub categories of the main section. In the case of a Books website with a main level of Mystery, sub navigation might include thrillers, cozies, forensic, etc.
- **Footer navigation** is for utility and client information links - website support links. These include pages like privacy rights, disclaimer, email link, links to a contact form, link to directions, the website sitemap, etc.

3 Web Authoring Rules

Okay, now you know something about the planning stage of a web project. It's not over yet. Before you touch fingers to keypad, you also need to add the design planning to the mix. Once you have a clear idea of the website purpose, target audience, nature of the content and a breakdown into categories, you have the core information to help you create a design concept. In this chapter we'll chat about the various considerations that go into translating the website needs into solid website design.

The first, main, and most important thing to remember is that websites are useless if they do not serve the audience they are intended to attract. If the visitors can't or won't use the website, the client's communications plan and the web author's design fails. Websites must be planned, designed, and produced with the visitor/customer as the first priority. This is known as User-Centered or Customer-Centered design.

Web Visitor Needs

Web visitors have pretty simple needs when they visit a website:

- Solve a problem.
- Get information fast to solve a problem.
- Buy something quickly and safely to resolve a need.
- Do something fun to relax their stress (like games).
- Make something happen quickly so they can move on.
- Get accurate information when they need to research/make a rapid decision.

Visitors do not need:

- Anxiety
- Stress
- Headaches
- Ear-splitting sounds
- Flashing or jumping content
- Distracting interruptions
- Irrelevant time wasting
- Dangerous or insecure transactions

- Tiny print that smacks of legalese
- Too many choices
- Unclear choices
- Wrong choices
- Misleading choices

Keep in mind the core of the Hippocratic Oath: do no harm.

Intuitive Websites

There is a terrific book, called *Don't Make Me Thin*k by Steve Krug, which offers excellent and detailed information about designing intuitive websites. The title states it all: when a visitor comes to a website, they need to easily intuit what to do to find information and take action.

First, you need to recognize that how we think people use websites is different than how they actually use them. While we may have loads of important content and images to illustrate, the visitor isn't going to read every single thing. They aren't interested in perfect text, being ushered from one important point to the next, or in reaching the conclusions we hope they will reach from reading everything thoroughly. They won't try every example, or examine every image, and make choose to ignore nice charts, graphs, and other important supplements to the content.

Axiom #2: The website experience is all about the visitor, not you or the client. It is the visitor that the website exists to communicate with – see *Axiom #1*.

Think about how you are using this book, and how you use a basic search engine results page, or Amazon.com, or your favorite news website. Visitors scan and get hooked by the items that make them think they will get the info they came for. They usually ignore much of the page, find whatever link looks like it will solve their problems, and click.

In addition, visitors can and do come into the website from every possible direction. Your website contact info may be linked from someone else's site, your sitemap may be reached from a mention in a review, or your product page may be linked to by some other website's review section. You can't control where visitors will land on your website, and you need all elements of your overall design to account for this. Then they look for whatever looks like the first reasonable option and go there. If they are not sure, they will muddle through, and if they can't get exactly where they want in 2-4 easy and clear clicks, they will leave. Bye-bye!

Axiom #3: The visitor will almost never, ever start where you want or expect them to on the website. See *Axiom #2*.

Here's your competition for a visitor's attention:

- Alternate websites they already use and are familiar with – why muddle through yours?
- Too many available choices of links just to get to you – competitor and semi-related links in the search engine results.
- Too much overstimulation from other website colors, effects, and hooks – makes your terrific design blur with the rest.
- Too many bookmarks to other pages they may also review – if they bookmark you as part of fact-finding research, they might forget why when they come back.
- Too little time to solve their information and action needs – visitors come during breaks, lunch, when they need a fast solution, while they are watching TV, and while getting phone and text interruptions.

A web author has a lot to consider just to design an effective, and attractive, and easy-to-use, and memorable, and relevant, and problem-solving, and. . . you get the picture?

Web Design Concepts

The common expectations of website presentation is constantly evolving, especially as technology, monitor size, mobile device needs, and download speeds develop. Design essentials to consider include:

- **Clear website identification**: You need to clearly identify, on every page, just what website this is. This can include the logo, title, consistent graphic identifier, etc.
- **Clear page identification**: The visitor needs to know what actual web page in the site s/he is on, so you need to clearly identify every page's name/purpose with some kind of heading.
- **Clear section identification**: A web page may cover more than one concept, or have more than one article, information box, or choice the visitor can make. If the content contains more than one topic or section, you need to clearly identify the sections with subheadings.
- **Simple and self-explanatory navigation**: Visitors should be able to determine right away where the navigation options might lead and be able to choose the one they need.

- **White space**: White space refers to the open space on a page where there is a visual break from blocks of text, images, etc. White space is important because the visitor's eyes get strained if things are too crowded together. They also lose their ability to get key decision-making information when everything looks squashed together and too busy.
- **Alignment**: Content can be left, right, or center-aligned. Choosing a generally consistent alignment is important since it allows for white space, is easier for the eye to follow, and looks tidy and put-together in a way that mixing alignments don't.
- **Think in "blocks":** Much of design is about knowing how to direct attention. Block out the navigation area, the identification area, the sub areas for text and image content, the footer area, etc. Make things easy to scan.

Design for Scanning

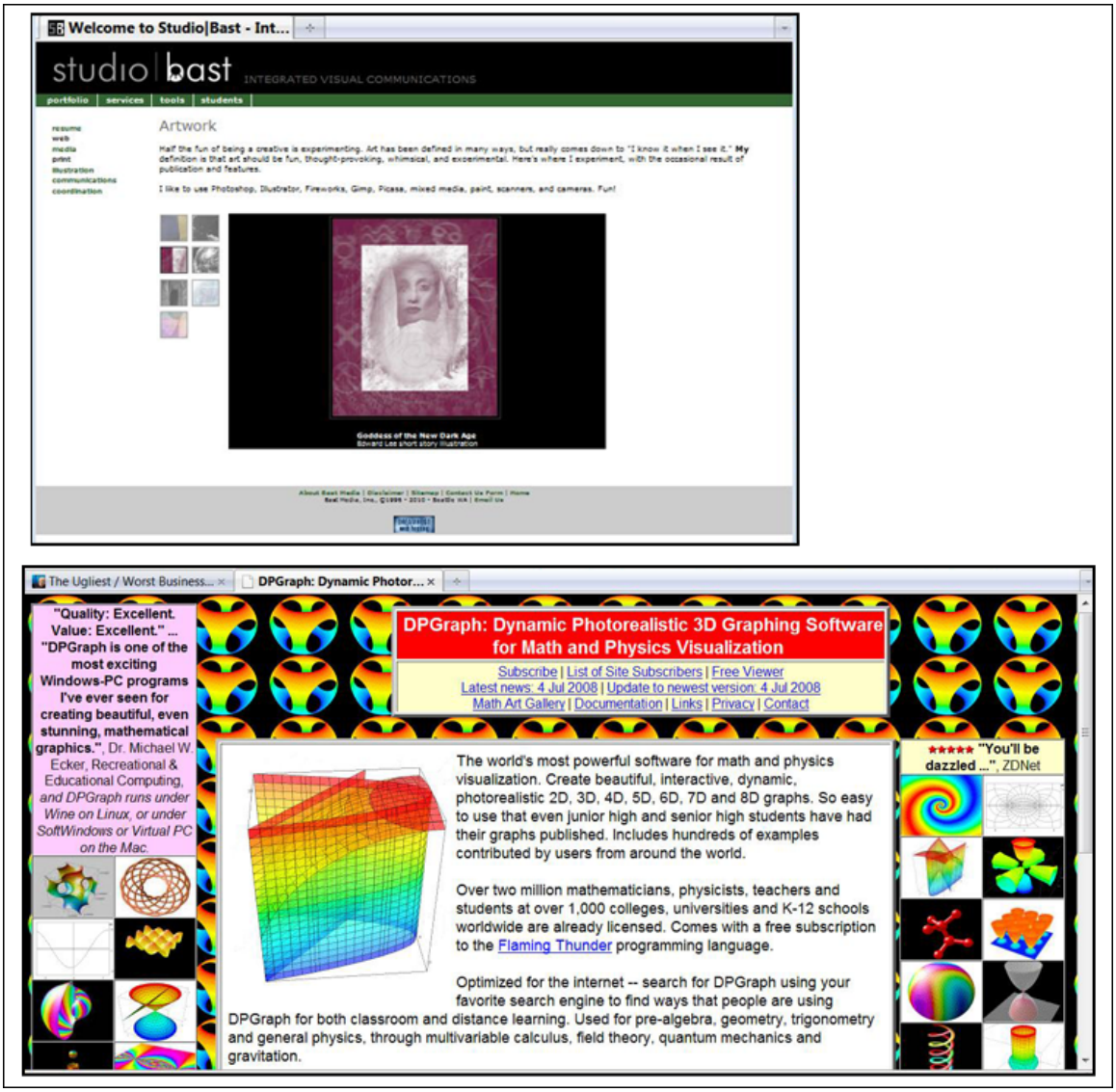
Don't make your visitors think. They are usually busy, in a hurry, and are not going to read any more than they have to in order to act or make a decision. Make any readable items very, very easy to see and read.

- **Visitors scan, they don't read**. Give them the key info in scannable bits, like subheaders and bulleted lists.
- **Break up information**. Use subheaders, and occasional sub subheaders. Use a back-to-top anchor when needed. Use a sidebar for nice but not critical information.
- **Place critical information and navigation above the scroll / on the first screen.** Think of a web page like the first side of a folded newspaper. A visitor looks at, and makes fast decisions, based on what they see at the top of the page before they have to scroll down. Do not put critical information below the first screen because the visitor will likely miss or ignore it.
- B**asic scanning methods**. Visitors scan, they don't read. One common scanning method is the Z-method, when a visitor scans top left-to-right, diagonally down to the lower left, and then lower left-to-right. Another way is the F-method, when a visitor scans top left-to-right, darts to middle left-to-right, then darts down to lower left.
- **Make sure the visitor has control over their use of the site**. Give them what they need without fuss and they will keep at it.

Enhance Readability

Even though visitors scan and choose, sometimes they do need to read to get more detailed information. Make it easy on them and effective for the message being delivered.

- **Font size**: The font size is often one of the least-considered things in a website. In order to get as much priority information above-the-fold, a designer may sacrifice readability by using a small size of text. Recognize that priority information is not useful if the visitor can't easily read it. Too large of text, particularly for paragraphs, can instead look like the website has nothing to say. Choose the font size that best suits your primary audience demographic and the message.
- **Font color**: Font color is also important for readability. When you use a light background, you need a dark font, and vice versa. If you choose a font of fairly equal intensity to the background color, they either blend together or shout out in a blur of unreadable color. Mixing several font colors can be confusing, while using only one can be boring and keep you from punctuating important points. A good rule of thumb is to choose a good base font color, like black or dark grey on white background, and use a secondary color in your palette for a couple of the heading styles, and perhaps another contrasty color in your palette for a third heading style.
- **Font emphasis**: You can and should judiciously use text bolding and italics when needed. The various HTML head tags build bold into the font display, and you can punctuate the occasional word or phrase as needed. Stay away from underlining anything (except links) because underlines make a visitor think the text is a link and because underlining text is not HTML standards compliant.
- **Text positioning**: You should format your text areas in ways that break up a solid block of same-sized text. Using headlines and subheads to break subject matter into smaller chunks is helpful. You can also use HTML indents, lists, line spacing, and more.
- **Issues**: Keep in mind that many viewers have some level of color blindness, which affects what they see. As long as your contrast is good, and you carefully use font emphasis, you can make this a minimal issue. For instance, it is tempting to use red as an alert color, but if a visitor cannot see red, s/he will not know there is an alert unless the colored word is also bold, and unless you do not bold a lot of other words.



Clean layout vs. Cluttered layout

Opposite page: 3 Layout Examples


LOS ANGELES ZOO & Botanical Gardens
Home
Visitor Guide
Conservation
Animals & Plants
Animal Health
Calendar / Events
Education
Membership
Support Us
About the Zoo
GLAZA
Volunteer
Fun Zone
Visit our Komodo Dragon Hatchlings!
Babies Galore!
The Los Angeles Zoo is proud to announce the birth of a Masai giraffe, a Calamian deer, and two Chacoan peccaries.
These recent births
L.A. COUNTY FAIR
PURE. FUN.
Sept. 4 - Oct. 3, 2010
Aquarium of the Pacific


KIDS FREE IN OCTOBER
TICKETS & HOTELS
BECOME A MEMBER
DONATE
SAN DIEGO ZOO
SAFARI PARK


WOODLAND PARK ZOO
Register | Login
Give Now!
eNews Sign Up
VISIT the Zoo
CONSERVE and Animal Info
EDUCATION and Programs
EVENTS and Parties
JOIN or Renew
SUPPORT your Zoo
my ZOO
Woodland Park Zoo wins national award for penguin exhibit!
Read the WPZ Press Release >>

Strengthen Viewability

Regardless of the visitor demographics and client's preferred color palette, there are a few style basics that enhance ease of use.

- **Contrast**: The contrast on a web page is the difference in visual properties that makes an object (or its representation in an image) distinguishable from other objects and the background. In web design, good contrast means that the weight and color of text and hyperlinks stands out easily from the background color of the page, for instance.
- **Size**: The size of objects (text blocks, images, thumbnails, navigation buttons, etc.) is important in helping a website be easy to use. Images that require scrolling to see, or big blocks of unbroken text, can make a page look crowded or overwhelming. Items that are too small can create too much white space and make it hard to discern the main point.
- **Colors**: Colors are important in two ways. The correct use of color combinations and contrast can really help a website be easy to digest, while poor color choices with bad contrast make it impossible. For instance, red text on a yellow background, or blue links on a black background, are very hard to see. Plus, colors can also trigger moods and associations in visitors. If your target audience is young, bright colors and lots of black appeal, while a senior audience benefits from conservative and generally muted but good contrast colors. If the website is meant to appeal to an international audience, knowing which colors are truly taboo to different cultures can help you make the site appealing rather than a turn-off.

Layout Size and Resolution

With so many different configurations that visitors have their computers set up with, it can be difficult to decide how a website should be sized so that it is attractive and viewable as much as possible. Here are a few tips:

- Design to have your page elements be very visible and tidy between 800×600 and 1280×1024, since this range captures the majority of current laptop screens and desktop monitor resolutions.
- The norm is to optimize for **1024x768** screen resolutions. For most sites this will cover most visitors since the average low-price monitor or laptop screen has this as a default resolution.
- Keep in mind that the real width of a 1024x768 screen is about 960 pixels. If you design with fixed widths, consider 900, 925, 950, and even 980 pixels in width. Don't design wider, because visitors don't usually scroll to the side and find it uncomfortable.

- Whenever possible, don't design for one size only, since browser window sizes vary depending on the monitor's resolution. Don't assume screen size equals windows size. Design for a reasonable minimum, but assume it will adjust.
- Do not design solely for a specific monitor size because screen sizes also vary among visitors. Window size variability is even greater, since visitors don't always maximize their browsers or their screens. Some have large screens, some have wide screens, some have standard width, etc.
- Use flexible layouts when feasible, which will adjust when the browser window is resized.
- Keep in mind that mobile devices are special. If you get lots of traffic from iPhones and other mobile/small devices, you will need to need to give them special treatment by using a different set of styles for those devices.
- The website height is usually not very important, in terms of resolution. The visitor will often scroll up and down. Just don't design so that web pages have to have more than five or six scroll-downs, because that becomes unwieldy.

The three main criteria in optimizing a page layout for screen size are:

- **Initial visibility**: Is all **key** information visible above the fold so visitors can see it without scrolling? This is a tradeoff between how many items are shown vs. how much detail is displayed for each item vs. the font sizes chosen.
- **Readability**: How easy is it to read the text in various columns, given their allocated width?
- **Aesthetics**: How good does your page look when the elements are at the proper size and location for the screen size? Do all the elements line up correctly? Do things appear to move out of alignment when the browser window is resized more than 20%?

Planning for Web Browsers

Your website should keep its layout regardless of what browsers the visitor uses. Despite this, different browsers can display web pages with slight changes, and older versions of these browsers experience more viewability issues. As a result, when doing your main coding, focus mainly on Firefox or IE, whichever your client feels is mostly in use by his/her average visitor. Test with **all** of them, but choose one to work out your main code and styles first.

As of 2010, the general consensus (Wikipedia, etc.) is that the following browsers are most in use:

- Firefox: 46%
- Internet Explorer: 31%
- Chrome:17%
- Safari: 3-4%
- Opera/Other: 2%

Search Engine Optimization (SEO)

The purpose of Search Engine Optimization (SEO) is to make a website as search engine friendly as possible. This is so that it can be found when searched for and whenever possible have high rankings in search engines like Google. Search engines collect data about a website by sending an electronic spider to visit the site and copy its content, which is then stored in the search engine's database for future reference.

Spiders (also known as 'bots') start reading in the top left hand corner of a site, and will read content line by line from left to right. If columns are used spiders will follow the left hand column to the end, then move to the remaining columns. If a spider encounters a link it can follow, it records the link and sends another bot to copy/record data found on the linked document. The spider will proceed through the original website until it records everything it finds.

SEO makes it easier for the bots to get around so the search engine's database fills accurately. You should:

- Have an SEO sitemap. Create and link to an XML text-based sitemap page in your footer that lists your main pages and/or main website topics, with links to the pages/anchors where they are. (Note, this is different than a design sitemap or a visitor sitemap.)
- Use mini-link lists on web pages that have more than three text subjects. This helps visitors jump to the section on the page, and also adds the keyword again for the spider.
- Limit access to your website with a robots.txt file in order to keep the robots from searching every external link.
- Choose a good website URL domain name. Make a site's domain name as relevant to the site's purpose/uniqueness as possible: http://www.**studiobast**.com – studiobast is the company name.
- Create an effective **title** tag in your code's head section. This title shows in the

tab of the website browser window and is picked up by search engines. Example: *Welcome to Studio|Bast – Integrated Visual Communications in Seattle.*

- Add unique/specific keywords in the **keywords** tag in your code's head section. Focus on niche keywords related to a product, service, or unique promotion of the site's contents, rather than many general ones. Also, use keywords that people actually use when performing searches on the subject your website targets. Add common misspellings that people might type in, your name, and even names of product distributors or other websites you are part of.
- Craft an elevator-speech style **description** meta tag in your code's head section. This uses two or three complete sentences with the strongest keyword phrases woven into each sentence. It is also the summary that a potential visitor sees on a search engine results page under your website's title.
- Have good, clear text content throughout the website. Search engine spiders read basic body text 100% of the time. A solid SEO guideline is if you need to more than one topic on a page, you need more pages. More pages are actually less confusing to spiders.
- Use bold and italics judiciously for keyword emphasis. Don't bold or italicize every keyword all the time, but you can, in image captions, in subheadings, and in the occasional important reference, emphasize the keyword so the spider records its importance.
- Have back links. Back links are websites that link directly to your website. Per Google, the more back links you have, the higher your pages will be ranked. This means that if you or your client have relationships with peers, suppliers, reviewers, and customers, asking them to link to your website can legitimately add those backlinks.
- Include **alt** and **title** text in tags. Always use title and alt text as needed for your images and links. This is important for accessibility and also allow you to repeat useful keywords that relate to the item being alt-tagged.
- Share resource links. If your website message really can benefit from sharing resource links with the visitor, you can add a few of these, and use keywords as you used subheadings to separate link types.

Badly used SEO can alienate search engines and get you kicked out of the indexing databases. You should **not**:

- Use too many/non-relevant back links. Back links are good if they are from websites that are related to yours, such as LinkedIn and Monster links to your resume

site. They are not useful or good if they are on irrelevant sites that list links for no reason.

- Engage in link farming. Exchanging reciprocal links with websites (particularly unrelated websites) solely to increase search engine optimization is considered spamming.
- Use irrelevant keywords. Using keywords words totally unrelated to the site's content and purpose just to get rankings is disreputable and can lead to bans. These would include keywords like sex, porn, booze, partytime, free, and other attention grabbers that are not at all part of the purpose, message, and target audience needs of your website.
- Hide links and keywords. Trying to over stimulate spiders with hidden keywords in the body of your website (like words that are the same color as the background or hidden in comment batches) is a bad idea. So bad that search engines can (and do) ban your site.
- Create duplicate pages. These are seen as spamming by search engines and also call for banning.
- Use poor or non-descriptive link anchor text. Anchor text or text link tells the search engines what the linked site is about. 'Click Here' doesn't mean anything to a spider. 'Visit Studiobast's new design templates page' does.
- Ignore best practices. An example of good practice: put keywords in alt tags, page file names, headlines, title tags and link anchor text.

Accessibility

Web accessibility refers to the practice of making websites usable by people of **all** abilities and disabilities. Accessibility focuses not only on people with disabilities, but on anyone who may experience difficulty in reading or navigating a website. For example, users with a very old computer system, a visually impaired user, or someone with a broken arm who can't use a mouse should be considered. When sites are correctly designed, developed and edited, all visitors can have equal access to information and functionality.

The needs that Web accessibility aims to address include:

- **Visual**: Visual impairments including blindness, various common types of low vision and poor eyesight, and various types of color blindness.
- **Motor/Mobility**: e.g. difficulty or inability to use the hands, including tremors, muscle slowness, loss of fine muscle control, etc., due to conditions such as Parkinson's Disease, muscular dystrophy, cerebral palsy, or stroke.
- **Auditory**: Deafness or hearing impairments, including individuals who are hard of hearing.

- **Seizures**: Photo epileptic seizures caused by visual strobe or flashing effects.
- **Cognitive/Intellectual:** Developmental disabilities, learning disabilities (dyslexia, dyscalculia, etc.), and cognitive disabilities of various origins, affecting memory, attention, developmental "maturity," problem-solving and logic skills, etc.

Therefore, keep these tips in mind when designing for accessibility:

- A site should be coded with semantically meaningful HTML, with textual equivalents provided for images and with links named meaningfully. This helps blind visitors using text-to-speech software and/or text-to-Braille hardware.
- When text and images are large and/or enlargeable, visitors with poor sight can more easily read and understand the content.
- When links are underlined (or otherwise differentiated) as well as colored, this ensures that color blind visitors will be able to notice them.
- Clickable links and large link areas help visitors who cannot control a mouse with precision.
- Visitors who cannot use a mouse or a standard keyboard benefit when pages are coded so they can navigate by means of the keyboard alone, or with a single switch access device alone.
- When videos are closed captioned or a sign language version is available, deaf and hard of hearing visitors can understand the video.
- Avoid flashing effects so that visitors prone to seizures caused by these effects are not at risk.
- When content is written in plain language and illustrated with instructional diagrams and animations, visitors with dyslexia and learning difficulties can better able to understand the content.
- When sites are correctly built and maintained, all of these visitors can be accommodated while not impacting on the usability of the site for non-disabled visitors.

Legal Content Use

Where you get the website content is as important in producing a website as making it look good. If the client does not provide you with specific content and you do not create and sell rights to it yourself, you will likely purchase images and other items from various sources. You need to protect yourself, as well as your client, by using content sources right.

Intellectual Property

Owners/creators have legal property rights over creations of the mind, both artistic and commercial. Under intellectual property law, owners are granted certain exclusive rights to intangible assets such as musical, literary, and artistic works; ideas, discoveries and inventions; and words, phrases, symbols, and designs. Know where your content and images come from, and be sure you/the client has the contracted rights to use this content.

Copyright

Copyright is a set of exclusive rights granted to the author or creator of an original work, including the right to copy, distribute and adapt the work. Copyright does not protect ideas, only their expression or fixation. In most jurisdictions copyright arises upon fixation and does not need to be registered. Copyright owners have the exclusive statutory right to exercise control over copying and other exploitation of the works for a specific period of time, after which the work is said to enter the public domain. Uses which are covered under limitations and exceptions to copyright, such as fair use, do not require permission from the copyright owner. All other uses require permission and copyright owners can license or permanently transfer or assign their exclusive rights to others.

Public Domain

The public domain is a range of abstract materials, commonly referred to as intellectual property, which are not owned or controlled by anyone. These materials are therefore considered "public property", and available for anyone to use for any purpose. A work may be in the public domain if no laws establish proprietary rights over the work, or if the work or its subject matter are specifically excluded from existing laws.

Licensing

Licensing means you purchase the right to use an image, text, pdf, etc. under terms set out in a license. The rights are set out in the license, which is a legally binding agreement between the buyer and the seller.

Royalty-Free

Royalty-free means that you pay once for an image that can be used (but not passed on to anyone else) for multiple purposes over an unlimited period of time. You still need to be careful how you use it. For example, if the image features recognizable people and you know of no model release, then do not use the image for promotional purposes.

Traditional Usage License

Rights-managed images are sold with the fee based on your specified use. This fee is calculated based on size/amount, placement, duration of use and even geographic distribution.

Reserved Rights

This includes terms restricting the photographer or writer from selling the item for a specified period in a specified location(s). This gives the buyer exclusive use.

Model Contracts

If you/the client determine that images need to be created for the Web project, they will likely be based on photography set up for the project. You might even be asked (and paid) to handle the photography yourself. Keep in mind that:

- You/the client should always get a signed and dated release from any models you take photos of.
- Taking random shots of individuals without their express signed consent is a no-no. Same goes for group shots where people can be recognized. Consider this – if one of those individuals is in witness protection, or in hiding from an abusive family member, and you publish a picture that could "out" them? **Don't do it**.
- Images of children and disabled individuals are particularly sensitive, Even if you get a signed release, aim to get neutral images in reference to the location of the individuals. Why? Unfortunately, there are predators out there who fixate on children and less-than-able folks, and if they can suss out the location of the image, they have a bead on where to start looking. Be safe.

The long and short of content use is that you always need to know where it comes from, and that you/the client have legal rights to use it.

Open Source Content

Sites such as Wikipedia and Wiktionary have embraced the open-content GFDL and Creative Commons content licenses. These licenses were designed to adhere to principles similar to various open-source software development licenses. Many of these licenses ensure that content remains free for re-use, that source documents are made readily available to interested parties, and that changes to content are accepted easily back into the system. In a sense, open source is treated like public domain work.

Keep in mind that open source licensing does come in several formats, and be sure which you understand the terms of any you might use. Some open source is free to use but not sell. Some is free and freely distributable. Some may be modified, edited, and used for educational, or commercial, or personal use.

QuickLook

Open Source and Copyright

Web Authoring Boot Camp uses some information from completely open source content, like Wikipedia and Wikiversity, to enhance the author's (my) experience and content. Wikipedia and similar sources are considered completely open source, especially since most of the information is provided by visitors.

Meanwhile, this published book is also licensed under Creative Commons for editing, modification, and use for education purposes as long as the author is acknowledged. I, as the author, have the copyright on it simply because I completed and fixed this content and can register the copyright formally if I choose.

The copyright on any website belongs to the client, not the designer. The web author should be able to take screenshots for his/her portfolio, retain core design code structure for generic reuse, and take design/production credit, but the client owns the content and everything proprietary. If the web author leases or sells an image that is used on such a site, the web author owns the image, which is licensed by the client, who overall has the website copyright. Fun, yes?

4
Web Page Basics

Web pages in a cohesive, professional website should have an organized, standardized, and easy to view layout. There are several ways to create a layout, with different navigation, content, and sidebar orientations, but usable and accessible websites have several things in common, and we'll take a look at those.

Layout Structure

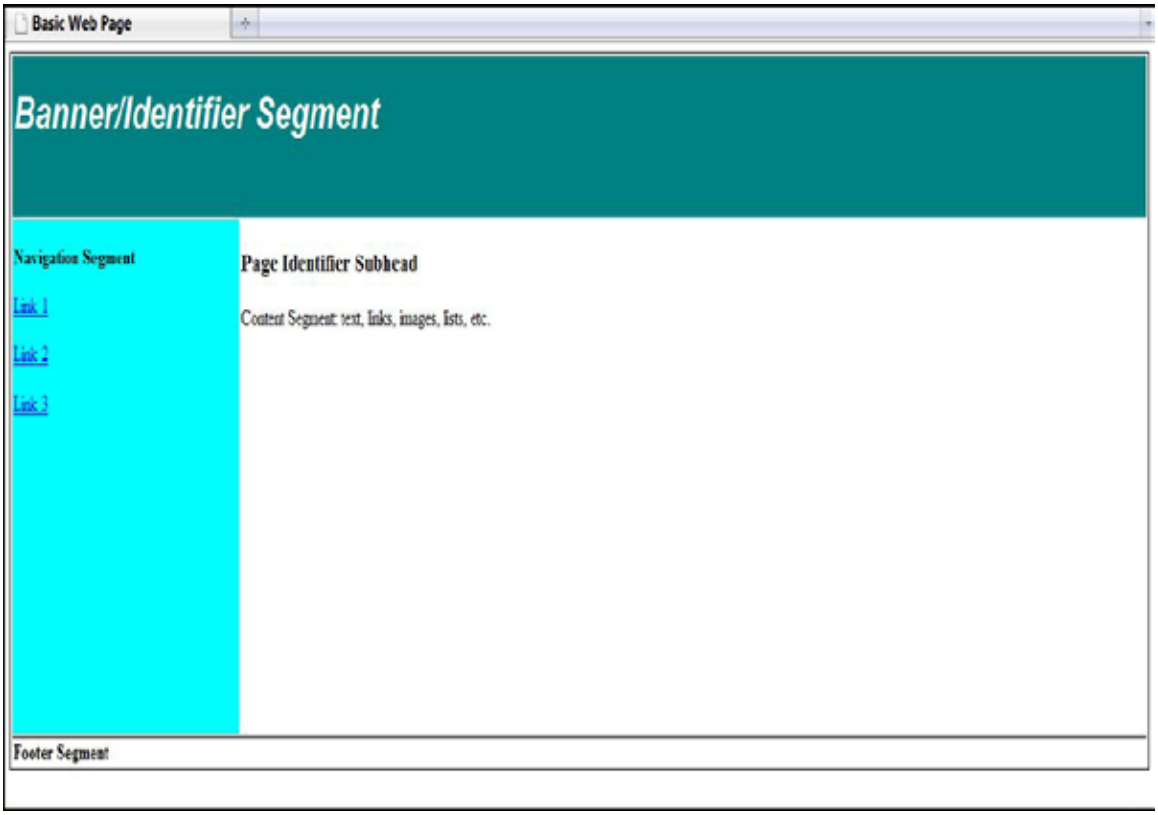
Study successful websites to get clear on basic website layout patterns. You can see many examples, and you can intuitively tell which ones are easy for you to comprehend and use. Here are a few tips.

- **Banner identifier segment**: The banner segment is usually used as a **website wide** identifier segment – where the site's logo, title, or other site identification content goes.
- **Primary navigation segment**: Should be set in some way above-the-fold so that the visitor sees it immediately and can click on the category links.
- **Content segment**: The text, and most images related to the text, will fall into the content segment of the web page. This is where the bulk of the website's communication efforts take place, and is also where you will likely place the specific web *page* identification.
- **Footer segment**: The footer area of a page includes useful but not critical information, like copyright and email links.

Banner/Identifier Page Segment

The banner segment on a web page is usually used as a website-wide identification area. It is horizontal, occurs at the top of the web page and generally spans the page's full width. This is where the design starts, in terms of where the visitor first looks and the impact the client needs to create right away. It includes the:

- Client's logo or logotype
- Tagline
- Website title
- Other site identification content, like a unifying image or design.



Basic web page layout

Some websites leave much of the banner area empty, with only a logo or website title.

Others use the banner area to add extra functionalities, like a site Search input box, a hotlink promotion to something important, a randomly-changing phrase to seem dynamic, etc. Some websites even do away with the top banner altogether, using the upper right or upper left side of the web page to place indentifying information above vertical navigation.

Navigation Page Segment

The navigation is critical, and should be prominent, consistent through the entire website, and have excellent contrast and clickability. Your visitor needs all this to reach information elsewhere in the website. If it is at all confusing or misleading, the visitor is likely to leave.

Main Navigation

The link to top-level pages in the site is the main navigation. This could include the Home, About Us, Products, Contact Us, Gallery, and other main areas that directly support the purpose and messaging of the website. Ideally there should be somewhere around 5-8 links, based on you how categorized the content structure of the website. There are several basic formats:

- **Standard left navigation**: The main (top-level) navigation links are vertical, down the left side of the page.
- **Right navigation**: The main navigation links are vertical, and down the right side of the page. Blog sites often do this.
- **Top navigation**: The main navigation links are horizontal and placed across the top of the page below the banner segment. Sometimes it is placed above the banner segment instead.
- **Mixed navigation**: Very large websites, with divisions, may use a mixed navigation. The main navigation for the *whole organization* may be horizontal at the top of the site, and the main navigation for a company's *division* might be vertical on the left.
- **Image maps**: Image maps aren't used much for main navigation any more, since a key to design consistency, and ease for the visitor, is to use the same navigation style on every page of the website. However, in some cases a designer has used a large images on the home page, and hyperlinked parts of it to key navigation sections.
- You will also find examples of versatile navigation thinking as you explore other websites for ideas. Just keep usability and easy of choice in mind when you experiment.

Secondary Navigation

Secondary-level navigation is based on the sub pages of the main top-level navigation. For instance, if you selected one of the main navigation links, such as *About Us*, it should take you to the About Us page. Now, the About Us section of the website might encompass 3 smaller pages, including *Contact*, *Staff Bios*, and *Reviews*. The navigation to these pages from the *About Us* page would the **secondary** navigation from the *Home* page.

Secondary navigation can also appear in many formats - some subtle, some creative, and some not even appearing on the home page. Whichever you choose, it should use colors, fonts, and styling consistent with the rest of the site and with some style relationship to the main navigation. It also should never intrude upon or confuse the visitor from the pri-

mary first-level navigation.

- You can build in secondary navigation as horizontal dropdowns or vertical slide-outs to the main navigation, or
- Add the secondary "section" navigation as part of the web page identifier/header area.
- Only have the secondary navigation appear on the specific top-level page it is linking from.

Inline links

You can hyperlink text inside paragraphs, address blocks, your footer, and around images. A good rule of thumb is to use inline links carefully, since they can distract the visitor from the scanning and choosing s/he needs to make. At the same time, linking inline can be useful for showing a tooltip or moving to another critical page on your website. It can also be great when you are sharing a list of useful links that the viewer can enjoy. Use your judgment and err on the conservative side.

Link Rot

Link rot (or linkrot) is an informal term for the process by which, either on individual websites or the Internet in general, increasing numbers of links point to web pages, servers or other resources that have become permanently unavailable. The phrase also describes the effects of failing to update out-of-date web pages that clutter search engine results. A link that doesn't work anymore is called a broken link, dead link or dangling link.

You need to keep linkrot in mind when you are designing how our navigation and linking works on your website. The more links you add, the more confusing keeping them straight, and maintaining a low or no-linkrot site you will have. Fortunately, Google and other search engines have some tools to help you keep an eye on this, but being careful and savvy about your linking in the first place is easiest.

Content Page Segment

The text, and most images related to the text, will fall into the content segment of the web page. This is usually most of the center of the page, offset to the right or left by side navigation. It may be broken into two or three columns for sidebars and easier reading. Here you will use a combination of:

- Web **page** identifier (a heading)
- Subheadings

- Paragraphs, lists, and callouts of text
- Images and captions
- Media embeds and/or links
- Tables of information
- Boxes with content in them

Content layout can include one wide column, two medium columns, three narrow columns, a wide column and a narrower sidebar, and endless other possibilities. Explore the Web for visual ideas.

Image Use

Images are half the fun of a lively and attractive website. You can do all sorts of things with them, but keep a few hints in mind:

- Design clean and tidy alignment for placing images, rather than placing images helter skelter to fill up space.
- Choose images that directly support the purpose of the website and enhance the color palette and the needs of the audience demographic.
- Keep image contrast and coloring in mind as well, so that the images don't distract from the important information and links.
- Add captions when needed to identify the image and/or give credits.
- Make the image sizes consistent, rather than mixing and matching various sizes and shapes, since this looks more professional and easy to scan.
- If you do want a mixed and matched look, consider creating a collaged image and bringing it in at once, rather than trying to place several images in a collage sort of way.

Footer Page Segment

The footer portion of the web page should, like the banner and navigation segments, remain consistent throughout the full website. It provides a recognizable section where visitors know to look for certain information, like the copyright. It should stand out but not overwhelm any of the rest of the web page layout, since it is a supporting segment. You can include:

- Copyright information
- Client address
- Email contact link

- Link to a contact form
- Link to a legal disclaimer page
- Link to a privacy rights page
- Link to press releases or other media information
- Link to careers and about the company information
- Link to help or FAQs pages
- Link to the website's sitemap

Copyright

Information, logos, and graphics on a Website are protectable under the United States Copyright Act. Copyright status can prevent others from copying and using tangible works of authorship, such as information contained on a Web page. Make sure to add the copyright date on your footer. A typical copyright notice is as follows:

> **Copyright © 2010 by YourName. All Rights Reserved.**

5 Planning Your Work

A web author's project can be a new website, revision of an existing website, maintenance and updates only of an existing website, and/or expansion of an existing website. This book will look at the web project process from the new website angle, although much of the information provided will be relevant to the other web projects.

Planning for Usability

Planning for usability **before** you start designing is crucial to creating a successful, easy-to use website. This will include synthesizing the answers to several core questions, your client's needs, some research into the kind of audience you expect to use the site, and cleanly organizing the results. Then you translate those results into an organizational structure for the website (design sitemap), then build the look and feel on top of that.

Core Questions

First, ask the client these questions, which provide the core information about the project. Then summarize the site/client assignment for yourself.

- What is the primary purpose of the website - what does the client want it to do for his/her company?
- Who is going to be the primary visitor (customer) of the website - est. demographics, anticipated customer needs, etc.?
- What are the 2-3 main tasks the visitor is expected to accomplish using this website?
- What minimum information must the website provide that will allow the visitor to easily and successfully handle these tasks?
- What limitations are being placed on this web project - by the client, your timeline or resources, customer demographics, etc.?

Client Survey

Set aside enough time to have an interview and/or go over a questionnaire with your client, so you can get a lot of detail about what the client's current status and projected project needs will be. Things you should ask several clarifying questions about include:

- Project logistics
- Project requirements and business needs
- Website audience details
- Concept needs
- Resources needed

Meeting with your client is the best way to discuss these things. Here is a simplified **Client Survey** I use that might give you some ideas for your own preparation.

Project Preparation

- Who is the client's primary contact and decision maker? Get name, title, email addresses, & phone numbers.
- What is intended launch date for website, and why?
- What is the budget range already established for this project. Can project be divided into phases to accommodate budget / timing constraints?

Project Purpose

- Why does client want/need a web project at this time?
- Is this a new website concept or a redesign of an existing website?
- What is the client's primary business objectives/value proposition with the new materials? Secondary? (Examples include increased sales, marketing/branding awareness, and fewer customer service calls.)
- What is the main business problem the client needs to solve with new materials – what key message must get through?
- What specific areas of current client materials are successful? Why?
- What shortcomings exist with current client materials, and what 3 things should change?

Website Audience

- Who does the client believe is the audience for the website? (general profiling, more than one audience type?)
- What is the primary "action" the website visitor should take? (make a purchase, become a member, search for information)?
- Has audience been considered before selecting the technologies/extras for this

website? (Often newest technologies/modules overwhelm audiences when not needed)
- Has client gathered audience feedback for current materials? If so, how long ago? What is the feedback?
- What are the key reasons why the target audience chooses/should choose the clients products/services over competitors? (cost, service, value)
- Use a few adjectives to describe how the audience should perceive the website. (Examples: prestigious, friendly, corporate, fun, forward thinking, innovative, cutting edge.)

Brainstorm Concepts & Needs

- Client should share examples of materials (ads, brochures, publications, etc.) or websites s/he finds compelling.
- What visual elements or content should be utilized from client's current materials or website site (logo, color scheme, navigation, naming conventions, etc).
- Does client have a brand/identity in place? Specific color palettes & fonts to use? Is client open to completely new concepts and colors?
- Will website require specific technologies (Flash, DHTML, JavaScript, Real Audio, e-commerce, database-driven, logins, etc.)? If so, how should they enhance and clarify your audience's experience? Please describe in detail.

Resources

- Does website have / need a domain name? A server to host? Ability to update in-house?
- If this is a website redesign, are there specific tools/languages the client requires be used? (Like Dreamweaver, FrontPage, VS2005, php, etc.)
- Who will provide the website content (text, logo, images)? Is all the provided content the legal property of the client? (If not, do NOT use)
- Does client need graphics/graphs/charts created by designer, or purchased from stock photo houses? Does client have appropriate approvals / content/image use contracts in place?
- How will client maintain the website?

Based on the client interview/questionnaire, you would summarize the client's needs, expectations, and resources.

Audience Research

Next, do some audience research. In short-term website projects you may need to intuit as much audience information from the client interview as you can, and supplement with general demographic research. When possible, find ways to actually interact with people from the potential audience. Things to consider should include:

- Who is the **real** target audience?
- What will this audience want/need to make this website work for them?
- What content should be included to best suit the audience?
- What technology will the audience likely have to access/use this website?
- What comprehension/language/visibility issues should you consider for the audience?
- How/where will the audience consume the website - work, home, school, library, wifi, dial-up, palm?
- What would be the simplest and most effective navigation for the audience to find information?
- Based on the client's expectations and audience research, what is the best use of colors, fonts, images, and layout?
- What language and writing style will provide the easiest and most successful use of the website?

Planning Tools

There are a few tools/steps you'll want to include in your planning. These help you sort, clarify, and present the information you collect so you can work from it. You may also need to share some or all of these pieces with the client as you prepare a design, since your client may see needed changes or misconceptions of what you discussed. Correcting problems early will make your web project more cost-effective and polished the first time around.

AIA Table

An AIA (audience, issues, and approach) table is a simple way to collect key information about the potential audience(s) you will be serving with the website. Create a grid table, and define the following.

- **Audience**: What audience will be using the site? Consider at least 2-3 audience

types, one of which might well be the client.

- **Issues**: What issues may there be in reaching these audiences? Consider things like accessibility, age, language, time, steps to expected actions, and so on.
- **Approach**: How will you approach these issues so you can plan to meet the audience needs?

AIA Table

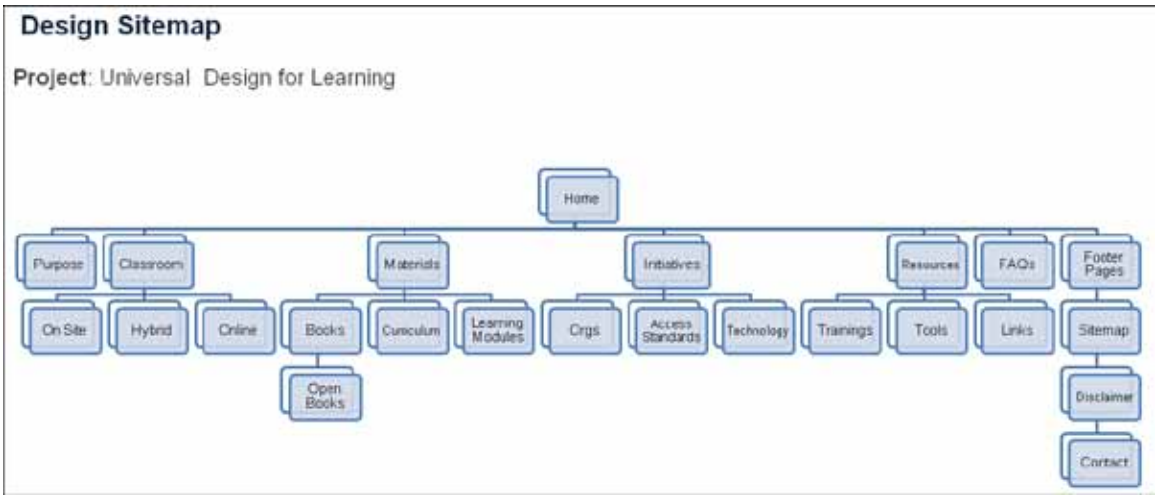
Design AIA

Project: Universal Design for Learning

Audience	Issues	Approach
Educators Scenario: Instructors who plan to integrate UDL techniques and resources in the classroom.	•Time available for research •Download speeds can be limited •Need info quickly/easily •Need regulatory info for use •Limits to curriculum additions	•Navigation relevant to educators •Small images, PDFs and modules •Multi-purpose linking styles •Easy link to required info, like FAQs •Classroom tools and resource links •Note: Needs to be bookmarkable
Administrators Scenario: Institution administrators who need to support UDL in school offerings and resources	•Time available for research •Download speeds can be limited •Need to know where to get info •Need national standards & usage •Need to know how tools relevant to national education program requirements	•Navigation relevant to administrators •Small images, PDFs and modules Strong SEO/indexing support for finding on search engines •Easy link to regulatory info, like FAQs •Access to US educational initiatives and standards •Note: Needs to be professional-looking, searchable and printable
Students Scenario: Students who will use UDL techniques, such as ESL, non-conventional students, and those with learning disabilities	•Download speeds can be limited •Need to know where to get info •May have ESL or other special learning needs •Need to know what supports student-specific needs	•Small images, PDFs and modules •Strong SEO/indexing support for finding on search engines •Cross linking general resources and instructor materials •Note: Needs good viewability and excellent accessibility

Visual/Design Sitemap

A visual sitemap is a kind of organizational chart that illustrates which website pages will be the primary top-level, second-level, third-level, and utility pages. It can be hand-drawn for your own use, but should be generated with a program for professional/presentation use. You will likely want to show it to your client so s/he can verify you have planned for the needed pages.



Wireframes

A website wireframe is a basic visual guide used in interface design to suggest the structure of a website and relationships between its pages. A webpage wireframe is a similar illustration of the layout of fundamental elements in the interface. Typically, wireframes are completed before any artwork is developed. They should be generated during or after the sitemap step so that you have an accurate idea of the actual pages that will be designed and listed in the navigation.

Wireframes allow for the development of variations of a layout to maintain design consistency throughout the site. This is an important part of the initial development stage because it creates user expectations and helps develop familiarity with the site. Wireframes also act as a way to communicate with clients and stakeholders, such as content creators, engineers, and developers.

The simpler the wireframe, the better. This doesn't mean a quickie sketch, since the wireframe needs to accurately show layout. It means:

- Lay out the general design concept without any distracting colors and styles.
- Stick to all black, white, and grays.
- Use standard fonts only, but no font coloring, styles, or special types.
- Don't insert blocks of text where text will go, but instead black lines to show the text rows.
- To illustrate where images will go, use light gray boxes or simple black outlined

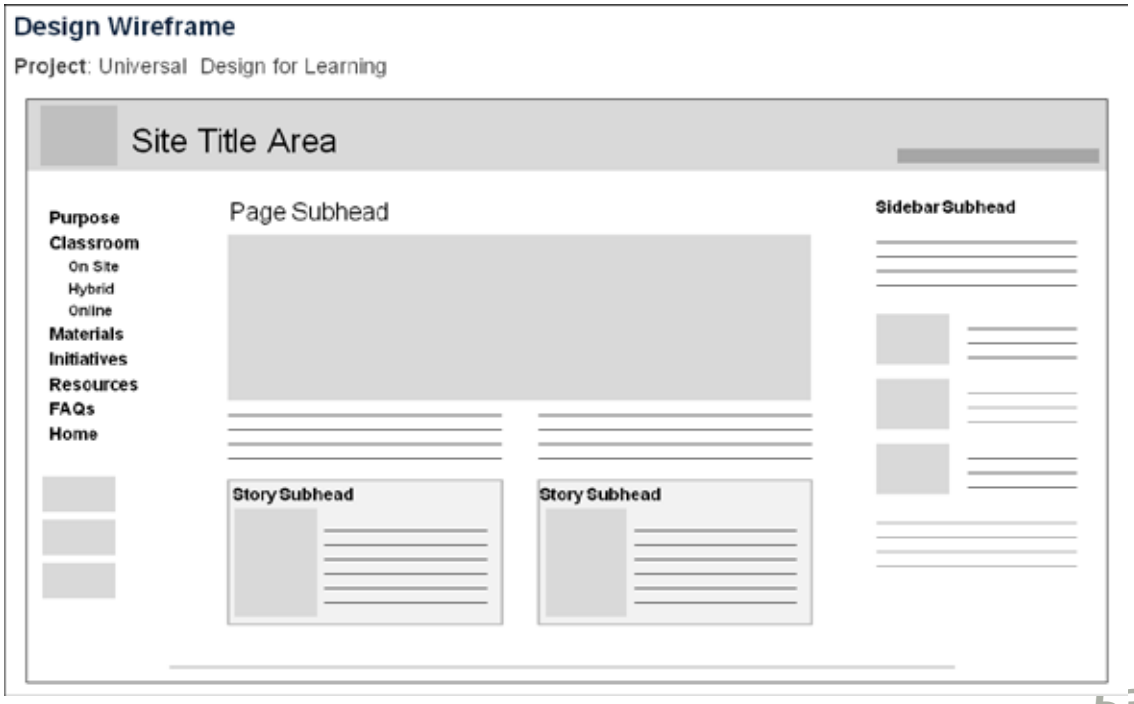
boxes.

- Use identifying phrases for main points, like Logo Here, Header, Subheader, Caption.
- Spell out the Navigation links so the client can see what the top-level links will be.
- Add estimated measured widths and heights for key elements.

You should show at least two wireframed pages, especially if you will have differences to the template layout.

- Lay out website placement of banner/ID, navigation, main content, footer
- Show the home/landing page since this is a main concern for the client.
- Show a sample second page.
- If you have differences in a second-level page layout, wireframe one of these too.
- If you have something unique being included in the website, like a gallery, a table listing products, etc., make sure to wireframe those pages too.
- Add small, clear notations of the expected width and height of each web page segment (banner, navigation, content, footer, sidebar, etc.)
- Be accurate and plan for your abilities to actually code the represented website.

Wireframe



Design Comps

A design comp is a comprehensive layout of the way the website should actually look after it has been coded. The illustration element may incorporate stock photography, clip art, or other found material that gives an idea of what should be visually communicated, before entering any negotiations concerning the rights to use a specific image for the purpose. The comps should be done after you and the client have worked through any issues brought up during wireframing.

Follow your wireframe layout but flesh it out with real text, images, color choices, font styles, and all the visual elements the client needs to okay. Create the comp in an image editing program, or if you are more comfortable, a design publishing program. Just save the final version in a non-editable format like PDF.

- Use the design color palette, selected fonts, lists, tables, etc in your concept.
- Illustrate the real banner segment with the logo and title, the top-level navigation, the main content segment, real appropriately-sized images, and the footer segment.
- Again, include the home page. Create comps for the pages you used in your wireframes.
- If you do not have real text yet, and likely you won't, use lorem ipsum filler text, with the accurate header and subheader names the site will actually use.
- If you do not have the final images, you can use free comps or find/create filler images in the theme and style the site needs.

Remember, these design comps are for the client's approval before you ever start coding the actual site. DO NOT code before doing layout comps, or try to replace the comps stage with code. Comps really are a time and money saver, even if they feel like an extra step. They will help you have the final details nailed down before you start applying code and creating code styles.

Time Management

As you can see, you have a number of things to research, plan, and do before you start the business of the website build. Time management is a critical skill you need to develop right away so that you can manage the web project from one end to the other. You will face short projects, mid-size multi team member projects, and ongoing work. Setting your priorities, learning your limits, and keeping excellent track of your workflow and resources is a must. Here are a few tips:

- Know the drop-dead launch date, and plan to be fully ready to launch at least days before.

Comprehensive (Comp)

Time management tips, continued

- Know and create a calendar for each stage of the web project.
- Set deadlines at every step of the way, and make sure you hold yourself and your client and contacts to them.
- Get your project plan, research, and resources squared away early.
- Consider using a free/inexpensive online project management program to collaborate with your client and vendors and team members you work with. Use something simple and no-brainer, since the tool should help your speed, not slow you down.
- Use your AIA table to solidify the website audience solutions.
- Use your sitemap to determine the number and levels of pages you will author.
- Use your wireframes to create an accurate layout.
- Create fleshed-out comps to guide your coding.
- Determine the project needs versus your capabilities.
- Get help when you can.

Web Authoring **Boot Camp**

BUILDING BLOCKS

section 2

Tools, process, and code

In this section, we're going to look at an efficient web building workflow, plus all the basics of HTML coding you really need.

6 Tools For Web Authoring

Web authoring doesn't happen in a vacuum. You need to have specific tools in place so you can code efficiently. There are a number of products on the market – free, shareware, and commercial, and they have their place. This book will focus on free tools that let you hand-code from scratch. Eventually you may choose one or more commercial tools, and certainly work with them in employer settings. However, for the purpose of web authoring boot camp tasks we'll be using basic free tools.

Text Editors

A text editor is a program used for editing plain text files. A plain text file is represented and edited by showing all the characters as they are present in the file. The only characters usable for 'mark-up' are the control characters of the used character set; in practice this is newline, tab and form feed. The most commonly used character set is ASCII, especially recently, as plain text files are more often being used for programming and configuration, and less frequently for documentation (e.g. detailed instructions, visitor guides) than in the past.

Some text editors are small and simple, while others offer a broad and complex range of functionality. Some text editors include specialized computer languages to customize the editor (programmable editors), and some

include source code syntax highlighting and automatic completion to make programs easier to read and write. Programming editors often permit one to select the name of a subprogram or variable, and then jump to its definition and back.

Typical features

- **Cut, copy, and paste**: Most text editors provide methods to duplicate and move text within the file, or between files.
- **Text formatting**: Features are available like line wrap, auto-indentation, bullet list formatting, comment formatting, and so on.
- **Undo and redo**: As with word processors, text editors will provide a way to undo and redo the last edit.
- **Search**: You can often search and replace strings of info with other info.

- **Data transformation**: Reading or merging the contents of another text file into the file currently being edited.
- **Filtering**: Some advanced text editors allow the editor to send all or sections of the file being edited to another utility and read the result back into the file in place of the lines being "filtered".
- **Syntax highlighting or coloring**: this contextually highlights software code and other text that appears in an organized or predictable format.
- **View in browser**: Many text editors have a command that allows you to open the current file into your computer's default browser. You may even be able to change the default browser for the editor program.
- **Code formatting**: you should be able to wrap long lines of code, and to turn on line numbers to help you with troubleshooting validation errors.
- A good text editor will have other simple features, like macros, snippet saving, encoding and file formats, printing, and FTP interface.

QuickLook

Mac, PC, and Linux Text Editors

Here are a few free text editors. Just search online and download, install, and explore.

- Aptana Studio
- BBEdit Lite
- ConTEXT
- Crimson Editor / Emerald Editor
- Komodo Edit
- Notepad+
- NoteTab Light
- TED Notepad
- TextWrangler

Browsers

You will need at least one web browser on your computer to review your code output, test layout differences from other browsers, etc.

Although browsers are primarily intended to access the World Wide Web, they can also be used to access information provided by Web servers in private networks or files in file systems. This means you can look at your coded pages locally, even before you upload them.

You should download and use at least the current Internet Explorer, Firefox, Chrome, and the previous version of Internet explorer. If you have a Mac, you should have Safari available.

Mobile Device Testing

Currently, Opera is the only regular browser that also can show how your website will be viewed on mobile devices. More microbrowsers have been developed to work with mobile phones, but most are proprietary to the mobile device and you can't test web page views with them on your computer.

You can also find a website, like http://mobiready.com/, which can run a virtual test from a live URL. Also, the W3C has an available W3C mobileOK Checker which shares errors and concerns but does not show the actual web page image on a virtual mobile device image.

File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is a standard network protocol used to copy a file from one host to another over a TCP/IP-based network, such as the Internet. FTP is built on a client-server architecture and utilizes separate control and data connections between the client and server. FTP is used with visitor-based password authentication or with anonymous visitor access. Essentially, FTP is just like transferring your files from your USB Flash drive to your computer – only you need to log into someone else's computer so you can transfer your files there.

How it Works

FTP is easy. Once you log into the Web host computer and reach the directory where you website files should go, you just drag and drop your files from the local (your) computer to the remote (web host) computer. You should receive a prompt asking if you want to overwrite existing files, if you have already placed files there. If you want to update with the new files, choose yes.

While FTP interfaces have their differences, the commonalities include:

- **Left** Pane = Local files (your computer).
- **Right** Pane = Remote files (web host computer).
- Left and Right side folder views must be exactly the same on each side before you transfer files.
- File names should ALWAYS be all **small** text - no capital letters.

FTP Programs

Here are a few FTP client programs. Just search online and download, install, and explore.

- Cyberduck
- Filezilla
- winSCP
- Fetch (for Mac, not free)
- FreeFTP
- FireFTP

Website Host

A web hosting service is a type of Internet hosting service that allows individuals and organizations to make their own website accessible on the Web. Web hosts are companies that provide space on a server they own or lease for use by their clients as well as providing Internet connectivity, typically in a data center. Web hosts can also provide data center space and connectivity to the Internet for servers they do not own to be located in their data center, called colocation.

The host may also provide an interface or control panel for managing the Web server and installing scripts as well as other services like e-mail. Some hosts specialize in certain software or services (e.g. e-commerce). They are commonly used by larger companies to outsource network infrastructure to a hosting company.

Code Validation

The Markup Validation Service by the World Wide Web Consortium (W3C) allows Internet visitors to check HTML documents for conformance to HTML or XHTML standards. It also provides a quick method for web page authors to check their posted pages for markup errors. You need to get into the habit of doing this for all your web work so that your websites have the best chance of working properly on different browsers.

As for every language, HTML and XHTML have their own grammar, vocabulary and syntax, and every document written with these computer languages are supposed to follow these rules. However, just as texts in a natural language can include spelling or grammar errors, documents using markup languages may (for various reasons) not be following these rules. The process of verifying whether a document actually follows the rules for the language(s) it uses is called validation, and the tool used for that is a validator. A document that passes this process with success is called valid.

With these concepts in mind, markup validation can be defined as the process of checking a web document against the grammar (generally a DTD) it claims to be using.

HTML validators operate by comparing the mark-up on a web page to the W3C standards. The standards vary depending upon the declared doctype version and so the validator will start by reading the doctype declaration to see which set of standards to apply.

Once the validator has read the page and determined the applicable standards it looks for such things as missing opening or closing tags, missing quotation marks and other hand-coding errors. The validator then provides a report indicating that the coding is correct or not. Errors are noted in a list.

Professional Practices

The busier you get, the less time you will have to build everything from scratch. Therefore, you should create a template directory in your web authoring file area where you can keep reusable information. Just have stuff ready to grab and go!

Placeholders

You will often, in building a web page template, need to add placeholder text and images while you wait for the real thing from clients or design team members. Here are some tips:

- **Use Lorem Ipsum**: Lorem ipsum text is placeholder text (filler text) commonly used to demonstrate the graphic elements of a document or visual presentation, such as font, typography, and layout. The lorem ipsum text is typically a section of a Latin text by Cicero with words altered, added and removed that make it nonsensical in meaning and not proper Latin. You can find free sources and generators online: http://www.lipsum.com/.
- **Images**: Create placeholder image boxes of several sizes. You should have standard thumbnail, small, medium, large, and screen-wide placeholder image boxes at your fingertips so you can place them in place of pending images. Consider the dummy image generator: http://dummyimage.com/.

Code Snippets

As you discover code items that you consistently reuse, make sure to capture this code and save it in an HTML file you can grab it from.

- Doctype
- Meta tags
- Cascading style rules
- Table layouts
- Footer code starters
- Disclaimer text
- Sitemap page starters
- Full-page code templates
- Cascading style sheets
- Script libraries

Cheat Sheets

There is so much to cover for HTML and CSS styles that you really could use a reference book to get all the detail we can't cover here. However, you can save yourself a few bucks by making sure you get a few free cheat sheets that are online. You should use a search engine to look for the following sheets. You can also try **http://www.cheat-sheets.org/** although there are no promises as to how long that link will be active.

- HTML 4.0 Cheat Sheet
- XHTML 1.0 Cheat Sheet
- CSS2 Cheat Sheet
- CSS Shorthand Cheat Sheet
- Web Safe HEX Colors
- HTML Characters

7

Web Coding Workflow

Now that you know about (and have already downloaded and installed, right?) your basic web coding tools, you need to learn the steps to making good code happen.

Coding Workflow

Coding workflow is different than your overall web project workflow, although you will exercise the same self-disciplined, organized, and time management muscles. The following steps are very useful for keeping track of your code, files, drafts, resources, etc.

- Develop your own clean, efficient workflow for your actual code files. For instance, in your code, you should chunk areas together that make something happen, and give visual breaks in the code between these areas.
- Use clear, frequent, and obvious commenting in your code. Assume someone who knows nothing about your project may review or maintain your code, and excellent commenting will help them know (and you remember) why you coded certain things a certain way.
- Backing up. Your client never ever cares that your dog ate the pen drive, the laptop fell off the top of your car, or that your server crashed. Need I say more? Just back up your work in more than one place.

Assets Workflow

Part of web coding includes handling your website assets, such as images, style sheets, javascript files, snips of borrowed code, your template, etc.

- Streamlined graphics are key. All graphics used should be the final, web-ready jpgs, pngs, and gifs for the web. You should also try to size the saved image as close to the style you expect to use it in the web page, rather than grabbing a 5x5 inch image and using code to shrink it visually to 2x2 in.
- In creating graphics, save a layered master file in your image editor that you can adjust later, then save a copy of the layer you need for the web as a web-ready image file.
- When you graduate to using cascading style sheets, make a backup copy before you make significant changes or start experimenting.
- When you create a template page for your website, save it in a subfolder called

template so that you do not accidentally overwrite it in the your main web pages directory. When you need to use the template, open it, and save a copy under whatever name you need to give it.

Extra hint: I like to create reusable items I can plug into my projects - snippets, comments, code, outlines, etc. If you make it work, find a way to save and reuse! Just add them to your templates folder during the project, then make a copy of that folder and save it in your web authoring resources directory for later use.

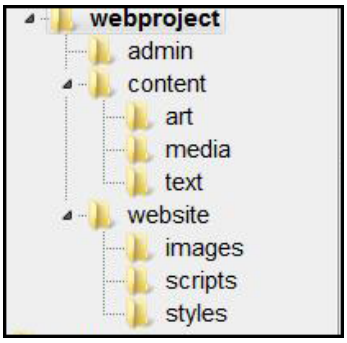
File Directory Structure

A directory structure is the way an operating system's file system and its files are stored. Files are typically displayed in a hierarchical tree structure in your computer's file directory program and in the FTP left and right panes. For web projects,

- Use a single file directory for your full web project.
- Create subfolders as needed for the elements of your overall project, such as for administrative files, raw images, instructions, website, etc.
- For your coding of the website, create its own subdirectory in the project directory.
- Create separate folders inside your website directory for different assets of your site. It's tempting to put all your html, css, image, and notes files in the same directory, but DON'T! Always have a separate folder, in your website coding directory, for images, styles (css), scripts, includes, and content you are keeping notes on.

Create a simple and sustainable set of directories, folders and files so the website can interlink both files and media without "losing" track of where the linked files actually are.

File Directory



File Naming Protocol

Files need a definite naming protocol in web authoring. This is partly because the browsers only recognize certain file extensions, partly because code doesn't work right with certain file names, and because consistent file naming helps you and your client keep the many web project files organized.

HTML files

The normal filename extension for files containing HTML is .html. A common abbreviation of this is .htm, which originated because some early operating systems and file systems, such as DOS and FAT, limited file extensions to three letters. Stick with *.html* because it is most familiar to visitors, but whichever extension you choose, be consistent.

Other coding languages have different file extensions, but for HTML/XHTML coding we are concerned only with .html.

CSS Files

CSS (Cascading Style Sheet) files use the.css extension.

Text Files

You may choose to keep text-only notes to yourself, or use a text file to store snippets of code. The extension for this file type is .txt.

Image Files

Image files use the extensions of the format they are saved for the Web in: .jpg, .gif, .png.

File Names

- Due to the evolution of coding and other programming and scripting languages, the specification for naming your files is to use all small letters – **no** capital letters.
- Also, the extension part of the file name needs to be all small letters.
- Whenever possible (hopefully always), use single word names or two words together. Try to avoid hyphens and underscores in the name, since those are more difficult for a visitor to recognize and/or memorize, should they wish to do so.
- Keep your file names fairly short, but also consider the main keywords and sections of your site as file names. For instance, the About Us page should be named about.html. An image of a red rose should be **redrose.gif**, not **image283.gif.**
- Check your spelling! Misspelled file names can drive you crazy after you check, and recheck, and check again your code for the reason why the image or file is not showing in the browser.

Web Coding Steps

Using the Text Editor

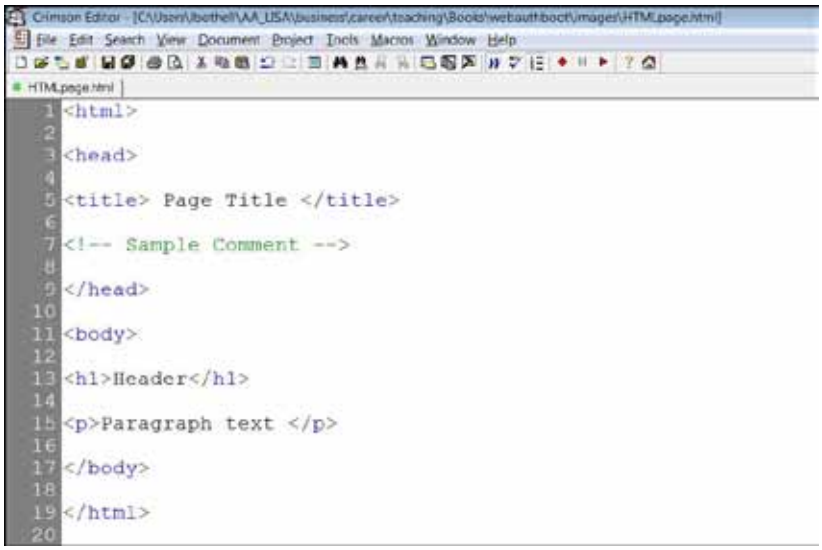
Text editors are pretty easy to use, and this is where you will do your hand coding. Just open your text editor of choice, and:

- For a new web page, create a new document and SAVE it as an .html file immediately before starting to work on it.
- The .html extension will activate any code coloring the editor offers that you'll need to see to help you find errors.
- Make sure to save the file in the web project directory where you plan all the HTML files to go.
- Show *Word Wrap*, so your long lines of code will wrap on your screen.
- Show *Line Numbers*, so that when you compare validation problems to your code you can match lines.
- Find out which command you need to activate to view work you have done in a browser.

To start your basic web page,

- Begin code in a saved file in the open window.
- Make sure you have open and closing html tags for every page - <html> </html>.
- Make sure you have open and closing head tags for every page, after the opening html tag - <head> </head>.
- Make sure you have an open and closing title tag inside your head section - <title>Page Name </title>.
- Make sure you have open and closing body tags for every page - after the closing head tag and before the closing html tag - <body> </body>.
- Make sure you have placed some content to view inside your body tags, such as a open/closing header, open/closing paragraph, etc.

Text Editor Page



Reviewing Your Code Output

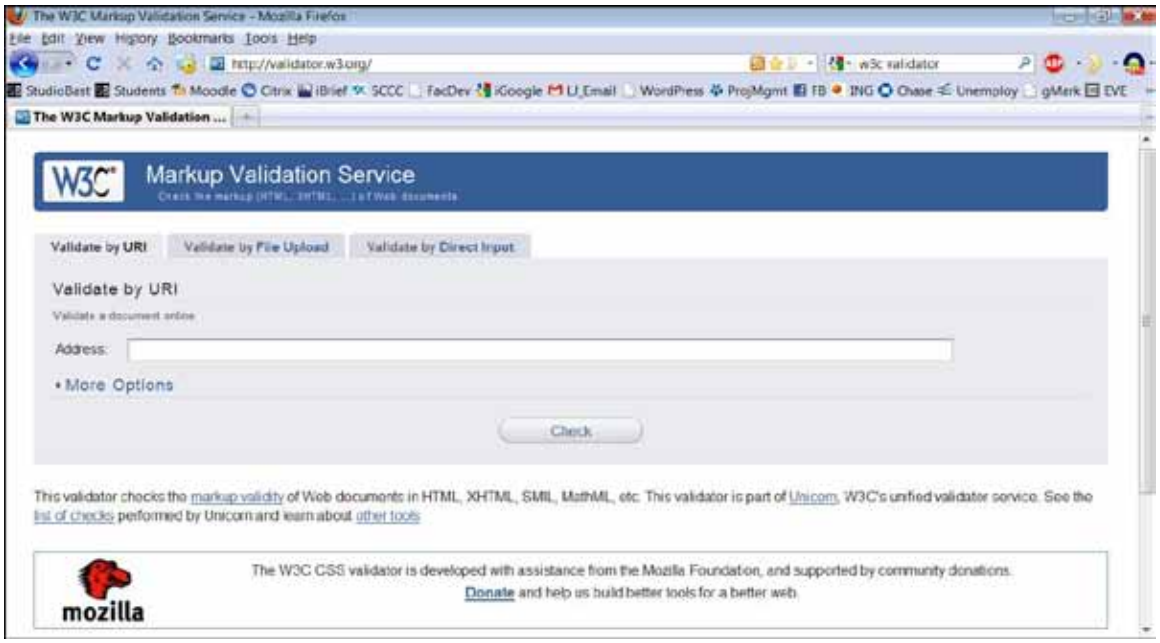
You will need to check your code output frequently to make sure you are getting the results you want.

- Use the text editor's *View in Browser* option to see your web page in your computer's default browser.
- If your text editor does not have that command, open your web browser, and under *File* use the *Open File* option to locate and open your .html file.
- When you make code changes, return to the browser view of the page and refresh it to see the code changes. F5 is the refresh hotkey.
- **Tip**: Keep the same browser window open, rather than repeatedly opening new versions every time you make a code change you want to review.

Validating Your Code

After you have completed coding a page to your satisfaction, you need to run it through the W3C Markup Validation Service (W3C Validator). This helps you standardize your code so that your web page is compliant with the standards used by various browsers – so that your pages will look and work right.

W3C Validator



To use the W3C Validator:

- Open your web browser and go to http://validator.w3.org/.
- Choose the 'Validate by Direct' Input tab.
- Copy your full web page code from your text editor and paste it inside the input space provided on the W3C validator page.
- Click the 'Check' button below the input space (you may need to scroll down the Validator page to see it).

The W3C Validator will check your code and provide a new pages, with the results.

- A **red** bar that reads "Errors found while checking this document as HTML or XHTML (whatever version)!" means that your code has errors, and that you need to scroll down the page to see them.
- A **green** bar that reads "This document was successfully checked as HTML or XHTML (whatever version)!" means that your code has validated. You are authorized to add the W3C icon on your web page if you wish.

The W3C Validator results can be very hard to decipher for newcomers and experts alike, so the W3C maintains a list of error messages and their interpretation: http://validator.w3.org/docs/errors.html

If you get a bunch of error messages, refer to **Axiom #4.**

Axiom #4: Don't Panic!

Common validation errors include:

- **Doctype**: A doctype declaration is mandatory for most current markup languages and without one it is impossible to reliably validate a document. One should place a doctype declaration as the very first thing in an HTML document and be sure to use the correct one for the coding being done.
- **No Character Encoding Found**: An HTML document should be served along with its character encoding. You can choose from a number of encodings, though "iso-8859-1" (Western Europe and North America) and "**utf-8**" (Universal, and more commonly used in recent documents) are common encodings if you are not sure which to choose.
- **Flash or embedded media does not validate**: Many Flash authoring tools rec-

ommend, or enforce, the usage of the <embed> element to include flash animations or applications in Web pages. <embed> is not part of any standardized version of HTML, and this practice produces invalid markup.

- **The validator complains about "&" or other symbols in code**: You will need to replace the & and other complaint symbols with the HTML character code that stands for it. For instance, & = *&*

For other common errors, be sure to check the http://www.htmlhelp.com/tools/validator/problems.html and the http://validator.w3.org/docs/help.html#faq-doctype pages.

QuickLook

Errors vs. Warnings

Validation warnings must be corrected before the web page code is considered valid to HTML standards. However, you may get the full validation and still have warnings. Why?

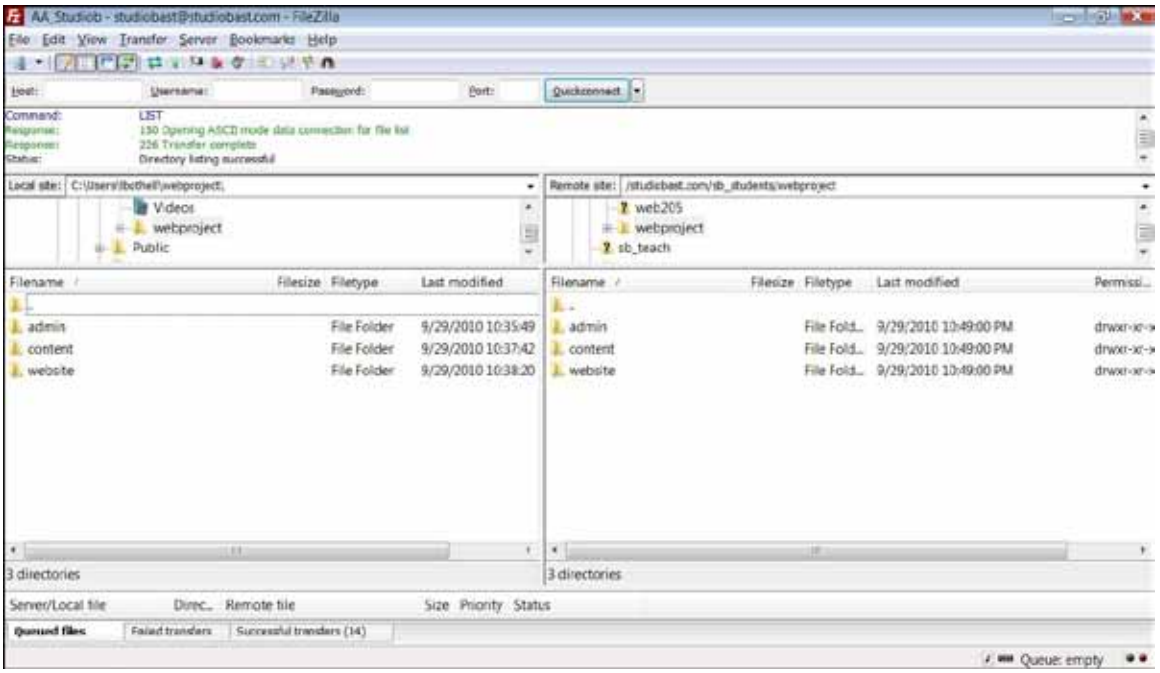
W3C Validator notes and warnings highlight missing or conflicting information which caused the validator to perform some guesswork prior to validation, or other code affecting the output below the warning item. If the guess or fallback is incorrect, it could make validation results entirely incoherent. It is highly recommended to check these potential issues, and, if necessary, fix them and re-validate the document.

Uploading Your Page

The File Transfer Protocol (FTP) program will allow you to upload your files from your local computer to the Web host you choose. Each FTP program has interface differences, but here is the gist of the process:

- Open the FTP program.
- Input your web host, username, and password that you received or set up with the Web host you chose. Example: Host = mydomain.com, Username= mylogin, Password = mypassword.
- You can likely ignore the Port unless your web host informed you to choose a specific one.
- Choose the *connect* option (button or on the toolbar of the FTP program).

An FTP Interface



The FTP program will then try to connect with the Web host computer, and when it does so, your local computer directories will appear in the left pane, and the remote computer directories you will transfer to will appear in the right pane.

- Choose the specific file folder you want to transfer files from on your local left side, and open the corresponding folder on your remote right side.
- Left and right pane folder views must be **exactly the same** on each side before you transfer files.
- On your left page (local), choose the file you want to transfer and right-click on it. The context menu should give you the choice to upload the file. You can also just drag the file from the local pane to the same space in the remote pane.
- Always remember to transfer any associated images subfolder with the images inside over to the web host computer. Same with any other subdirectories you use in your web project.
- Remember, file names should ALWAYS be all **small** text - no capital letters.

To test your recent uploads, Open your web browser, type the address of your website, and when your page loads, be sure to *refresh* (F5) to load your newest changes.

If you have problems getting an FTP connection, you might have active/passive mode issues. FTP can be run in active or passive mode, which determines how the data connection is established. In active mode, the client sends the server the IP address and port number on which the client will listen and the server initiates the TCP connection. In situations where the client is behind a firewall and unable to accept incoming TCP connections, passive mode may be used, where the client sends a PASV command to the server and receives an IP address and port number in return, which the client uses to open the data connection to the server. You can try:

- Checking the Internet Explorer FTP setting: Most PCs come with Internet Explorer (IE) as the default browser, and IE defaults to passive FTP. If you are having connection problems with your FTP program, check the Tools/Internet Options/ Advanced, Browsing segment, and **uncheck** *Use Passive FTP*.
- Increase the FTP Timeout default: An FTP program may have a default timeout set to 30 seconds, and occasionally the web host computer needs a little more time. If you are having timeout issues, check this setting in your FTP program and set it to 150 (seconds).
- Create an active profile for your website in your FTP program: An FTP program should let you set a reusable website profile, Open the program's Site Manager tool, and create a site with your web host name, login, and password. Then, in the Site Manager's Transfer Settings area, make sure to choose "Active".

8

HTML/XHTML Basics

HTML (Hypertext Markup Language) is the starting code language for building the most basic of web pages. You need to really know and understand this to understand and apply more complex languages and scripts for making later web pages dynamic. The current version is HTML4.

XML (Extensible Markup Language) is a set of rules for encoding documents in machine-readable form. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications. XML's design goals emphasize simplicity, generality, and usability over the Internet. It is a textual data format with strong support via Unicode for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures, such as in web services.

XHTML (Extensible Hypertext Markup Language) moves past HTML to the more precise, manageable, and World Wide Consortium (W3C) validated current standards for basic web coding. It is a blend of HTML and XML which results in organized and consistent HTML code, allows clean validation, and keeps you away from deprecated tags and quirky practices that give browsers (and you) huge headaches. The current version is XHTML 1.0.

HTML5 is currently being used in some part and the boundaries of browser interpretation pushed as far as designers and developers can go. HTML4 and XHTML 1.0 in terms of additional functionality for websites that is now provided by Flash, JavaScript, and plug-ins. HTML5 adds more markup syntax for documents, especially for defining sections of a web page. It includes a bunch of new tags that this book will not cover for beginner's use because, as hot as HTML5 looks for more web "devsigner" websites and mobile platforms, it is currently not fully supported at a consistent and reliable level by the current browsers. However, we will look at the non-developer aspects of HTML5, like the doctype and charset which are supported by the W3C Validator and Google/ search engine indexing. We will also look at what can be reliably added to your web work at this point, and offer resources for you to keep your eye on – tutuorials, examples, standardization reports, etc.

Our aim in this book is to focus on XHTML 1.0, because it meets current standards for

web browsers and validation, and also requires web authors to be professional and accurate with their code. We will also add in some of the mark-up oriented HTML5 that is reliable, in order to get a leg up on the upcoming changes that is being adopted by browsers.

Therefore, coding practices we will cover will focus on **XHTML1.0**, and added HTML5 mark-up basics. We will not integrate HTML5 section classes other than as a segment of advanced HTML (there are ongoing excellent tutorials for you to explore on your own as HTML5 evolves), and we will not support the coding of deprecated (discarded) HTML4 and earlier formats.

Note: Even though we will focus on XHTML Transitional, this book will refer to tags as HTML-related tags. This is because XHTML is simply stricter HTML and uses the current HTML tags. Should you move to XHTML Strict, your coding will need to be even tighter or will face more validation errors.

Doctype

The doctype is critical to the build of a web page, because it tells the browser and validator which version of HTML is being used. This is necessary since browsers and the validation process need this information to interpret and display accurate web pages.

There are several current doctypes:

HTML 4.01 Strict

This DTD contains all HTML elements and attributes, but does not include presentational or deprecated elements (like font). Framesets are not allowed.

```
<!doctype HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.
w3.org/TR/html4/strict.dtd">
```

HTML 4.01 Transitional

This DTD contains all HTML elements and attributes, including presentational and deprecated elements (like font). Framesets are not allowed.

```
<!doctype HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

HTML 4.01 Frameset

This DTD is equal to HTML 4.01 Transitional, but allows the use of frameset content.

```
<!doctype HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://
www.w3.org/TR/html4/frameset.dtd">
```

XHTML 1.0 Strict

This DTD contains all HTML elements and attributes, but does not include presentational or deprecated elements (like font). Framesets are not allowed. The markup must also be written as well-formed XML.

```
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.
w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

XHTML 1.0 Transitional

This DTD contains all HTML elements and attributes, including presentational and deprecated elements (like font). Framesets are not allowed. The markup must also be written as well-formed XML. We will focus on this doctype in Section 2 because its use seems more balanced for new web authors.

```
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

XHTML 1.0 Frameset

This DTD is equal to XHTML 1.0 Transitional, but allows the use of frameset content.

```
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

XHTML 1.1

This DTD is equal to XHTML 1.0 Strict, but allows you to add modules (for example to provide ruby support for East-Asian languages).

```
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.
w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

HTML5

There are 3 different doctypes in HTML 4.01 and 3 in XHTML 1.0. In HTML 5 there is only one:

```
<!doctype HTML>
```

Currently, the normal doctype for new and developing websters to start with is XHTML 1.0 Transitional, which helps browsers assume the web page has been designed using standards mode. It is also more forgiving that XHTML 1.0 Strict and allows for the occasional creep in of a gracefully depreciating tag. Designers then often move into XHTML 1.0 Strict, but you may find it easiest to work into the simple HTML5 doctype as soon as possible.

HTML Tag Namespace

XML namespaces are used for providing uniquely named elements and attributes in an XML document. They are defined in Namespaces in XML, a W3C recommendation. An XML instance may contain element or attribute names from more than one XML vocabulary. If each vocabulary is given a namespace then the ambiguity between identically named elements or attributes can be resolved.

Because XHTML is a blend of XML and HTML, a namespace is also assumed. While HTML4 and earlier pages can use just the plain *<html></html>* tags, the opening HTML tag in all XHTML documents uses namespace information. Important validation note: This opening HTML tag must abut the end of your doctype, not be tabbed down to the next line in your code. Otherwise, the W3C Validator doesn't recognize the HTML opening tag properly, and you will get errors.

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

HTML5, however, does not require a stated namespace.

Head Section Charset

A character encoding (charset) is a method of converting bytes into characters, which addresses the issue of what abstract characters may be part of an HTML document. To validate or display an HTML/XHTML document, a program must choose a character encoding.

The basic charset is UTF-8, which is compatible with ASCII characters. Because the ISO character-sets are limited in size, and are not compatible in multilingual environments, the Unicode Consortium developed the Unicode Standard, which covers all the characters, punctuations, and symbols in the world. We will focus on UTF-8, especially since it is also the charset recommended for HTML5.

ISO-8859-1 is also commonly used, since it allows for characters to be used from North America, Western Europe, Latin America, the Caribbean, Canada, and Africa. However, it has limits compared to UTF-8.

Head Section Metadata

Metadata is loosely defined as information about data. Meta elements are HTML or XHTML elements used to provide structured metadata about a web page to browsers and search engines. These elements must be placed as tags in the head section of an HTML or XHTML document, and are always passed as name/value pairs. They may be used to specify page description, keywords and any other metadata, and most search engines use this data when adding pages to their search index.

Meta elements have significantly less effect on search engine results pages today than they did in the past, and their utility has decreased dramatically as search engine robots have become more sophisticated. This is due in part to the nearly infinite re-occurrence (keyword stuffing) of meta elements and/or to attempts by unscrupulous website placement consultants to manipulate (spamdexing) or otherwise circumvent search engine ranking algorithms.

HTML/XHTML documents still require several of these meta tags, the information from which is usually hidden to the visitors of the web page. Meta tags can include:

- Content type: This specifies that the page should be served with an HTTP header called “content-type” that has a value: “text/html’.
- Content StyleType: This specifies that the page should be served with an HTTP header called “content-style-type” content=”text/css”> in order to support Cascading Styles.
- Author: Informs search engines of the web page author’s name.
- Copyright: Informs search engines of the web page’s copyright date.
- Description: Specifies a description of a web page, and usually shows up in search engine results as the summary after the web page title.
- Keywords: This meta element identifies itself as containing the “keywords” relevant to the document. It has little effect on ranking at any of the major search engines today; however if the keywords used in the meta can also be found in the page copy itself, Yahoo still refers to this tag.
- Language: Tells search engines what natural language the website is written in (English, Japanese, or French), as opposed to the coding language (HTML).
- Robots: Supported by several major search engines, robot metadata controls whether search engine spiders are allowed to index a page, or not, and whether they should follow links from a page, or not.

9 Web Page Code Structure

Web code structure is very simple, even though a the code of a fully functioning web page looks confusing and complex. We're going to break things down, step by step, and you'll be coding in no time.

HTML works in a very simple and logical format. It reads like you do, from top to bottom, and left to right. That's important to remember. HTML is written with normal text. What you use to set certain sections apart as headings, subtitles, bold text, image links, etc is a series of what we call tags.

Think of tags as making your structure. Let's say you want a heading. You will put a tag at the exact point you want the heading to start and another tag where you want the heading to stop. If you want a specific word to be emphasized, you will place a start emphasis tag at the beginning of the word and an end emphasis tag at the end of the word. Make sense?

HTML Tag Building Blocks

HTML tags have three components that tell the browser what is to be displayed:

- **Element**: An HTML element is everything from the start tag to the end tag. Example: <**table** width="100%">
- **Attribute**: Provides additional information about HTML elements. Example: <table **width**="100%">
- **Value**: The value of the tag's attribute. The value must always be inside of opening and closing quote marks. Example: <table width=**"100%"**>

We'll take a good look at these three sections later, but first, a word about using the HTML coding tags. Remember these four things about using a tag:

- Open it
- Close it
- Use for specific purposes
- ALWAYS use all small text - no capital letters

Types of HTML tags

Basic Tags: These are tags of an open/close pair. These are used to mark the beginning and end of a bit of content, like a string or words, and must be placed around them so that browsers can parse what should happen. The closing tag looks a little different than the opening tag since a slash is placed before the tagname. For instance:

Open Tag | String of Words | **Close Tag**

```
<tagname>I am string data. </tagname>
```

Single Tags: These tags, of which there are very few, are stand-alone tags. In other words, they do not have an opening and a closing tag, because they have nothing they can wrap around. All the tag information is set inside a self-closing tag. In HTML the slash is not needed, but it is required in XHTML.

Open Carat | String of Data | **Close Carat**

```
<hr />  (a horizontal rule)
```

Comment Tags: HTML comment tags have a little more information, so that the browser knows that they and their comment is supposed to be hidden from the web page visitor. They are also self-closing tags, and have their own special format:

Open Carat and symbols | Comment text | **Symbols and closing carat**.

```
<!– I am a comment –>
```

Opening the Tag

All opening HTML tags start with a left carat, which looks like this: < .

Closing the Tag

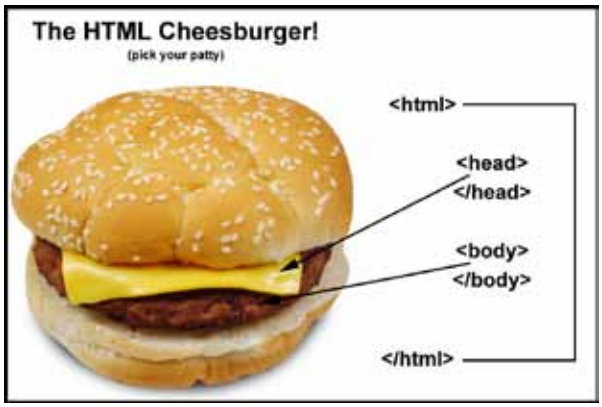
All opening HTML tags end with a right carat, which looks like this: > . In addition, the end tag also requires the use of a slash to indicate it is a closing, not opening tag: / . Closing tags have two primary formats:

- **Basic Tags**: For ending tags of an open/close pair. These are used to mark the end of a string of data, and look like this: </tagname>
- **Single Tags**: For stand-alone tag, which self-close. They look like <taginfo />. Note the space before the slash.

HTML Page Structure

The first thing to know about any HTML/XHTML page code structure is that, other than the doctype, all coding falls into one of three sections. This structure is built like a plain cheeseburger: a top and bottom bun, a slice of cheese below the top bun, and a patty (beef, chicken, garden) below the cheese and above the bottom bun.

- HTML - the bun
- Head - the cheese
- Body - the meat (or garden) patty



HTML Tag

Other than the doctype, **all** web page content **must** be placed in between an opening and closing html tag. It is considered the Root element of an HTML document; all other elements are contained inside this section. Otherwise, browsers and validation won't recognize the code as being for a web page. Drum roll, please!

```
<html>
All other web page coding here.
</html>
```

There you go. That's it.

Head – the hidden stuff

The head section of the web page code contains mostly hidden information about the page, called meta tags, title, script, and style. It looks like this:

```
<html>
<head>
Hidden info
</head>
All other web page coding here.
</html>
```

Body – the visible stuff

The body section of the web page code is where you will code for all the content the visitor will see – the banner segment, navigation segment, content segment, footer segment, and content segment sidebars.

```
<html>
<head>
Hidden info
</head>
<body>
All visual web page segments here.
</body>
</html>
```

QuickLook

First Web Page

Here is a super-basic web page, with a couple of important tags we'll be learning about in just a bit.

```
<html>
<head>
<title>First web page</title>
</head>
<body>
<p>I AM a web author!</p>
</body>
</html>
```

QuickLook

Real World Web Page

Of course, live web pages need the doctype (previous chapter) too:

```
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>First web page</title>
</head>
<body>
<p>I AM a web author!</p>
</body>
</html>
```

Head Section Contents

The head section of the web page requires various pieces of data about the web page. All of this is hidden from the visitor's view, but some of it does appear in ways that potential visitors will see in search engine results.

Title

All web pages in a website require a title. This title tells the search engines what the web page is, and appears in search engine results as the Link to the web page. It also appears on the browser tab opened to show the web page itself. Each page in a website can and should have its own specific title as needed. Use this with SEO and accessibility in mind – give a useful title name.

```
<title>My Web Page</title>
<title>Adopt-A-Furry-Friend: About Us</title>
```

Metatags

As discussed in the previous chapter, metatags provide search engines information about the web page. Every web page in your website should have metatags, but you can change some of the data, such as for robots, as needed. There have been many metatags in the past, and I am narrowing down to the currently most useful. Note that meta tags are **all self-closing** tags: <tagname />

In one form, meta elements can specify HTTP headers which should be sent by a web server before the actual content, for example:

```
<meta http-equiv="foo" content="bar" />
```

In the general form, a meta element specifies name and associated content attributes describing aspects of the HTML page. To prevent possible ambiguity, an optional third attribute, scheme, may be supplied to specify a semantic framework that defines the meaning of the key and its value. In this example, the meta element identifies itself as containing the foo element, with a value of bar, from the DC or Dublin Core resource description framework.

```
<meta name="foo" content="bar" scheme="DC">
```

Common tags

```
<meta http-equiv="content-type" content="text/html;charset=utf-8" />
<meta http-equiv="content-style-type" content="text/css">
<meta name="author" content="YourName" />
<meta name="copyright" content="2010 YourName" />
<meta name="description" content="Bring home a wonderful furry friend to
join your family!" />
<meta name="keywords" content=" YourName, pets, cats, dogs, adopt " />
<meta http-equiv="content-language" content="en" />
<meta name="robots" content="noindex, nofollow" /> Use this if you want
NO indexing and NO link pursuit.
```

Link References

You may want/need to refer to an external file, such as a style sheet or javascript library. Instead of using meta tags you would use a link tag. Note that link tags are **all self-closing** tags.

```
<link rel="style sheet" type="text/css" title="description_of_style"
href="filename.ext" />
```

A less-common, but important, usage is to supply navigation hints consistently through use of microformats. Several common relationships are defined, that may be exposed to visitors through the browser interface rather than directly in the web page.

```
<link rel="next" href=" filename.ext " />
```

Object

Used for including generic objects within the document header. Though rarely used within a head element, it could potentially be used to extract foreign data and associate it with the current document.

```
<object>...</object>
```

Script

Can act as a container for script instructions or link to an external script with the optional src attribute. Also usable in the document body to dynamically generate either both block or inline content. This would be used, for instance, to refer to JavaScript:

```
<script type="text/javascript">...</script>
<script type="text/javascript" src="filename.ext">...</script>
```

Style

The style tag refers to the use of Cascading Styles (CS) in a web page, and refers to embedded header section script.

```
<style type="text/css"> ...</style>
```

Note: In HTML 5, all scripts are assumed to be type="text/javascript" so when you choose the HTML5 doctype and use script, you can leave out the type="text/javascript" unless your script is something other than JavaScript.

Comments as Needed

Your head section is a great place to add comments to yourself or other web authors about the web page, such as the colors in your selected color palette, the login information for the FTP uploading (minus the password, of course), general notes another author might need to see, etc.

```
<!– http://www.yourdomain.com/ –>
<!– FTP: yourdomain.com/   Login:yourlogin –>
<!– To validate - http://validator.w3.org/#validate_by_input –>
```

Body Section Contents

The body section of a web page has a lot going on, and we'll discuss details in subsequent chapters. At this time we'll talk about the basic elements that allow your visitors to see text, images, links, and related content, then learn how to style and expand them later. All the elements and their attributes for tables, forms, frames, links, and images will have

their own chapters – here we'll just cover the basics.

The page layout that visitors see is based on how you decide to organize and structure the information for them in the web page. In the body code of the web page, you have two different type of tag elements that let you do this:

- **Block elements**: Block-level elements typically contain inline elements and other block-level elements. When rendered visually, block-level elements usually begin on a new line.
- **Inline elements**: Inline elements typically may only contain text and other inline elements. When rendered visually, inline elements do not usually begin on a new line.

Block Elements

Block elements begin new blocks of visual material, like new paragraphs.

…
Creates a paragraph, perhaps the most common block level element.

<h1>…</h1> <h2>…</h2> <h3>…</h3> <h4>…</h4> <h5>…</h5> <h6>…</h6>
Section headings at different levels. <h1> is the the highest-level heading, <h2> the next level down (sub-section), <h3> for a level below, and so on to <h6>. Heading elements are not intended merely for creating large or bold text — they describe the document's structure and organization. Some programs use them to generate outlines and tables of contents.

Most visual browsers show headings as large bold Times font by default, though this can be overridden with CS.

<dl>…</dl>
A definition list (consisting of definition terms paired with definitions).

<dialog>…</dialog>

HTML5 only, partly supported for semantic use. For marking up conversations — a chat transcript, an interview, a bit dialog from a screenplay, etc. Uses the definition list parts inside the dialog wrap.

<dt>…</dt>
A definition term in a definition list or a dialog.

<dd>…</dd>
The definition of a term, in a definition list or a dialog.

<address>…</address>
Contact information for the document author.

…
An ordered (enumerated) list. The type attribute can be used to specify the kind of ordering. The default is Arabic numbering.

…
An unordered (bulleted) list. CS can be used to specify the list marker. The default marker is a disc.

…
A list item in ordered (ol) or unordered (ul) lists.

<blockquote>…</blockquote>
A block-level quotation, for when the quotation includes block level elements, like paragraphs. The cite attribute may give the source, and must be a fully qualified Uniform Resource Identifier (URI). The default presentation of block quotations in visual browsers is usually to indent them from both margins. This has led to the element being unnecessarily used just to indent paragraphs, regardless of semantics. For quotations not containing block level elements see the quote (q) element.

~~...~~
Marks a deleted section of content. This element can also be used as inline.

…
A block-level logical division. A generic element with no semantic meaning used to distinguish a document section, usually for purposes such as presentation or behavior controlled by CS.

<hr />
A horizontal rule, which appears as a grey line across a page, and can be used to visually separate other block elements. A stand-alone tag.

<ins>...</ins>
Marks a section of inserted content. This element can also be used as inline.

<script>...</script>
Places a script in the document. Also usable in the head and in inline contexts. Script is not itself either a block or inline element; by itself it will not display at all, but it can contain instructions to dynamically generate either both block or inline content.

<noscript>...</noscript>
Replacement content for scripts, which is used to offer replacement information if the web author's script isn't able to be viewed by visitors. Unlike script this can only be used as a block-level element.

<pre>…</pre>
Pre-formatted text. Text within this element is typically displayed in a non-proportional font exactly as it is laid out in the file. Whereas browsers ignore whitespace for other HTML elements, in pre, whitespace should be rendered as authored. This element can contain any inline element except: image, object, superscript, or subscript.

<audio>…</audio>

HTML5 only, partly supported. Allows you to define defines sound, such as music or other audio streams. You can write text between the start and end tags to indicate that older browsers not support this tag.

<video>…</video>

HTML5 only, partly supported. Allows you to define video, such as a movie clip or other video streams. You can write text between the start and end tags to indicate that older browsers not support this tag.

<canvas>…</canvas>

HTML5 only, partly supported. Used to display graphics – only lets you create a "container" for graphics. You must use a script to paint graphics inside of the canvas tagset.

<embed>…</embed>

HTML5 only, partly supported. Defines embedded content, such as a plug-in. Note: You can NOT write text between the start and end tags.

<source>…</source>

HTML5 only, partly supported. Specifies multiple media resources on media elements, such as <audio> and <video>. The browser should choose which file (if any) it has support for. You can write text between the start and end tags to indicate that older browsers not support this tag.

<figure>…</figure>

HTML5 only, partly supported for semantic use. Make image embedding more descriptive

and easier to recognize, and is also used for diagrams, code snippets and other visual elements. Used when referring to an image in the body of an article, not all images in general.

<time>…</time>

HTML5 only, partly supported for semantic use. Defines a time or a date, or both, in a way that works with search engine spiders as well as human readers. The datetime attribute allows you to display a nice date format inside the tag.

Note: We will look at HTML5 Structural Elements in a later chapter about advanced page layout.

Inline Elements

Inline elements work inside of blocks of visual material, like changes to parts of text inside of a paragraph.

<a>… link text
An anchor element is called an anchor because web designers can use it to anchor a URL to some text on a web page, as well as to link to another web page. When visitors view the web page in a browser, they can click the text to activate a same-page anchored link or visit another page whose URL is in the link.

<abbr>…</abbr>
Marks an abbreviation, and can make the full form available: <abbr title=”abbreviation”>abbr.</abbr>

<acronym>…</acronym>
Similar to the abbr element, but marks an acronym: <acronym title=”Hyper-Text Markup Language”>HTML</acronym>

A forced line-break. A stand-alone tag.

An image tag, which is a stand-alone tag. Allows you to insert a reference to an image location so that you can ‘embed’ and image on your page.

<dfn>…</dfn>
Inline definition of a single term.

…
Emphasis (conventionally displayed in italics). Replaces the deprecated italic tag.

…
Strong emphasis (conventionally displayed bold). Replaces the deprecated bold tag.

…
The tag provides no visual change by itself, but provides a way to add a hook to a part of a text or a part of a document. When the text is hooked in a span element you can add styles to the content.

Computer Phrase Inline Elements

These elements are useful primarily for documenting computer code development and visitor interaction through differentiation of source code (<code>), source code variables (<var>), visitor input (<kbd>), and terminal output (<samp>).

<code>…</code>
A code snippet. Conventionally rendered in a monospace font: Code snippet.

<samp>…</samp>
Sample output (from a program or script).

<kbd>…</kbd>
Keyboard - text to be entered by the visitor

<var>…</var>
Variable

<tt>…</tt>
Fixed-width font (typewriter-like), also known as teletype.

<bdo>…</bdo>
Marks an inline section of text in which the reading direction is the opposite from that of the parent element.

<cite>…</cite>
A citation. Reference for a quote or statement in the document.

~~…~~
Deleted text. Typically rendered as a strikethrough.

<ins>…</ins>
Inserted text. Often used to mark up replacement text for <del>'d text. Typically rendered underlined.

Note, both <ins> and <del> elements may also be used as block elements: containing other block and inline elements. However, these elements must still remain wholly within their parent element to maintain a well-formed HTML document. For example deleting text from the middle of one paragraph across several other paragraphs and ending in a final paragraph would need to use three separate <del> elements. Two <del> elements would be required as inline element to indicate the deletion of text in the first and last paragraphs, and a third, used as a block element, to indicate the deletion in the intervening paragraphs.

<q>…</q>
An inline quotation (for block level quotation see blockquote). Quote elements may be nested.

<q> should automatically generate quotation marks in conjunction with CS. Practical concerns due to browser non-compliance may force authors to find workarounds. The cite attribute gives the source, and must be a fully qualified URI.

<script>...</script>
Places a script in the document. Also usable in the head and in block contexts.

_… and […]
Mark subscript or superscript text

Deprecated Elements

Deprecated elements, or tags, were used in past versions of HTML and XHTML. However, due to continued standardizing, they are no longer supported by some browsers or validation. **Do not use!!!**

Block Level

<basefont>…</basefont>
The <basefont> tag specifies a default font-color, font-size, or font-family for all the text in a document.

<dir>…</dir>
A directory listing. The original purpose of this element was never widely supported; deprecated in favor of <ul>.

<menu>…</menu>
A menu listing. Should be more compact than an <ul> list, but badly supported. Deprecated in favor of <ul>.

…
Creates a block-level center-aligned division. Deprecated in favor of or another element with centering defined using style sheets.

<applet>…</applet>
Embeds a Java applet in the page. Deprecated in favor of <object>, as it could only be used with Java applets, and had accessibility limitations.

Inline

As visual presentational markup only applies directly to visual browsers, its use is discouraged. Cascading styles should be used instead. Several of these elements are deprecated or invalid in HTML 4 / XHTML 1.0, and the remainder are invalid in the current draft of XHTML 2.0. The current draft of HTML 5, however, includes , , and <small>.

…
Sets font to boldface where possible. ... usually has the same effect in visual browsers, as well as having more semantic meaning.

…
Sets font to italic where possible. ... usually has the same effect in visual browsers, as well as having more semantic meaning.

<big>…</big>
Increases font size (bigger text).

<small>…</small>
Decreases font size (smaller text).

<strike>…</strike> (deprecated) and <s>…</s> (deprecated)
Strike-through text.

<u>…</u>
Underlines text.

…
…

10 Text Coding

Now, on the nitty-gritty of handling the text formatting options you have with HTML/XHTML. While we have looked at the basic tags, let's get into more detail and see how coding text content works.

Body Section Heading Elements

The six heading elements, H1 through H6, represent section headings. Although the order and occurrence of headings is not constrained by the HTML DTD, documents should not skip levels (for example, from H1 to H3), as converting such documents to other representations is often problematic.

The headings default to a bold version of whatever the default font of a visitor's computer is, usually Times or related serif.

H1: Bold, very-large font (approximately 24pt), centered, block-level. One or two blank lines above and below. Good for website identifier title in the banner segment.

```
<h1>I am Heading 1.</h1>
```

H2: Bold, large font (approximately 18pt), flush-left, block-level. One or two blank lines above and below. Good large subhead for web page identifier title in the body segment.

```
<h2>I am Heading 2.</h2>
```

H3: Italic, large font (approximately 14pt, slightly indented from the left margin, block-level. One or two blank lines above and below. Good smaller subhead for web page subsection headings in the banner segment.

```
<h3>I am Heading 3.</h3>
```

H4: Bold, normal font (approximately 12pt), indented more than <h3>, block-level. One blank line above and below. Good smaller subhead for web page subsection headings in the banner segment.

```
<h4>I am Heading 4.</h4>
```

H5: Italic, normal font (approximately 10pt), indented as <h4>, block-level. One blank line above. Good for small font uses, like image captions, footer address info, etc.

```
<h5>I am Heading 5.</h5>
```

H6: Bold, small font (approximately 8pt), indented same as normal text, more than <h5>, block-level. One blank line above. Good for small font uses, like image captions, footer address info, etc.

```
<h6>I am Heading 6.</h6>
```

These headings were created to separate blocks of information – headings and subheadings in report-style formatting.

Body Section Basic Texts

Paragraph

The <p> element indicates a paragraph, block-level. The exact indentation, leading space, etc. of a paragraph is not specified in the default use and may be a function of other tags, style sheets, etc. Typically, paragraphs are surrounded by a vertical space of one line or half a line.

```
<p>I am a paragraph.</p>
```

Bold and Italics

These two inline tags allow you to add emphasis to text, and must be used inside a text element like one of the headings or paragraph. There is no longer an underline tag, so that is one form of emphasis you may not use;

```
<p>I am a sentence with <strong>bold</strong> and <em>italic </em> text
in it.
<h2 I am an already default bold header with <em>italics</em> in it.</h2>
```

Blockquote

The block-level <blockquote> element is used for text quoted from another source. A typical rendering might be a slight extra left and right indent, and/or italic font. The <blockquote> typically provides space above and below the quote. A block of quoted text should normally be preceded by some introductory text that makes it clear that a quotation follows. It should be followed by an attribution of some kind, preferably with an exact citation of the source preceded by an expression like “Source:”

```
<blockquote>I am text that is indented on the right and left, but I am not the
right tag for this purpose.</blockquote>
```

An accurate use of blockquote, with the proper use of the cite:

```
<blockquote cite="URL">
The actual quotation.
</blockquote>
<p>Source:<cite><a href="URL" title="title of information:>Title</a></
cite>.</p>
```

Quote

The block-level <q> element is what you want to use if you just plan to indent with plain HTML tags. Unfortunately, it has not been supported in Internet Explorer, which is why the <blockquote> started being commonly used for indenting instead.

```
<q>I am text that is indented on the right and left and look like a blockquote,
but I can be used for this purpose.</q>
```

Pre

The block-level <pre> element represents a character cell block of text and is suitable for text that is being formatted for a monospaced font. The <pre> tag may be used with the optional width attribute. The width attribute specifies the maximum number of characters for a line and allows the HTML visitor agent to select a suitable font and indentation.

Within preformatted text:

- Line breaks within the text are rendered as a move to the beginning of the next line.
- Elements that define paragraph formatting (headings, address, etc.) must not be used.

```
<pre>I am preformatted text and pretty much look like the Courier font. </
pre>
```

Address

The block-level <address> element contains such information as address, signature and authorship, often at the beginning or end of the body of a document. Typically, it is rendered in an italic typeface and may appear indented.

```
<address>
Studiobast<br />
<a href="mailto:us@studiobast.com">Email us</a><br />
Address: Box 123, Seattle<br />
Phone: 1-555-1234
</address>
```

Text Spacing

You have simple ways you can space text, like sentences more than one space apart from each other, and paragraphs more than one line away from each other.

Single Space

There is no tag for adding space; instead, HTML allows you to add an invisible space character: * *

```
<p> I am the first sentence.    I am the second sentence. </p>
```

Break

The inline
 element specifies a line break between strings of data which will drop data after it onto the next line. For example:

```
<p>I am a paragraph with some text on this line and <br />
Some text on the next line. However, all of me is a block element because I
am a paragraph. </p>
```

Horizontal Rule

The block-level <hr /> element is a divider between sections of text; typically a full width horizontal line or equivalent graphic.

Body Section Text Lists

Lists, which are block-level elements, are a way of emphasizing text by indenting it with some form of bullets, like discs or numbers. All lists must contain one or more list elements. These can be great for listing resource links, steps in an activity, and providing definitions. Lists have two required tag types: the kind of list (unordered, ordered, defini-

tion) and the tag to *list* each separate item. Also, lists do not work with paragraph tags – you do not place a list inside any other text tags, or they will not validate correctly.

Unordered Lists

An unordered list uses the tags, like a wrapper. Each list item starts with the <li> tag. The list items are marked with bullets (typically small black circles).

```
<p>My favorite winter beverages are:</p>
<ul>
<li>Mocha</li>
<li>Latte</li>
<li>Cocoa</li>
</ul>
```

How the unordered list HTML code above looks in a browser:

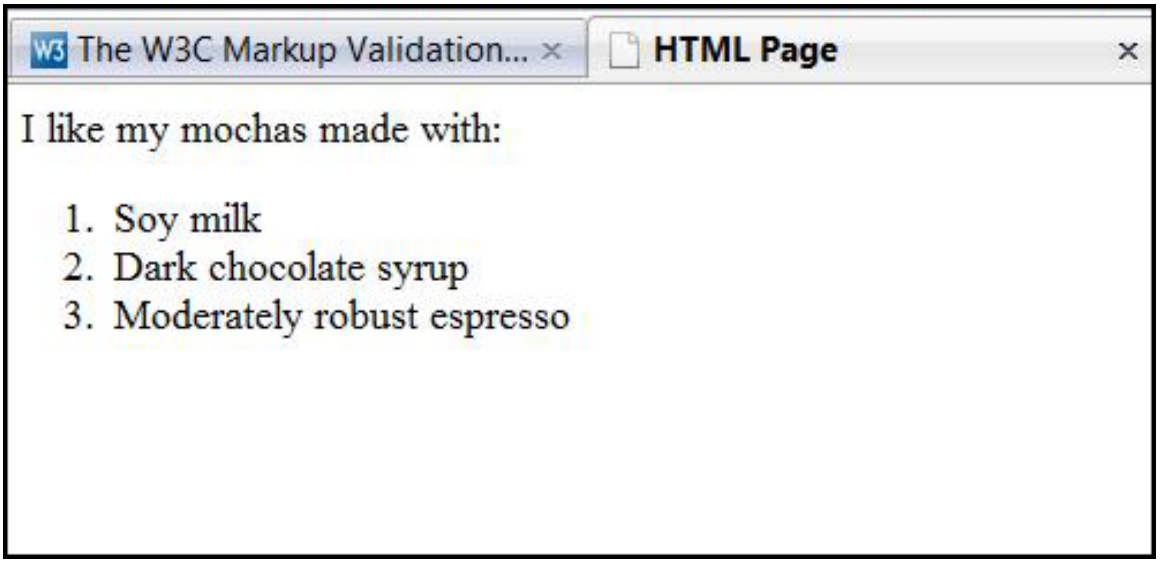
Ordered Lists

An ordered list uses the tags, like a wrapper. Each list item starts with the <li> tag. The list items are marked with numbers.

```
<p>I like my mochas made with:</p>
<ol>
<li>Soy milk</li>
<li>Dark chocolate syrup</li>
<li>Moderately robust espresso</li>
</ol>
```

How the ordered list HTML code above looks in a browser:

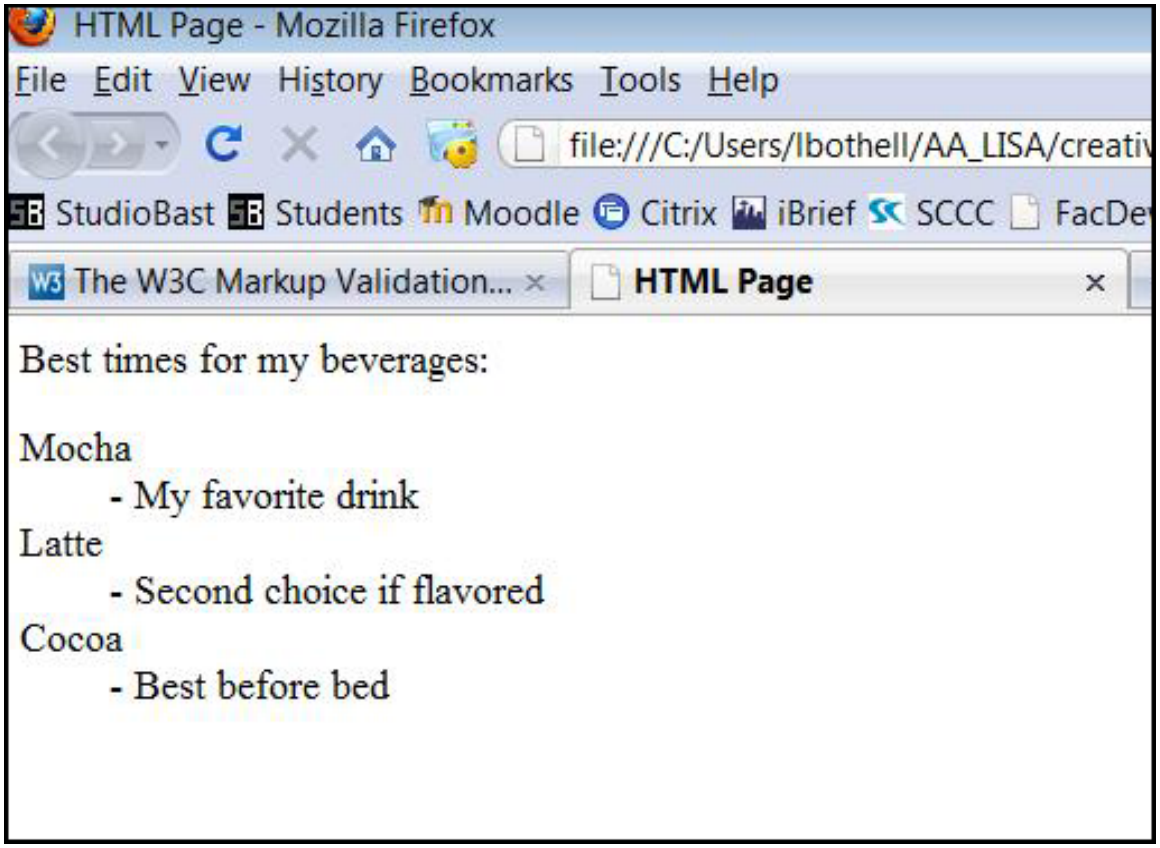
Definition Lists



A definition list is a list of items, with a description of each item. The <dl></dl> tags defines a definition list, like a wrapper. The <dl> tag is used in conjunction with <dt> (defines the item in the list) and <dd> (describes the item in the list):

```
<p>Best times for my beverages:</p>
<dl>
<dt>Mocha</dt>
<dd>- My favorite drink</dd>
<dt>Latte</dt>
<dd>- Second choice if flavored</dd>
<dt>Cocoa</dt>
<dd>- Best before bed</dd>
</dl>
```

How the HTML code above looks in a browser:



Body Section Nesting Tags

Nesting tags properly means that when using more than one pair of tags to affect code at the same time, you open and close these pairs in the right order as related to the other pairs. Whaaaaat?

The number one reason that you need to master nesting is if you're going to use cascading styles (CS), which require tags to be consistently nested within the document so that it can tell where styles begin and end. If you set up a style that would affect all bold and italic text on your page, the incorrect nesting would make it hard for the browser to know where the bold and italics ended and where the text was just italicized.

Nesting also effects accessibility and browser compatibility. If your HTML is incorrectly nested, it won't be as accessible to screen readers and older browsers – it will look bad. It also will not validate, earning head shaking from web authors of the world.

Full Nesting

The easiest way to understand nesting is to think of your HTML tags as little wrappers (or nests) holding your content (like chocolates). When you purchase chocolates, they typically come in a wrapper. Then the checker places the wrapped chocolates in a bag, and then you place that bag in your vehicle to get home.

The chocolates are your HTML content, such as text or images. And each wrapper is an tag around that content. You cannot put the chocolates halfway into the wrapper and then into the bag - this physically doesn't work. But you can write your HTML so that the tags are opened but not closed around the text.

If you have a block of text that you want bold, italic, and inside a paragraph, you'll have 3 HTML elements and the text itself.

```
This is my sentence about chocolates.
```

The sentence of text is your "chocolates". Since you want the whole sentence to be bold, you would make that tag your "chocolate wrapper":

```
<strong> This is my sentence about chocolates.</strong>
```

As you can see, the strong tags surround the entire block of text, so the entire sentence will be bold. Then you want the sentence to have italics too, so that tag would be the "bag" holding the wrapped chocolates:

```
<em><strong> This is my sentence about chocolates.</strong></em>
```

The em tags completely surround the strong tags and the sentence. However, for HTML, you still need the block-level text container. This final container is the paragraph or "vehicle":

```
<p><em><strong> This is my sentence about chocolates.</strong></em></p>
```

This is my sentence about chocolates

See how we close the tags in the opposite order that we opened them? Reading from left to right, the tags are opened: p, em, strong. So when you need to close them you need to close them in **reverse** order: strong, em, p.

Varied Nesting

You won't always want a whole block element, like a whole sentence, to use completely the same formatting – you might, for instance, want all bold but only partly italicized. You just have to keep track of what starts when and close the right tag for the currently opened tag. Huh?

```
<p><strong> This is my <em>sentence</em> about chocolates.</strong></
p>
```

This is my *sentence* **about chocolates.**

Start simply for nesting, and learn it well, because you can nest whole paragraphs, content areas, and will need to properly nest tags in building tables. Just think of chocolates.

QuickLook

Nesting Problems

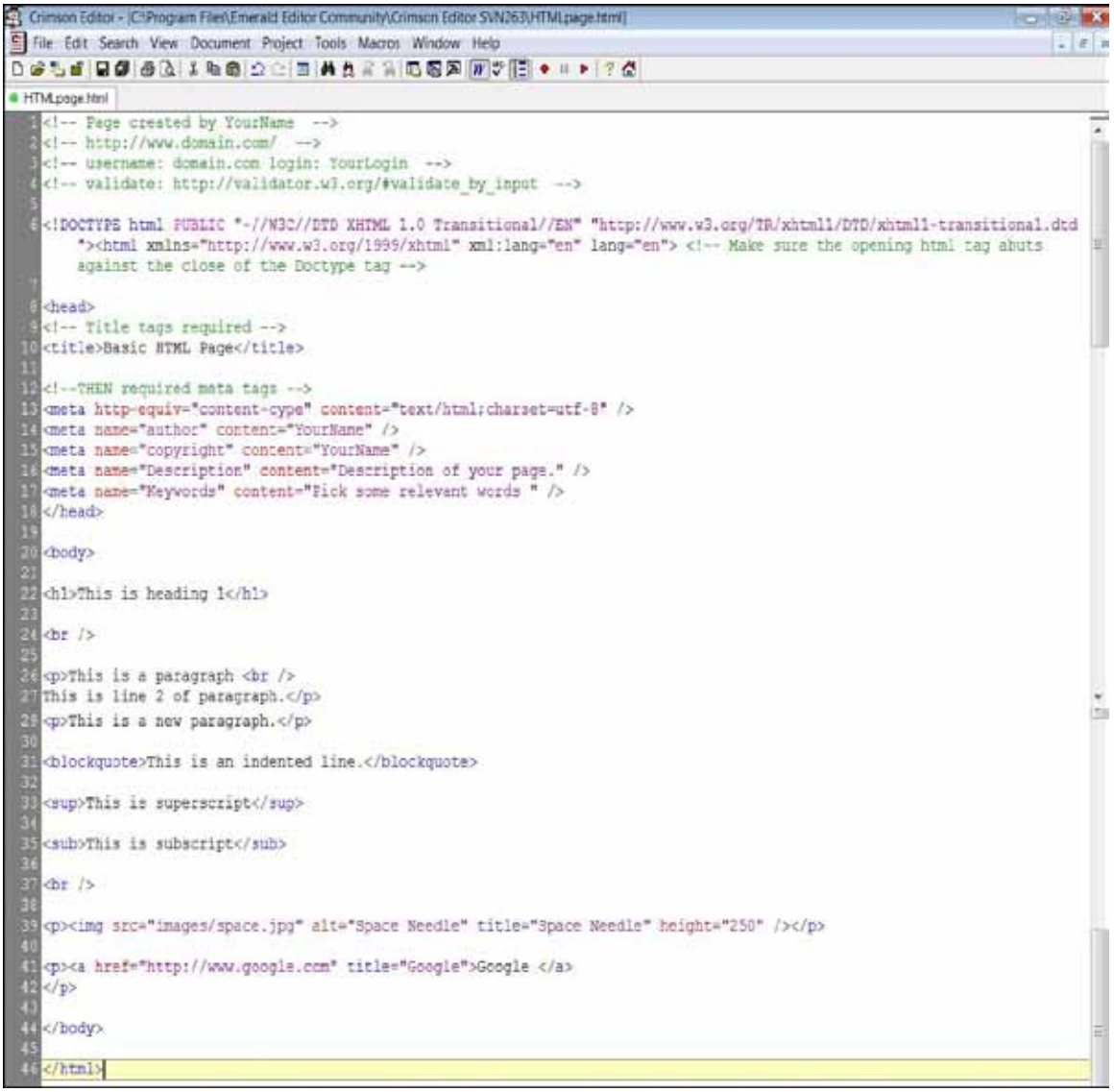
```
<p><em>Morning is a <strong>great</em> time for coffee.</em></p>
<ul>
<li>Vanilla coffee</li>
<li>Hazelnut coffee</li>
<li>Chocolate coffee</ul>
</li>
```

Text Styles

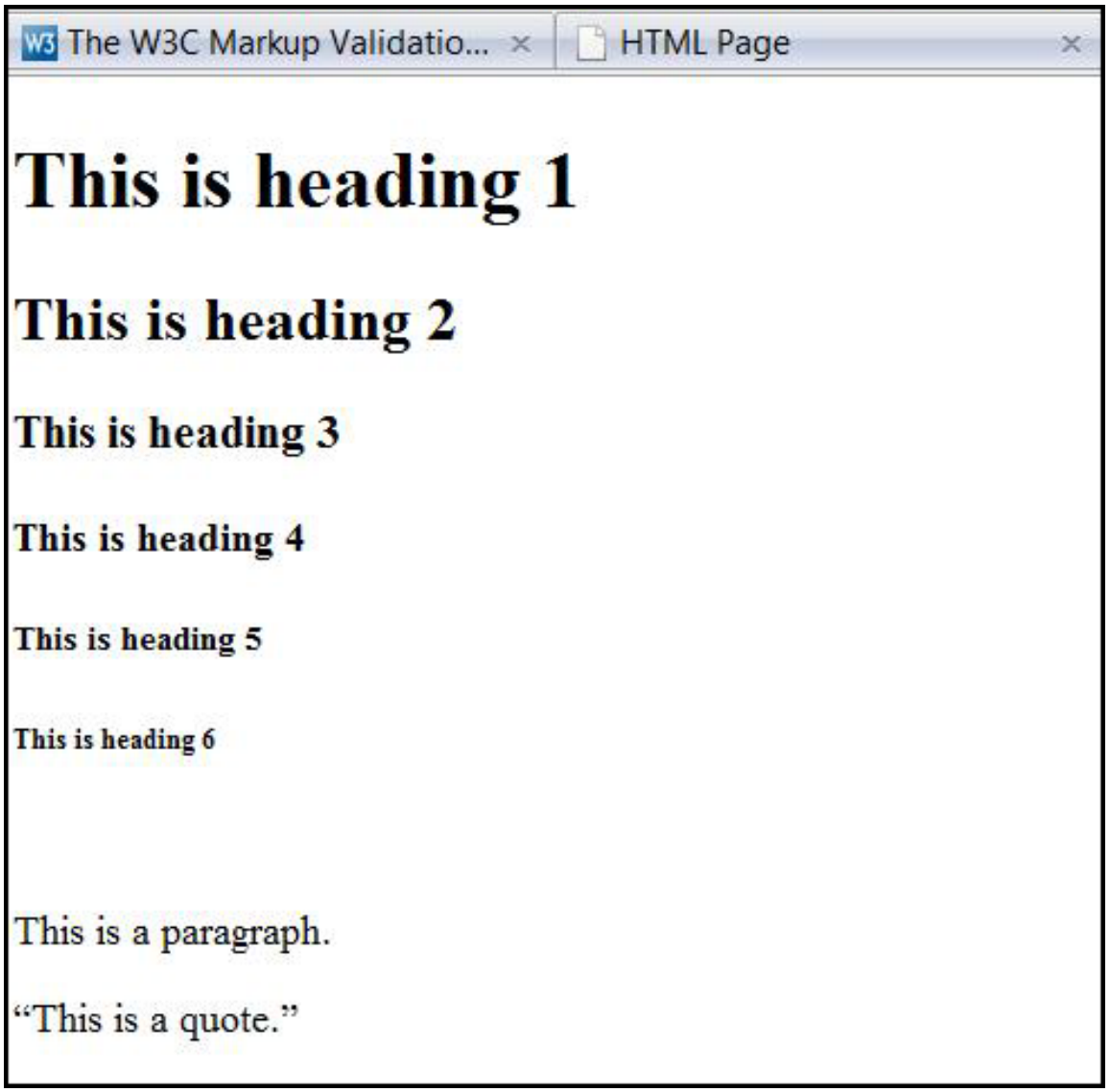
Okay, now you know a few things you can do with text in terms of the basic HTML tags. BORRRRINNNNGGGG! You want color, you want size, you want fonts, you want pizzazz! You want it now!

So sorry, not just yet. We've got more basic coding to go over first. We will go over the principles of inline cascading styles in a later chapter, and if you decide to work on that chapter now, I won't slap your hands. Have fun!

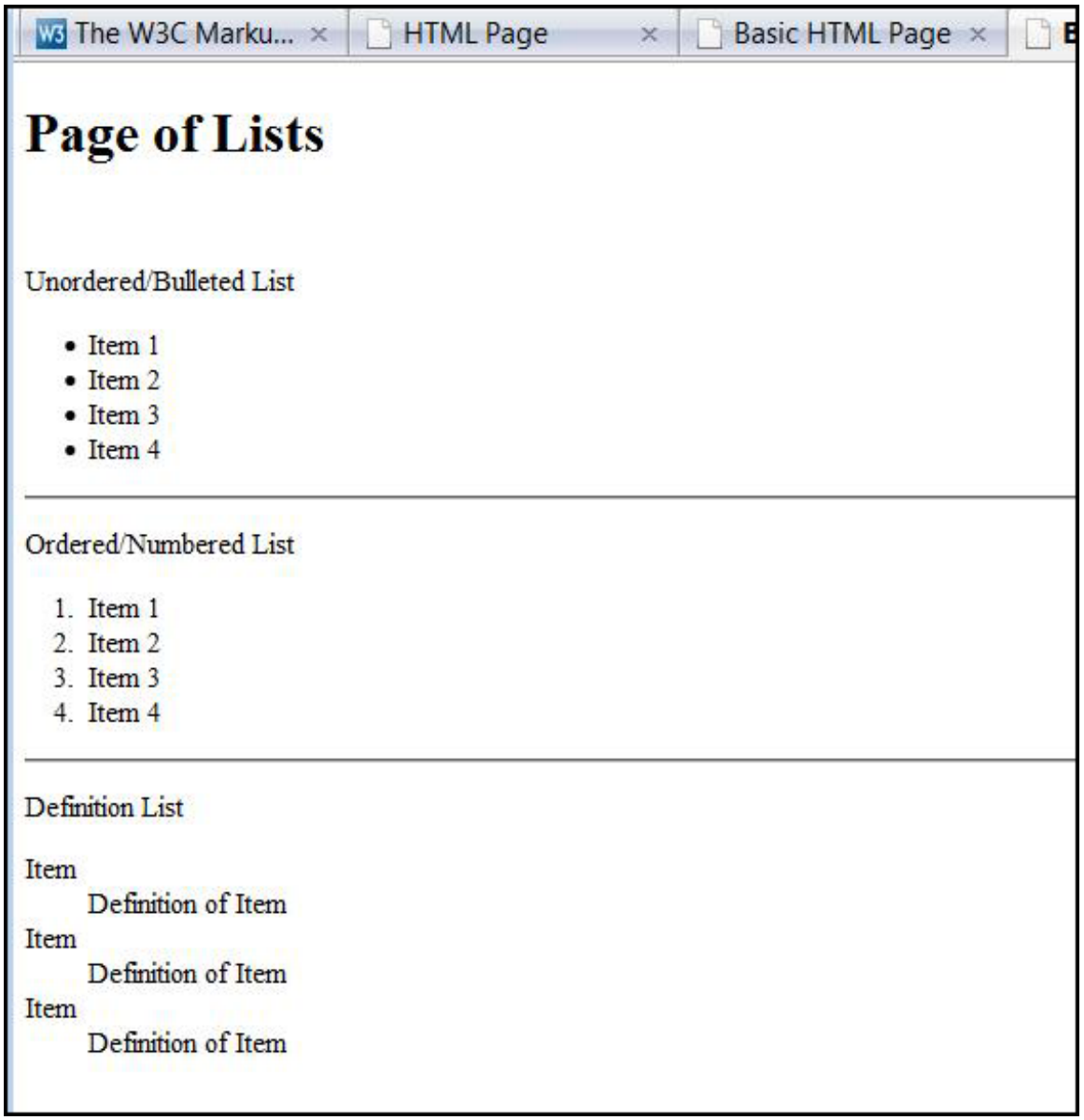
What We Learned



Try it Out


The W3C Markup Validatio...
HTML Page
This is heading 1
This is heading 2
This is heading 3
This is heading 4
This is heading 5
This is heading 6
This is a paragraph.
“This is a quote.”

Try it Out


The W3C Marku...
HTML Page
Basic HTML Page
Page of Lists
Unordered/Bulleted List
Item 1
Item 2
Item 3
Item 4
Ordered/Numbered List
1. Item 1
2. Item 2
3. Item 3
4. Item 4
Definition List
Item
Definition of Item
Item
Definition of Item
Item
Definition of Item

11 Images

What fun would the Web be without images? They have so many uses in web pages: illustration, navigation, backgrounds, image maps. Let's go over some of what they can do, and while we'll cover more on cascading styles later, we will touch on a little of those now so we can explore image options.

Image Element

The inline <img /> element is one of the few self-closing tags in HTML, and can only be used in the body section of web pages. This is because the image tag does not wrap around anything like text, but is itself the element that visitors see. Notice that images are not technically inserted into an HTML page, but rather are linked to HTML pages. The <img> tag creates a holding space for the referenced image that appears to be embedded in the page.

```
<img />
```

The image tag has important attribute/value pairs that make it actually display on the web page:

Source Attribute

The src attribute is required in order for the browser to know where to find the image to be embedded. The value of this is the image file's actual location, relative to the location of the page the image will be embedded on.

```
<img src="imagename.ext" />
```

Alt Text

The alr attribute is meant to be used as an alternative text if the image is not available, not as a mouse-over text. To show a mouse-over text on images or image-maps, use the title attribute. The alt attribute is required for validation and for web page accessibility. If a visitor cannot see the image, because it is slow to load or it is missing, the alt value – the info you give about the image – in place of the image on the web page. However, it does

appear on the web page as a tool tip when the cursor hovers over the image in Internet Explorer.

```
<img src="whitecat.jpg" alt="White Cat" />
```

Title Text

The title attribute is used for web page accessibility, like the alt attribute. It also appears on the web page as a tool tip when the cursor hovers over the image, but is only seen in this way in the Firefox browser.

```
<img src="whitecat.jpg" alt="White Cat" title="White Cat" />
```

Style Attribute

The style attribute allows you to use inline cascading style rules to style how the image will appear – specific width, height, alignment, with a border, etc.

```
<img src="whitecat.jpg" alt="White Cat" title="White Cat"
style="width:200px; float:left; border:1px solid black;" />
```

Body Section Image Use

Block-Level Graphics

Images are by nature inline elements, which means that they do not specifically push the code after it to start on a new line (though they can make text around them look messy). However, many images that the visitor sees on a web page, like a product picture or an image of a person in a staff bio, look like block-level graphics. This is achieved by placing the image tag inside a heading, paragraph, table, or other block-level element.

```
<p><img src="whitecat.jpg" alt="White Cat" title="White Cat" /></p>
```

Block-Level Navigation Links

Images used as a link can appear either inline or at a block-level if coded within a block-level element. For instance, when creating vertical navigation links, such as down the left side of the page, you might want to use images for visitors to click, instead of just words. Keep in mind that these will look most effective when they are consistent in style and size.

You would place the linking image inside of a table cell, a paragraph, or some other

block-level element so that the images for each link appear to be on one line after the next in a vertical menu.

```
<p><a href="whitecat.html"><img src="whitecat.jpg" alt="White Cat"
title="White Cat" /></a></p>
<p><a href="blackcat.html"><img src="blackcat.jpg" alt="Black Cat"
title="Black Cat" /></a></p>
```

Inline Graphics

You can place images inside of block-level elements in a way that makes them look like they are on the same line as the rest of, say, the paragraph text, or in columns of a table. This is also a technique used to place navigation linking images side by side is a horizontal menu.

```
<p>
<a href="whitecat.html"><img src="whitecat.jpg" alt="White Cat"
title="White Cat" />    
<a href="blackcat.html"><img src="blackcat.jpg" alt="Black Cat"
title="Black Cat" /></a>
</p>
```

Background Graphics

Images can be used as background in a table, table row, table cell, the body of a whole web page, as a background, etc. We'll cover more about how this works when we talk about styling.

```
<body style="background-image:url(path/img.ext); "> The rest of the body
coding</body>
```

Favicons

Have you seen a website where there is a tiny image to the left of the URL in the browser address bar? Check out Starbucks and Adobe for examples. A favicon is a graphic image (icon) associated with a particular web page and/or website. Many recent user agents (such as graphical browsers and newsreaders) display them as a visual reminder of the website identity in the address bar or in tabs.

To add a favicon to your website, you'll need both an image and a method for specifying that the image is to be used as a favicon. However, the format for the image you have chosen must be 16x16 pixels or 32x32 pixels, using either 8-bit or 24-bit colors. The for-

mat of the image must be one of .png, .gif, or .ico. This image should be stored in either the directory your actual HTML pages are in, or in your images folder.

```
<link rel="icon" type="image/png" href="images/myicon.png" />
```

Image Qualities

When you include images in your web project, you need to be sure you prepare them for accessible, lightweight, and effective use.

Importing Images

When you bring an image into your web page, your link to where the image file resides tells the browser which image to load when the page is visited. The physical size of an image is based on the file size – the number of bytes, based on the PPI (pixels per inch) that make up the image. The standard for fast and efficient web image upload is **72ppi.**

There are four image types which are consistently used for web pages:

- **JPG**: This format stores a single raster image in 24-bit color in a platform-independent format that supports the highest levels of compression; however, this compression is lossy. The level of file compression can be increased or decreased, but image quality is sacrificed for file size. When you edit and resave .jpg files, you compound the degradation of the original picture data, and this degradation is cumulative.
- **GIF**: This format stores single raster image data or multiple raster image data in 8-bits, or 256 colors, and support transparency, compression, interlacing, and multiple-image pictures. This compression is LZW compression, at a roughly 3:1 ratio, and supports lossless compression.
- **PNG**: This format stores a single raster image at any color depth, and is a platform-independent format that supports high-level lossless compression and alpha channel transparency. Older browsers and programs may not support .png files.
- **ICO**: This is a format used only for favicons, as described above.

Each format has its use, and .png images are truly cool because they can have a transparent background, which eliminates the white box background that other images load with. Whichever image format you choose, your images should be absolutely no larger than 100kb in file size, and ideally should be well under 50kb.

Visual Sizing

The size your image appears in the web page is only partly dependent on the image's ppi and file size. If you link the image at its native file size, it can be very large or very small, depending on the ppi and file size. However, you can visual size changes in your code when absolutely necessary. The rule of thumb, however, is to closely estimate the size the image needs to be in your web page, and use an image editor to save a web-ready copy at the ppi that shows that visual size. 72 ppi/dpi is the web standard.

- **Width**: You can use inline cascading styles to change the visual image width. When you do this without changing the height, the image will resize proportionally.
- **Height**: You can use inline cascading styles to change the visual image height. When you do this without changing the width, the image will resize proportionally.

Here are a few common image dimension sizes:

- **Wide skyscraper**: 160 x web page width
- **Skyscraper**: 120 x web page width
- **Half-page ad**: 300 x web page width
- **Full banner**: 468 x 60
- **Half banner** : 234 x 60
- **Micro bar**: 88 x 31
- **Button 1**: 120 x 90
- **Button 2**: 120 x 60
- **Vertical banner** : 120 x 240
- **Square button**: 125 x 125
- **Leaderboard**: 728 x 90
- **Medium rectangle**: 300 x 250
- **Square pop-up**: 250 x 250
- **Vertical rectangle**: 240 x 400
- **Large rectangle**: 336 x 280
- **Rectangle**: 180 x 150
- **Thumbnail**: 100x100
- **Small Thumbnail**: 50 x 50

Image Placement

Where you place images, in relation to your page and text, has a lot to do with how clean and viewable your web page will look. You can control image placement is these ways:

- **Alignment**: An image can be aligned, either alone or in context of the text it is placed within, by styling its alignment. You will use inline cascading styles to *float* the image, either to the left, center, or right. Left is the default, but when placing an image in text, specifying left will standardize your image alignment for any browser that needs explicit directions. Example: <img style="float:left;">
- **Tables**: You can create a tabular table, and use table cells to align images by placing the image link inside the cell. This method has some limitations, especially if your images have different dimensions from each other. However, this method also has the positive effect of letting you style the table cell padding and margin to clarify some image placement.
- **Divs**: Division tags are currently the ultimate layout styling tool in cascading styles, and will be covered in Section 3. However, web authors who move away from use of tables in their code will graduate to using div and span elements for laying out all parts of their web pages, including the image placement. Later, folks.
- **Background repeats**: Images that you use in the background of table, table row, table cell, and div elements have their own special cascading style attributes depending on how you want them to appear. You can have an image repeat itself to the right, to the bottom, both directions, or not repeat at all. In other words, you can use a small image as a background, like a paw print, and have it repeat so that it looks like wallpaper. Or, you can have a large background image not repeat at all, so that it shows only once.
- **Note**: The original image attributes of align=, vertical align=, and are no longer allowed in validating XHTML code. More web author head shaking here.

Image Maps

An image-map is an image with clickable areas. Web authors sometimes use this to create a splash page image, or to make more than one part of a specific image, like an advertisement, clickable for different results.

The <map> tag is used to define a client-side image-map. The name attribute is required in the map element. This attribute is associated with the <img>'s usemap attribute and

creates a relationship between the image and the map. The map element contains a number of area elements, that defines the clickable areas in the image map.

<area>..</area>
Specifies a focusable area in a map.

<map>…</map>
Specifies a client-side image map.

Methods to Use

As an image with clickable pieces. You might do this if you make one long navigation menu bar with the page identifier words built into the image. Your visitor needs to click on those words, but can't get sent to the linked page because the words are part of the image.

```
<body>
<img src="menu.gif" width="400" height="80" alt="Main Menu"  title="Main
Menu" usemap="#menumap" />
<map name="menumap">
 <area shape="rect" coords="20,5,40,30" href="about.html" alt="About Us"
/>
 <area shape="rect" coords="60,5,80,30" href="products.html" alt="Coffee
Products" />
</map>
<body>
```

As a large image loaded in small parts. You can have a large image, say for an enlargement of a gallery thumbnail, that will load slowly at full page size. You can instead load this image in pieces by 'gridding' it, which is to slice it into several equally sized pieces and loading each one as a separate image.

You would use an image editing program to slice a copy of the large image into pieces, say 2x2 inch pieces, and then load those images into a borderless table in your web page. In the example below, one of the loaded images has been brought in as a link, which means that this part of the image only will be clickable.

```
<table cellspacing="0" cellpadding="0" style="border:0;" > <!– These are not
deprecated –>
 <tr>
<td><img src="images/mochasl_01.gif" alt="m01" /></td>
<td><img src="images/mochasl_02.gif" alt="m02" /></td>
</tr>
<tr>
<td><a href="http://www.starbucks.com" target="_blank"><img
src="images/mochasl_03.gif" alt="m03" style="border:0;" /></a></td>
<td><img src="images/mochasl_04.gif" alt="m04" /></td>
</tr>
<tr>
<td><img src="images/mochasl_05.gif" alt="m05" /></td>
<td><img src="images/mochasl_06.jpg" alt="m06" /></td>
</tr>
</table>
```

Objects

HTML's multimedia features allow authors to include images, applets (programs that are automatically downloaded and run on the user's machine), video clips, and other HTML documents in their pages.

Previous versions of HTML allowed authors to include images (via <img>) and applets (via <applet>), but these methods had limits and are now deprecated for object use. Instead, the object element offers an all-purpose solution to generic object inclusion.

<object>…</object>
Includes an object in the page of the type specified by the type attribute. This may be in any MIME-type the visitor agent understands, such as an embedded HTML page, a file to be handled by a plug-in such as Flash, a Java applet, a sound file, etc.

```
<body>
<p>Here's a close-up of my Starbucks mocha:
<object data="mocha.png" type="image/png">
This is a <em>close-up</em> of my Starbucks mocha.
</object>
</body>
```

12 Links

A hyperlink (or link) is a word, group of words, or image that you can click on to jump to a new document or a new section within the current document. When you move the cursor over a link in a web page, the arrow will turn into a little hand. Links are what make web pages what they are.

Links are specified in HTML using the <a> tag, which can be used in two ways:

- To create a link to another document, by using the href attribute
- To create a bookmark inside a document, by using the id attribute

The <a> element indicates a hyperlink anchor (see section Hyperlinks). At least one of the name and href attributes should be present. Attributes of the <a> element:

- **Href**: Gives the URI of the head anchor of a hyperlink.
- **ID**: Gives the name of the anchor, and makes it available as a head of a hyperlink. For XHTML.
- **Name**: Gives the name of the anchor, and makes it available as a head of a hyperlink. For HTML, old.
- **Title**: Suggests a title for the destination resource.
- **Rel**: This attribute gives the relationship(s) described by the hyperlink. The value is a whitespace separated list of relationship names. The semantics of link relationships are not specified in this document.
- **Rev**: Same as the rel attribute, but the semantics of the relationship are in the reverse direction. A link from A to B with rel=”X” expresses the same relationship as a link from B to A with rev=”X”. An anchor may have both rel and rev attributes.
- **Urn**: Specifies a preferred, more persistent identifier for the head anchor of the hyperlink. The syntax and semantics of the urn attribute are not yet specified.
- **Methods**: Specifies methods to be used in accessing the destination, as a whitespace-separated list of names. The set of applicable names is a function of the scheme of the URI in the HREF attribute. For similar reasons as for the Title attribute, it may be useful to include the information in advance in the link. For

example, the HTML visitor agent may choose a different rendering as a function of the methods allowed; for example, something that is searchable may get a different icon.

```
<a href="URL" title="Webpage Title">Webpage Title</a>
```

Title Attribute for Links

Accessibility/usability requires adding a title attribute to all links, just like the alt and title attributes are added to images.

In addition, the title attribute is where you can apply SEO techniques. On one hand, you can simple offer a quick one or two-word bit of information about what the link is, or you can add reasonable SEO-enhancing description.

```
<a href="URL" title="About Us"> About Us</a>
<a href="URL" title="About Company Name and Our Team"> About Us</a>
```

Kinds of Links

- **Absolute URLs**: An absolute URL is a full http address for a website somewhere in the Internet, since this is the only way you can find the address. You would use it for links external to your website, and inside your own website for submitted form redirect pages. It is the fully typed out URL address.
- **Relative URLs**: Relative links are for pages in your own web project, and they link to these files relative to where they are in relation to the page being linked from. They contain partial link information: href="images/imagename.ext"
- **Email links**: This operates the same way as normal links, but instead of linking to a web page, you link to an email address: href="mailto:emailname@domain.com"
- **Anchor links**: While all links are anchors, anchoring links are usually to a specific point in either the same document as the link, or to a specific place in another document in your website. They are used to shortcut the jump from the link to the information, such as using a links of section links on a long FAQs page, or jumping from a reference on one page to its citation on a reference listing page. <a id="top">
- **Images**: Images are not linked using the <a> tag, but pulled in with the <img> tag. However, because you are still asking your code to show an image somewhere in your website, *the linking locations below apply*.

Different types of links

Link Locations

Linking only works when you clearly code where the item being linked to actually is. Sometimes you will link to a web page on the Internet; much of the time, you will link to other pages in your own website, Occasionally, you will even link to another segment of the same page you are linking from.

The key is to know in what directory the page you are working on is, and the folder, sub-folder, and/or just the file name you are linking to.

- **Same directory**: If you are linking from your index.html page to the about.html page in the same directory, you are moving in the same level. This link is easy: About Us
- **Moving down a level**: You would use this when you are linking to a file in a subfolder of the directory you are currently in, or one level below you. The usual

example would be linking to an image in your images subfolder:

- **Moving up a level**: You would do this if you were inside a subfolder of your website and needed to link to a file that was outside your folder – in essence, one level above you. Say you are in one of the About Us web page's secondary navigation files, like Staff Bio, and wanted to get back to the About Us page:
About Us
- **Moving up multiple levels**: This works the same way as moving up a level, except that you jump up more than one level by adding another **../**
About Us
- **Email address**: Email Us!

Linking on the same page is a different animal. This is an anchor link, and instead uses the id attribute. You would use this if you had coded a long page, and ever couple of screen scrolls down you wanted a Back to Top link that, when clicked, would send the viewer back to the top of the page. This code is actually a 2-step process – setting the anchor that is being linked to, and placing the link to that anchor:

XHTML Version

- Anchor - Top of the Page
Link - Back to Top

HTML Version

- Anchor - Top of the Page
Link - Back to Top

QuickLook

Linking Images

One good way to make sure a visitor can find your website's home page from wherever s/he happens to be in your site is to link the logo to the home page. Your logo is usually in the banner/identifier segment of every page, and you can wrap the logo image inside a link:

```
<p><a href="index.html" title="Furry Friends Home Page">
<img src="fflogo.gif" alt="FF logo" title="FF logo" />
</a></p>
```

Link Output

So, you have links. Where do the linked items open? On the Web, you've seen pages open in place of the page with the link you just clicked. In other cases, it opens in a new window or tab.

Output forms include:

- **Self**: When a visitor clicks a link without a target attribute, the new page will open over the page s/he is on. If s/he starts looking at that new website, s//he might not hit the browser's back button and return to your website – a lost visitor.
- **Window**: When a visitor clicks a link with a target attribute, the link usually opens in a new window, so that two windows are now open – your website and the linked page. This can help you keep visitors in the vicinity of your website, unless they get frustrated with too many open windows.
- **Tab**: If a visitor has a browser set to 'open new tabs', or chooses to do that from a right-click context menu on the link, the new page will open in a new tab in the same window. This is efficient but many browsers just default to new windows.

Target

The target attribute is currently supported in all major browsers, and is used to tell the link where the linked page should open. Sites are using target="_blank" for links to make sure they will open the link in a new window and keep the visitor on the site, rather than potentially losing the visitor if s/he doesn't use the browser's back button to return.

The values include:

- **_blank**: Open the linked document in a new window. Opens a new window every time it is clicked.
- **_new**: This will only spawn one new window and every link clicked with a target value of _new will replace the page loaded in the previously spawned window.
- **_self**: Open the linked document in the same frame as it was clicked (this is default)
- **_parent**: Open the linked document in the parent frameset
- **_top**: Open the linked document in the full body of the window
- **Framename**: Open the linked document in a named frame

Unfortunately, the target attribute is not recognized in strict doctype XHTML, so you need another option. You can instead use limited JavaScript to simply open the link in a new window.

```
<script type="text/javascript">
<!–window.open('URL to open','window name','attribute1,attribute2') –>
</script>
```

Visitor Choice

Many web professionals think that Internet ethics require you to let the website visitor decide how s/he wants to open the links. Web savvy individuals often do not like new windows since they know how to open a new window or tab when they want. Less experienced visitors might not know that you can right-click on a link to get a context menu with those choices. Some visitors might be filling out a form on a page and don't want to lose their work to check something else out on the website. Some visitors don't want to be bothered and want you to give them that convenience built in.

If you decide to go with not giving your links a target, keep in mind a couple of cases when you should still consider it.

- When opening a PDF or some kind of media: This can take more time than people like, so opening it in a new window lets them keep perusing your site while they wait.
- If you have anything linked on a form or info-gathering page that would be canceled out if the visitor navigated away.
- If you really need to visitor to keep your page active while visiting another page.
- If you are opening a link to someone else's website from your own.

13 Tables

The HTML table model allows web authors to arrange data -- text, preformatted text, images, links, forms, form fields, other tables, etc. -- into rows and columns of cells. Tables are block-level elements which are defined with the <table> tag, and can only be used in the body section of web pages. They cannot be placed inside any other block elements except DIVs.

A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, other tables, etc. If you do not specify a border attribute, the table will be displayed without borders.

Table rows may be grouped into a head, foot, and body sections. Authors may also group columns to provide additional structural information. Cells may span multiple rows and columns.

Proper tag nesting is critical to successful tables.

<table>…</table>
Identifies a table. Several attributes are possible in HTML Transitional, but most of these are invalid in HTML Strict and must be replaced with style sheets. The summary attribute is however informally required for accessibility purposes, though its usage is not simple.

<tr>…</tr>
Contains a row of cells in a table.

<th>…</th>
A table header cell; contents are conventionally displayed bold and centered.

<td>…</td>
A table data cell, multiples of which make up the appearance of columns.

<colgroup>…</colgroup>
Specifies a column group in a table.

<col> or <col/>
Specifies a column in a table.

<caption>…</caption>
Specifies a caption for a table.

<thead>…</thead>
Specifies the header part of a table. This section may be repeated by the visitor agent if the table is split across pages (in printing or other paged media).

<tbody>…</tbody>
Specifies a body of data for the table.

<tfoot>…</tfoot>
Specifies the footer part of a table. Like <thead>, this section may be repeated by the visitor agent if the table is split across pages (in printing or other paged media).

Table Uses

Tables are meant to display rows and columns of data, like in a spreadsheet. They can be used for:

- Columns/rows of items like names and addresses
- Layout of product pictures and descriptions
- Minor layout of images in a grid, like thumbnails
- To create a simple background box that expands with content in it.
- Web page segment layout to control the areas of the banner, navigation, content, and footer areas. Note: Using tables for page structure is a good starter guide for new web authors, but should be upgraded to cascading styles layout formats as you gain more experience.

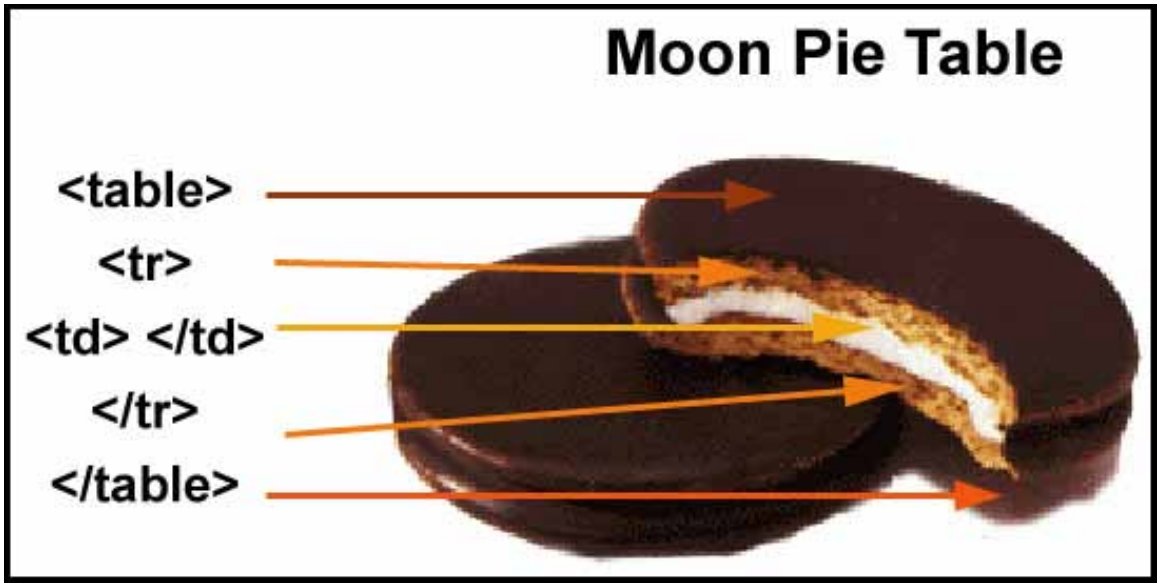
Table Structure

The essential table structure includes a table wrap, a row, and a cell inside the row, all placed in the body section of the web page code. The opening and closing tags work like the other HTML opening and closing tags. Data like text, images, links, lists, etc. can only be placed *inside* of table cells. You can think of a table like a moon pie.

Table shell

Like HTML tags wrap an entire web page, the table tags wrap the whole table.

```
<table>Rows and cells here</table>
```



Moon Pie Table

Table rows

The table row is the HTML table element that wraps around table cells. Cells cannot be displayed in a table unless they are nested inside of rows.

```
<table>
<tr>
Table cells here
</tr>
</table>
```

Table cells

The table cell is the space in which data, like text and images, can be placed.

```
<table>
<tr>
<td>I am table cell #1</td>
<td>I am table cell #2</td>
</tr>
</table>
```

Table Headers

You can use the table header tag to use the default bold and center cell content format to give your table "columns" specific heading styles. This <th> would be used in only the first table row, and the remaining rows would use the <td> tags to present the data in its normal style.

```
<table>
<tr>
<th>I am table header #1</th>
<th>I am table header #2</th>
</tr>
<tr>
<td>I am table cell #1</td>
<td>I am table cell #2</td>
</tr>
</table>
```

Expanded Tables

If you are creating complex tables of data for your web page, you will want to expand your table structure by adding the header, body, and footer tags. The <thead> tag is used to group the header content in an HTML table. The tbody element is used to group the body content in an HTML table and the tfoot element is used to group the footer content in an HTML table. Note: **<tfoot> must appear before** <tbody> within a table, so that a browser can render the foot before receiving all the rows of data.

Table Example

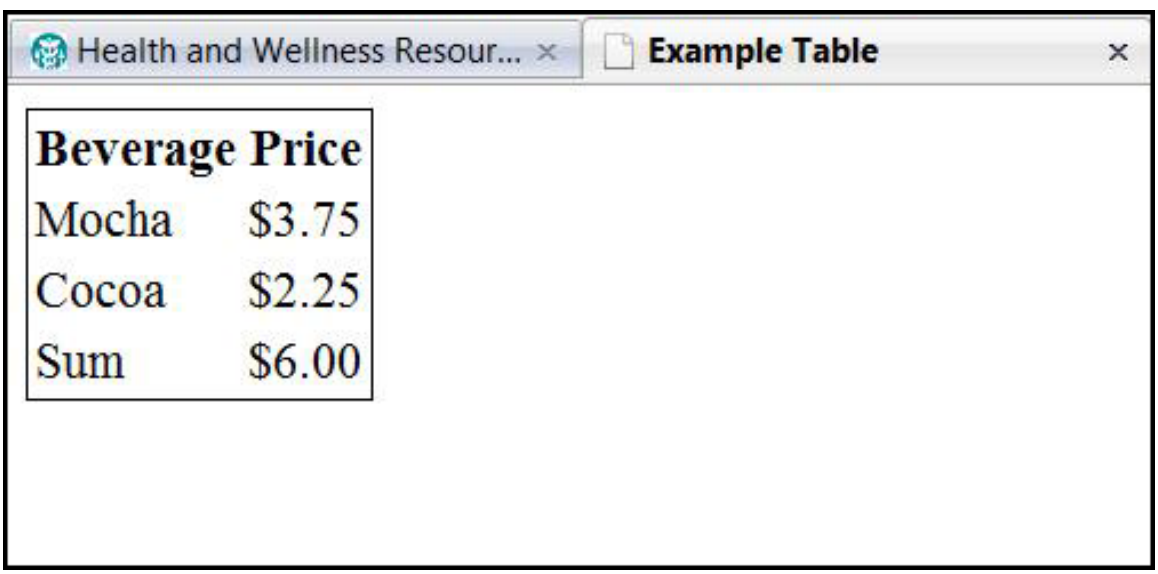


Table Code

```
<table style="border:1px black solid;">
 <thead>
  <tr>
   <th>Beverage</th>
   <th>Price</th>
  </tr>
 </thead>
 <tfoot>
  <tr>
   <td>Sum</td>
   <td>$6.00</td>
  </tr>
 </tfoot>
 <tbody>
  <tr>
   <td>Mocha</td>
   <td>$3.75</td>
  </tr>
  <tr>
   <td>Cocoa</td>
   <td>$2.25</td>
  </tr>
 </tbody>
</table>
```

Table Groups

The <colgroup> tag is used to group columns in a table for formatting, and is useful for applying styles to entire columns, instead of repeating the styles for each cell, for each row. This can be great if you want a different background color for one column, or the data to be right aligned on one column and left aligned in the other. Note that the colgroup tags of each "column" **need to be specified** before the rows of the table.

```
<table style="width:100%; border:1px black solid;">
 <colgroup style="width:30%;"></colgroup>
 <colgroup style="width:70%; color:#ccc;"></colgroup>
 <tr>
  <th>Beverage</th>
  <th>Size</th>
 </tr>
 <tr>
  <td>Mocha</td>
  <td>50oz</td>
 </tr>
</table>
```

Table Cell and Row Spans

Cells may span several rows or columns. The number of rows or columns spanned by a cell is set by the rowspan and colspan attributes for the TH and TD elements. This can be helpful for tables where your first row acts as a singled header for several columns below it. Or when you want to make a single column span two rows to insert a larger image.

Colspan

The colspan is short for column span, which means coding a table data cell to stretch the length of its own column plus one or more next to it.

When you stretch a column to take the space of two in its own row, you define that it actually takes the place of the second table data cell. This is why you see only two table data cells coded in row one of the table below.

Rowspan

The rowspan is short for row span, in which you code a table data cell to stretch to the height to its own row and one or more below it.

When you stretch a row to take the space of two, you define that it actually takes the place of the following row's table data cell that would visually appear just below it. This is why you see only two table data cells coded in row three of the table below.

In order for you to even tell this is a table in the browser view, we need to add border information to the table shell and the table data cells: style="border:1px black solid", etc.

```
<table style="border:1px black solid">
<tr>
<td colspan="2" style="border: 1px gray solid;">I am table row 1 cell 1
stretching 2 columns wide.</td>
<td style="border: 1px gray solid;">I am table row 1 cell 2</td>
</tr>
<tr>
<td style="border: 1px gray solid;">I am table row 2 cell 1</td>
<td style="border: 1px gray solid;">I am table row 2 cell 2</td>
<td rowspan="2" style="border: 1px gray solid;">I am table row 2 cell 3
stretching 2 rows</td>
</tr>
<tr>
<td style="border: 1px gray solid;">I am table row 3 cell 1</td>
<td style="border: 1px gray solid;">I am table row 3 cell 2</td>
</tr>
</table>
```

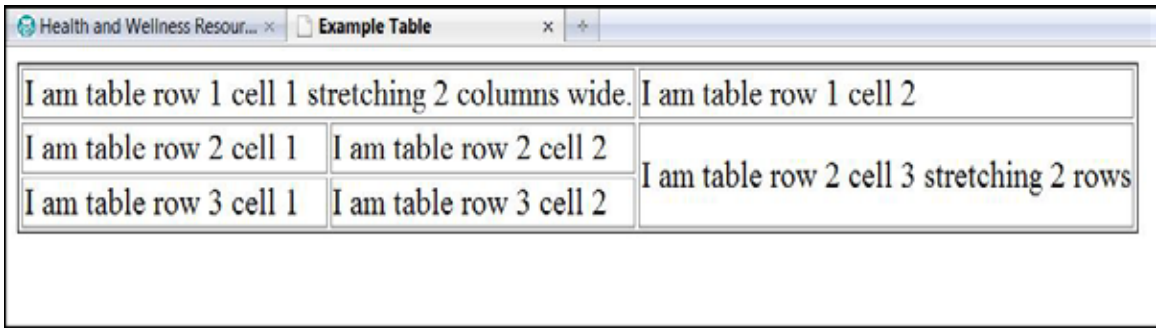


Table with spans

Table Spacing

Tables exist for you to place information inside of them, through putting content inside the table data cells. The default spacing for content in these cells defaults to 0, so your content can look pushed up against the borders of your cells.

The values for adjusting these spacings are auto, %, and length (in terms of pixels).

Padding

The padding is the space between the inside of a cell's border and the content. You can adjust it two ways:

- **Padding style**: Use a cascading styles rule: style="padding:2px;" This would affect all of the cell's borders, but you can choose padding-left, padding-top, etc.
- **Cellpadding**: You would use this as <td cellpadding="2">, which the browser will interpret as 2 pixels. Best not to use since it is on its way out to deprecated tag lala land.

Margins

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent. This is how you adjust this:

- **Margin style:** Use a cascading styles rule: style="margin:2px;" This would affect all of the cell's borders, but you can choose margin-left, margin-top, etc.

Spacing

This specifies the space between cells. You can adjust it two ways:

- **Spacing style:** Use a cascading styles rule: style="border-spacing:2px;" This would affect all of the cell's spacing, but you can choose border-spacing-left, etc.
- **Cellspacing**: You would use this as <td cellspacing=2>, which the browser will interpret as 2 pixels. This is the only version that truly works with collapse-borders to eliminate any border space in a table being used for something like a faux image map.

Table Styling

Tables default to no border and no background color. If you want your table to look tabular on your web page, you need to code it to show borders, colors, widths, etc. You need to do this with inline cascading styles, which we'll explain later in more detail. Here are some basics related to table styling.

Table Position

Table positioning defaults to the top left of the place where you code it, which means it will be set to the left of your page. In a sense, the *body* tag is the boss of the table as well as everything else inside the body, and you have to tell the body, somewhere before the table, how you want the table to be positioned. Therefore, if you want the table to be po-

sitioned in the center of the page, or to the right, you can't tell the table, but something outside of it.

This something is called a division element. While you can't code a table inside block-level items like a paragraph or an image, you can build one inside a <di>. This acts like a wrapper around the table element, so if you code the to do something, it will affect the whole table:

```
<div style="text-align:center;">
<table>Contents of table </table>
</div)
```

This affects the position of the whole table, not the text inside the table data cells.

Table Horizontal Content Alignment

Items inside of a table – text, images, links, etc., all default to left alignment. Once you have created a table, you have the ability to align the items inside the table. These items cannot be aligned inside the row, which does nothing except act as a placeholder for the table data cells. You can only align contents of a table data cell, regardless of whether they are text, images, lists, media, etc., using the text-align cascading style:

```
<td style="text-align:right">Right-aligned text</td>
```

Table Content Vertical Alignment

Items inside of a table – text, images, links, etc., all default to a middle vertical alignment. This means that if you have a table with several rows and columns, and the cell in one row has 6 lines of text inside it, the text in each of the other cells in that row will vertically align to the middle of the text in that 6-line cell. Ugh.

Once you have created a table, you have the ability to vertically align the items inside the table. These items also cannot be aligned inside the row - you can only align contents of a table data cell:

```
<td style="vertical-align:top">Top-aligned text</td>
```

The great news is that you can use the Table Groups discussed above to set these kinds of styling for a whole column of table data cells.

Borders

Tables don't really show up on the screen unless you have borders for them, and the default for HTML is no visible border.

You can set the border that outlines the table and its cells like a grid in the table tag:

```
<table style="width:50%; border-style:1px;">
<tr>
<td>Table cell 1</td>
<td>Table cell 2</td>
</tr>
</table>
```

You need to add style to your borders using CS rules. These include color, style, width, and collapse. None of the border properties will have any effect unless the border-style property is set

Border-style values:

- **none**: Defines no border
- **dotted**: Defines a dotted border
- **dashed**: Defines a dashed border
- **solid**: Defines a solid border
- **double**: Defines two borders. The width of the two borders are the same as the border-width value
- **groove**: Defines a 3D grooved border. The effect depends on the border-color value
- **ridge**: Defines a 3D ridged border. The effect depends on the border-color value
- **inset**: Defines a 3D inset border. The effect depends on the border-color value
- **outset**: Defines a 3D outset border. The effect depends on the border-color value

Border - Individual sides

This allows you to click and choose of individual sides of a table data cell border will look.

- border-top
- border-right
- border-bottom
- border-left

Border -Collapse

The border-collapse property sets whether the table borders are collapsed into a single border or detached as in standard HTML.

- **collapse**: Borders are collapsed into a single border when possible (border-spacing and empty-cells properties will be ignored)
- **separate**: Borders are detached (border-spacing and empty-cells properties will not be ignored). This is default
- **inherit**: Specifies that the value of the border-collapse property should be inherited from the parent element

Widths

The default width of a table, and therefore the cells that make up the table, is the width of the longest piece of content inside one of the cells. You need to set the width of tables and table cells yourself. Table rows extend the width of the table automatically, and cannot be adjusted.

This table is styled to extend 100% width of the browser window, and the two columns (table data cells) are respectively set to 30% wide and 70% wide. This is a flexible width table.

```
<table style="width:100%;">
<tr>
<td style="width:30%;">>Table cell 1</td>
<td style="width:70%;">Table cell 2</td>
</tr>
</table>
```

This table is styled to extend exactly 600 pixels wide, and the two columns (table data cells) are respectively set to 150 pixels wide and 460 pixels wide. This is a fixed width table.

```
<table style="width:600px;">
<tr>
<td style="width:150px;">Table cell 1</td>
<td style="width:450px;">Table cell 2</td>
</tr>
</table>
```

Backgrounds

You can change the background inside of a whole table, a row, and/or a cell, using cascading style rules. You have two options:

Background color:

```
<table style=" background-color:#999;">
<tr>
<td>Table cell 1</td>
<td style=" background-color:#ccc;">Table cell 2</td>
</tr>
</table>
```

Background Image: This requires you to link to an image by coding where its location is.

```
<table style="background-image:url(images/tablegrid.gif)">
<tr>
<td>Table cell 1</td>
<td>Table cell 2</td>
</tr>
</table>
```

Content in Tables

Now, a few words about the content you put into the tables. Here are a few key things to remember:

- All content – text, images, headers, links, lists, etc. that you want to display inside a table can only go **into** table data cells - <td></td>

- You do **not** need to use the tags to place your paragraph text in a table data cell: <td>I am a paragraph</td>
- If you do want to use other text styles inside of table data cells, like the headers and lists, you **do** need to use their respective tags, or else the built-in bold and heading sizes will not work. <td><h3>I am a Heading 3</h3></td>
- If you place images, links, or any other non-paragraph items in a table data cell, you **do** need their respective tags.

We'll be learning more about external cascading style sheets, which will alter some of this information, but for now, here we are!

We will also talk about nesting tables (tables inside of tables) in the next chapter. Woo-hoo!

QuickLook

Background Images Revealed!

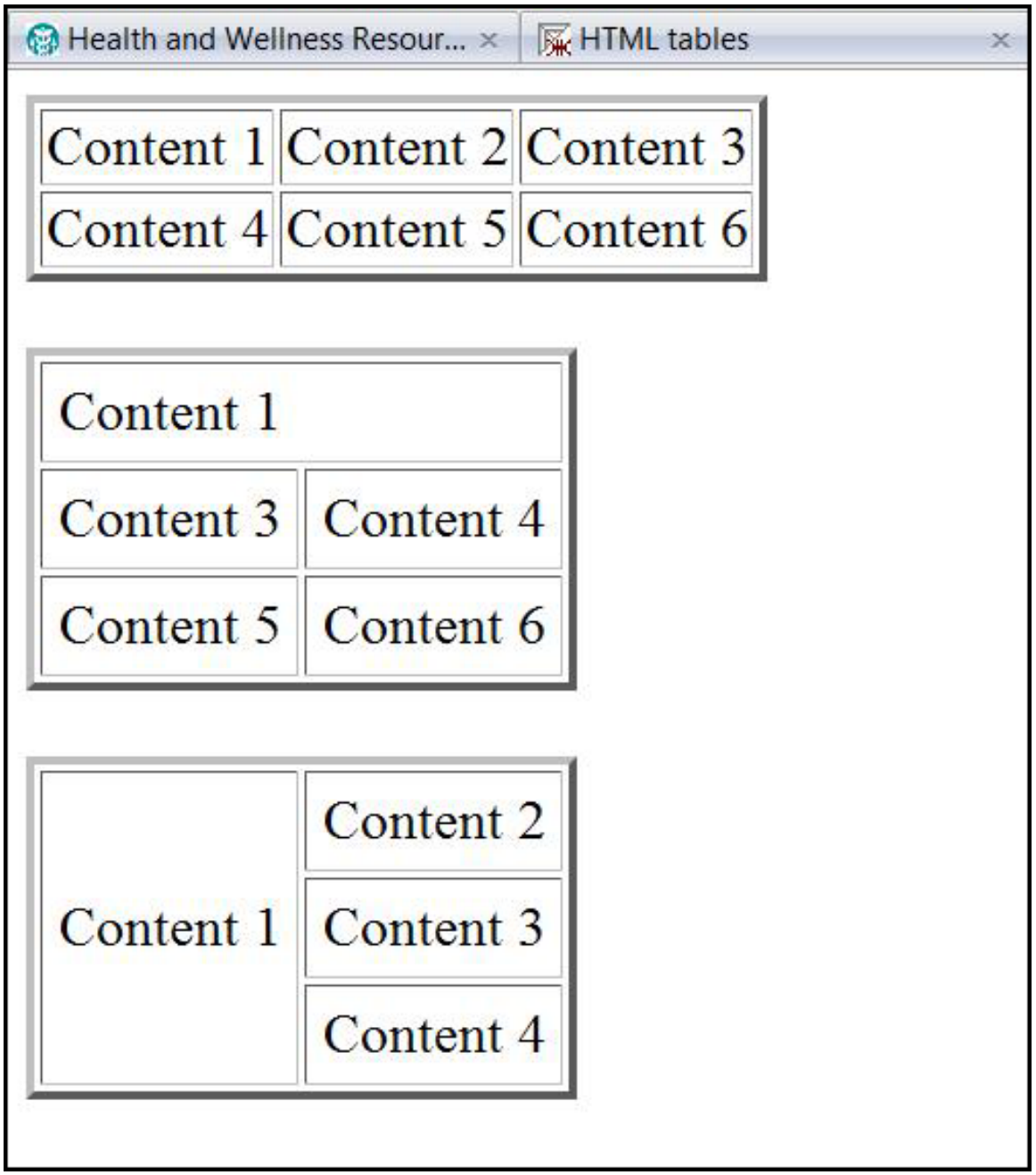
You code a table cell to have a background image, and the image doesn't show up. Why?

The background image in a table and td tag only shows up if there is other content inside the tags to push the cell open. If you don't have text inside the cell, it basically stays closed to its default height, because the cell doesn't know you want it to be wider or taller.

If you don't plan to put text in a table cell, you can:

- Add style="padding-top:____px;" or
- Add a few
 or   tags to push down a bit.

Try It Out


Health and Wellness Resour...
HTML tables
Content 1
Content 2
Content 3
Content 4
Content 5
Content 6
Content 1
Content 3
Content 4
Content 5
Content 6
Content 1
Content 2
Content 3
Content 4

14
Page Layout Code – Part 1

We've been learning about HTML elements, attributes, and values in web pages. We've even seen some really basic web pages, but they have been pretty straightforward up-and-down text that looks very little like anything you like looking at on the web. How do we get from here to there?

Small steps, grasshopper.

Seriously, we get to start working toward solid web page builds. One big step is in building web page layout structures, like we discussed eons ago in an earlier chapter. Now that we've covered tables, we'll learn how to use them to set up a straightforward web page layout.

A word about this first. HTML/XHTML has been moving towards as much cascading styling (CS) as possible. Web page layouts are an important part of this movement, and professional web authors focus on as much CS as possible. Despite this, a web author in today's work market is as likely to inherit existing and unstyled work, in content management systems and company sites that are too big and entrenched to redesign from scratch. They will inherit small, low-paying freelance gigs where they get to maintain and update existing websites. And in many of these cases, the site has been built using tables for the site's layout structure.

Our goal in this chapter is to learn a clean, efficient, and well-commented way to build web page layouts using tables, and how to nest other tables inside these page layout tables. The cool thing is that when we get to the transition from table layout to CS division layout, your attention to detail here will help you easily translate your table into the CS format.

Measurements

A key thing the web author needs to decide, when designing a website, is how wide it will be in relation to the open browser window. As you look at examples on the Internet, you will see all sorts of layouts, from flexible full-width to fixed 80% width (like many Wordpress blog layouts).

- Pixel measure results in fixed or static content

- Em measure results in proportional content that is relative to font-size
- Percent measure results in fluid content that shrinks and grows to "fit" display windows

The great thing is that, with tables (and later CS divs) you can mix and match:

- Flexible full-screen width and flexible columns
- Flexible full-screen width and a mix of fixed and flexible columns
- Fixed partial or full-screen width and flexible columns
- Fixed partial or full-screen width and a mix of fixed and flexible columns

Outline Page Structure First

Your wireframes and comps of your planned website should give you an clear idea of what layout you plan to translate into a table structure. You still can benefit from sketching out the table you plan to create and making notes to yourself about the widths, colors, borders, spacings, and other considerations you will build. You can also benefit from penciling in the <td> tags and where your colspans and row spans will be. A sketched table planning diagram helps you stay on track and mark the beginning and end of columns and rows in case your code gets you a bit turned upside down.

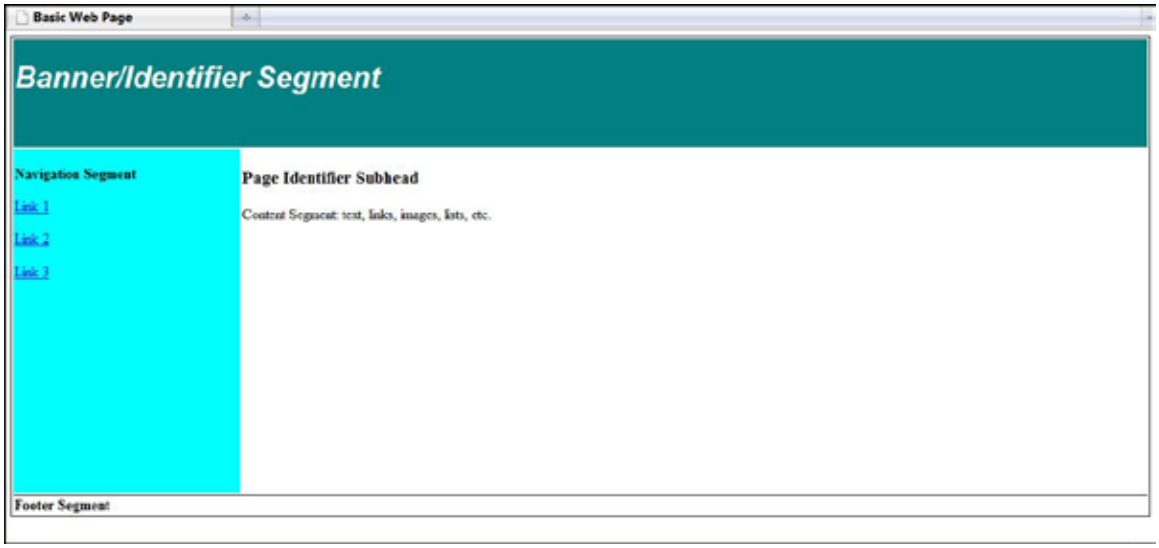
Table Layout

Let's look at a common layout for web pages, and translate this info a table format. First, to recap earlier discussion, key items visitors expect to see (in some way) in a web page include:

- Banner/identification segment
- Navigation segment
- Content (text, images, lists, links, etc.) segment
- Footer segment

A common layout for a web page that includes these is a left-side navigation page, where you see:

- A page-wide and 1-2 inch high banner segment
- A 2-3 inch left-hand navigation segment
- A several inch wide content segment taking up the middle and right of the browser window
- A page-wide and 1-2 inch high footer segment



How do you translate that into a table layout?

Keeping it simple, we'll look at the main table staples – the TABLE, TR, and TD tags. We will also consider column spans and row spans, if needed. For the proposed design, we really have a simple table which will have 3 rows and 2 columns.

Unstyled Left Navigation Layout Table

```
<body>
<table>
<tr>
<td colspan="2">Banner Segment</td>
</tr>
<tr>
<td>Navigation Segment</td>
<td>Content Segment</td>
</tr>
<tr>
<td colspan="2">Footer Segment</td>
</tr>
</table>
</body>
```

Now, let's start good coding practices by adding commenting to tell ourselves what is going on:

Unstyled and Commented Left Navigation Layout Table

```
<body>
<!– Left Navigation Layout Table–>
<table>
<!– Row 1 – Banner Segment–>
<tr>
<!– Column 1 – Spans 2 columns, no second TD needed  –>
<td colspan=”2”>Banner Segment</td>
</tr>
<!– Row 2 – Navigation and Content Segments–>
<tr>
<!– Column 1 – Navigation segment, let's make 220 pixels wide –>
<td>Navigation Segment</td>
<!– Column 2 – Content segment, let's leave at about 80% wide –>
<td>Content Segment</td>
</tr>
<!– Row 3 – Footer Segment–>
<tr>
<!– Column 1 – Spans 2 columns, no second TD needed  –>
<td colspan=”2”>Footer Segment</td>
</tr>
</table>
</body>
```

Next, we'll just leave the comments as is, but add basic table width styling. We are not adding a border, because we don't want grid lines in a web page layout. We are setting the vertical alignment of cell contents to *top* in the table data cells, and giving a flexible table width, and a semi-flexible widths to the 2 columns that make up the navigation and content cells.

Styled and Commented Left Navigation Layout Table

```
<body>
<!– Left Navigation Layout Table–>
<table style="width:100%;">
<!– Row 1 – Banner Segment–>
<tr>
<!– Column 1 – Spans 2 columns, no second TD needed  –>
<td colspan="2" style="vertical-align:top;">Banner Segment</td>
</tr>
<!– Row 2 – Navigation and Content Segments–>
<tr>
<!– Column 1 – Navigation segment, let's make 220 pixels wide –>
<td style="width:220px; vertical-align:top;">Navigation Segment</td>
<!– Column 2 – Content segment, let's leave at about 80% wide –>
<td style="width:80%; vertical-align:top;">Content Segment</td>
</tr>
<!– Row 3 – Footer Segment–>
<tr>
<!– Column 1 – Spans 2 columns, no second TD needed  –>
<td colspan="2" style="vertical-align:bottom;">Footer Segment</td>
</tr>
</table>
</body>
```

Now, you may be wondering why we have not set any **height** to any of the cells? If you code this in your text browser, you will see a flattened-looking screen-wide table with no borders. Huh?????

We do not add height to the cells because, for the most part, it is superfluous. Why? Because the height of your layout table will be determined by how much content you have in the longest table data cell. Heights are also not supported in HTML 4.01 Strict and XHTML 1.0.

Consider this: your banner and footer segments may only contain 1-inch high content, and your navigation segment might only look like it has three inches of text or image links. However, the content segment, if it has ten inches of height in text and images, will push the the page downward so it becomes longer.

Padding

This attribute specifies the amount of space between the border of the cell and its contents. If the value of this attribute is a pixel length, all four margins should be this distance from the contents. If the value of the attribute is a percentage length, the top and bottom margins should be equally separated from the content based on a percentage of the available vertical space, and the left and right margins should be equally separated from the content based on a percentage of the available horizontal space.

Fortunately, the CS padding gives you some control over a table cell's height. After all, what happens if you have a page with about navigation links about four inches high, but content of only three inches, and you'd like your footer to appear at least seven inches down? If you can't set the height, what can you do?

The good news is padding. Much of the time, folks think of padding as helping them push text or images away from the right or left edges of a table cell so they don't look squished against the neighboring cell. However, you can push a cell's padding up or down, when needed. Consider:

Styled Left Navigation Layout Table With Padding-Bottom

```
<body>
<table style="width:100%; border:1px black solid;">
<tr>
<td colspan="2" style="vertical-align:top; border:1px gray solid;">Banner
Segment</td>
</tr>
<tr>
<td style="width:220px; vertical-align:top; padding-bottom: 400px;
border:1px gray solid;">Navigation Segment</td>
<td style="width:80%; vertical-align:top; border:1px gray solid;">Content Seg-
ment</td>
</tr>
<tr>
<td colspan="2" style="vertical-align:bottom; border:1px gray solid;">Footer
Segment</td>
</tr>
</table>
</body>
```

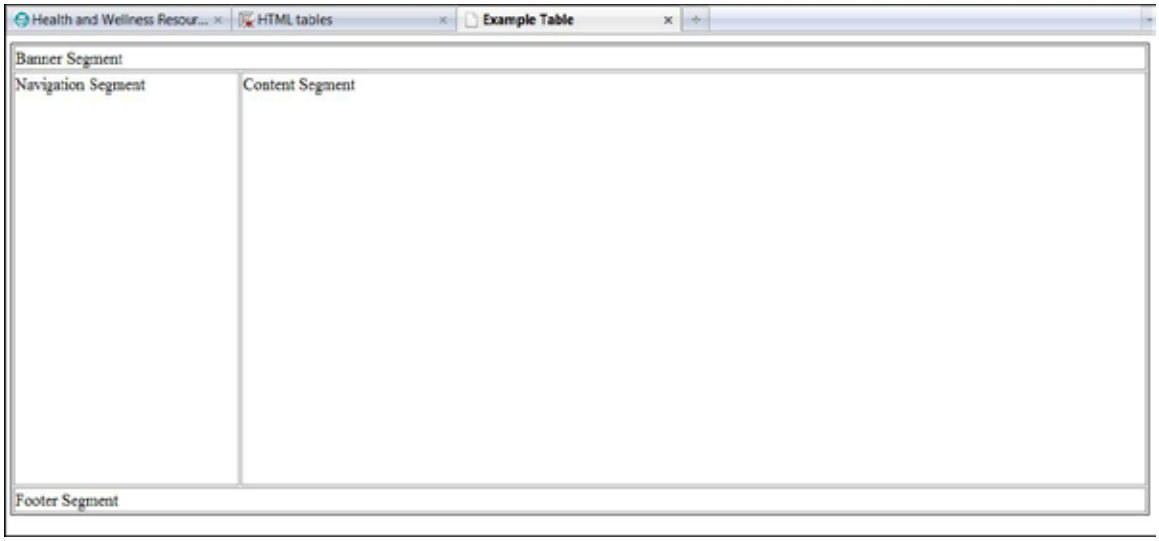


Table with padding

What will happen here is that the <td> making up the navigation segment has added 400px of padding at the bottom of the cell, which pushes it down from the bottom of the text that would make up the navigation links. This, in turn, pushes the height of the content segment cell down by default, so that both have greater height even though they may be mostly empty of content.

Margin

This attribute specifies how much space should be left between the left side of the table and the left-hand side of the leftmost column, the top of the table and the top side of the topmost row, and so on for the right and bottom of the table. The attribute also specifies the amount of space to leave **between** cells.

In table layout, you may find that using the cascading style margin property will allow you to push cells away from the border of the table.

Tables Inside of Tables – Yes, Nesting!

Accurate and efficient nesting of tables sometimes feels like a world unto itself. Like nesting of other web page elements, nesting table-related tags properly means that when using more than one pair of tags to affect code at the same time, you open and close these pairs in the right order as related to the other pairs.

In the case of tables, if you do not do this and do it well, your whole table for page structuring will look like it exploded all over your screen. The W3CValidator will also make you want to blow up.

A Table Inside a Table

Why would you want to nest a table inside a table? Well, we waited until this chapter on creating a web page layout with a table to help answer that. What happens if, in the content segment of your web page, you need to create a table of, say, books and their prices? You would use a table with two columns and however many rows are needed for your list of books, right?

But wait! You have created your whole web page structure using a table for the structure, like our left-side navigation structure above. And you need your book table inside of it. What now?

Let's look at the unstyled basic structure table code again, with a comment as to where the nested table needs to go:

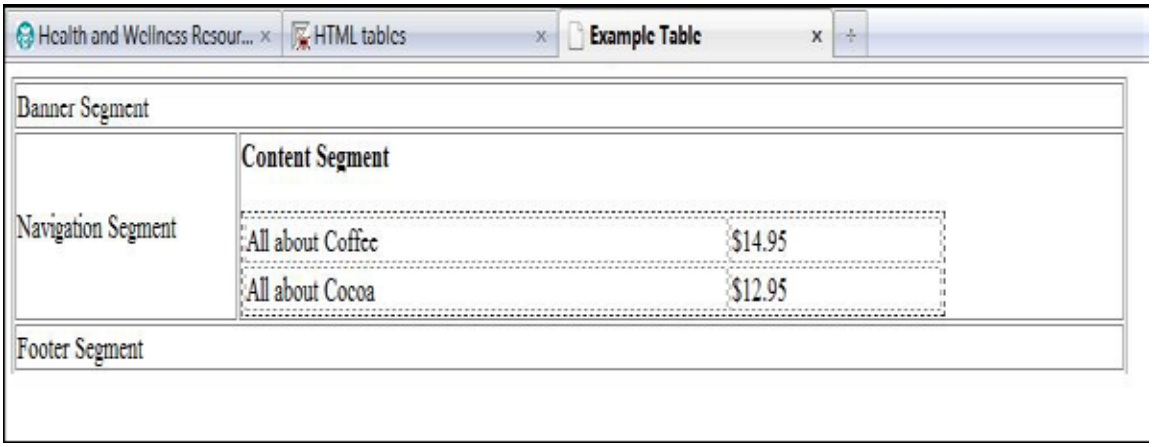
Unstyled Left Navigation Layout Table

```
<body>
<table>
<tr>
<td colspan="2">Banner Segment</td>
</tr>
<tr>
<td>Navigation Segment</td>
<!– This Content Segment cell needs a table of books and prices –>
<td>Content Segment</td>
</tr>
<tr>
<td colspan="2">Footer Segment</td>
</tr>
</table>
</body>
```

The simple key is to create a new table inside of that existing content segment <td>:

Styled Left Navigation Layout Table With Nested Table

```
<body>
<table style="width:80%; border:1px gray solid;">
<tr>
<td colspan="2" style="border:1px gray solid;">Banner Segment</td>
</tr>
<tr>
<td style="width:20%; border:1px gray solid;">Navigation Segment</td>
<!– This Content Segment cell needs a table of books and prices –>
<td style="border:1px gray solid;"><strong>Content Segment</strong> <br
/><br />
<table style="width:80%; border:1px black dashed;">
<tr>
<td style="border:1px gray dashed;">All about Coffee</td>
<td style="border:1px gray dashed;">$14.95</td>
</tr>
<tr>
<td style="border:1px gray dashed;">All about Cocoa</td>
<td style="border:1px gray dashed;">$12.95</td>
</tr>
</table>
</td>
</tr>
<tr>
<td colspan="2" style="border:1px gray solid;">Footer Segment</td>
</tr>
</table>
</body>
```



Styled Left Navigation Layout Table With Nested Table

See how the nested table fits right inside the <td>…</td> of the page layout table?

Your real issue will be keeping the <td>…</td> clearly in sight when you are building a long and robust table inside of it – such as when you create a table-structured contact form inside your web page layout table. Here is a tip – **use it**!!

```
<td style="border:1px gray solid;"><strong>Content Segment</strong> <br
/><br />
<!– @@@@@@ START Nested Table @@@@@ –>
<table style="width:80%; border:1px black dashed;">
<tr>
<td style="border:1px gray dashed;">All about Coffee</td>
<td style="border:1px gray dashed;">$14.95</td>
</tr>
<tr>
<td style="border:1px gray dashed;">All about Cocoa</td>
<td style="border:1px gray dashed;">$12.95</td>
</tr>
</table>
<!– @@@@@@ END Nested Table @@@@@@  –>
</td>
```

QuickLook

New web authors tend to get table nesting convoluted when they first start. For instance, the following simple table layout with a nested table inside the content segment is nested properly.

```
<table>
<tr>
<td colspan="2">Banner Segment</td>
</tr>
<tr>
<td>Navigation Segment</td>
<td>
<h3>Content Segment</h3>
<p>Here is a nested table.</p>
<table style="width:80%; border:1px black dashed;">
<tr>
<td style="border:1px gray dashed;">All about Coffee</td>
<td style="border:1px gray dashed;">$14.95</td>
</tr>
<tr>
<td style="border:1px gray dashed;">All about Cocoa</td>
<td style="border:1px gray dashed;">$12.95</td>
</tr>
</table>
</td>
</tr>
<tr>
<td colspan="2">Footer Segment</td>
</tr>
</table>
```

Unfortunately, it can be easy to forget that the nested table has to occur inside an opening and closing <td> shell of the page structure table, and new authors forget to close this shell, instead thinking that the end of the nested table is the end of the page table, for instance.

A Table-Structured Layout

```
<!– This page is produced by YourName–>
<!– http://yourdomain.com/ –>
<!– FTP: yourdomain.com/ Login:yourlogin –>
<!– To validate - http://validator.w3.org/#validate_by_input –>
<!–REQUIRED DOCTYPE INFO FIRST –>
<!doctype html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”><html xmlns=”http://
www.w3.org/1999/xhtml” xml:lang=”en” lang=”en”> <!– Opening HTML tag
must abut against end of doctype –>
<!–THEN HEAD SECTION –>
<head>
<!– Required personalized title  and meta tags –>
<title>Basic XHTML Table Template Page </title>
<meta http-equiv=”content-type” content=”text/html;charset=utf-8” />
<meta http-equiv=”content-style-type” content=”text/css”>
<meta name=”author” content=”YourName” />
<meta name=”copyright” content=”YourName and/or Company “ />
<meta name=”description” content=”Web design and development, Your-
Name.” />
<meta name=”keywords” content=”web design, xhtml, web page, pick some
relevant words “ />
</head>
<!–THEN BODY SECTION –>
<body>
<!–PAGE LAYOUT USING TABLE STRUCTURE BEGINS HERE–>
<table style=”width:100%; border:2px black solid;”>
<!– BEGIN ROW 1 WITH 1 WIDE COLUMN FOR BANNER SECTION –>
<tr>
<td colspan=”2” style=”border:2px green dashed;”><h1>Banner area to iden-
tify website</h1></td>
</tr>
<!– END ROW 1 WITH 1 WIDE COLUMN –>
<!– BEGIN ROW 2 WITH 2 COLUMNS FOR NAVIGATION AND CONTENT SEC-
TIONS–>
```

```
<tr>
<!– Column 1 –>
<td style=”width:20%; vertical-align:top; border:2px blue
dashed;”><p>Navigation links of website</p></td>
<!– Column 2 –>
<td style=”width:80%; vertical-align:top; border:2px purple
dashed;”><p>Content area of website- text, images, links, lists, etc.</p></
td>
</tr>
<!– END ROW 2 WITH 2 COLUMNS –>
<!– BEGIN ROW 3 WITH 1 WIDE COLUMN FOR FOOTER SECTION –>
<tr>
<td colspan=”2” style=”border:2px red dashed;”>
<h5>Footer of website - copyright, contact email, home link</h5></td>
</tr>
<!– END ROW 3 WITH 1 WIDE COLUMN –>
</table>
<!–PAGE LAYOUT USING TABLE STRUCTURE ENDS HERE–>
</body>
<!–THEN CLOSING HTML TAG –>
</html>
```

Table to lay out web page structure

Try it Out


Health and Wellness Resour...
HTML tables
Basic XHTML Table Templ...
Banner area to identify website
Horizontal navigation for website
Content area of website- text, images, links, lists, etc.
Right sidebar of website
Footer of website - copyright, contact email, home link

15
Styling Code, Part 1

So far, we've been looking at elements, values, and attributes, and sticking with rudimentary style information. Now we are going to look into the basics of Cascading Styles (CS). It's usually referred to as CSS, but technically CSS stands for Cascading Style Sheets, and if you are not linking external style sheets to your web pages, you aren't using CSS. Just the Cascading Styles part of the benefit.

CS is a style language used to describe the presentation semantics (the look and formatting) of a document written in a markup language – HTML, XHTML, and XML. It is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, and reduce complexity and repetition in the structural content (such as by allowing for table-less web design).

It is called 'Cascading' in reference to Cascading Style Sheets (CSS), where one change made in a style sheet file can cascade changes in the appearance of that style in all the web pages linked to it. Also, CSS specifies a priority scheme to determine which style rules apply if more than one rule matches against a particular element. In this so-called cascade, priorities or weights are calculated and assigned to rules, so that the results are predictable.

Cascading styling specifications are different than those used with HTML style attributes and values (most of which are now deprecated). These specifications are exacting for CS to work.

You can associate formatting instructions with a web page by defining CS rules in either an embedded, linked, or inline style sheet. Although there are many ways of defining styles, the main difference between them is where they are located. CS can be coded in three ways:

- **Inline style rules**: Cascading styling is coded inline right in the HTML tags of web pages.
- **Embedded head section style rules**: Cascading styling for items used repetitive-

ly in a web page can be styled in STYLE tags in a web page's head section.

- **Cascading style sheets**: Cascading styles are created in their own document, with an extension **.css**. All pages in a website can link to this one CSS file, and if one style change is made in that file, all the web pages in the site will immediately reflect that style change.

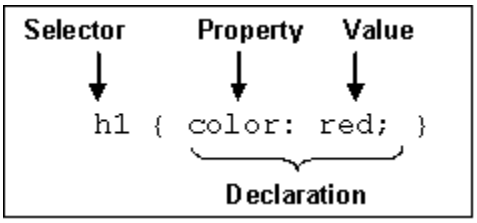
Linked style sheets are preferred for live websites. Embedded and inline style sheets are useful for prototyping but do not have the maintenance savings associated with linked style sheets.

Cascading Style Rules

Style rules are the method you use in CS to create a style to apply to your HTML code. The code style requires a tag/selector, attribute/value pair, and semi-colon concluder for each attribute/value pair used.

Each rule has two parts: a selector and a group of one or more declarations:

Inline: <selector style="property:value;">
Embedded, Linked Sheet: selector {property:value;}



Let's break it down:

- **Selector**: The tag which determines which HTML element will affected - <**p** style="font-size:2; color:#D00000;">
- **Declaration**: One or more attribute/value pairs that will give style to the selector/tag: <p style="**font-size:2; color:#D00000;"**>
- **Attribute**: The part of the declaration that is the attribute of the HTML tag: <p style="**font-size**:2; **color**:#D00000;">
- **Value**: The style choice made for the selector/tag and its attribute: <p style="font-size:**2**; color:**#d00000**;">
- **Closing clarification**: Every attribute/value pair needs a semi colon to 'finish' it. <p style="font-size:2**;** color:#d00000**;**">

Keep in mind that

- Inline styles can only be used inside of a style="" attribute for another tag, such as <p **style=""**> </p>.
- ALL rules need to have the attribute/value paired style shown after **style="** and must close with another **"** (quote marks)
- All measurements need a qualifier - px, %, etc.
- You cannot use text, image or link styles for table, table row, or table data cell tags.
- You cannot use table, table row, and table data cell styles for text, image or link styles

CS Measurements

- **em**: The height of the element's font.
- **ex**: The height of the letter 'x' in the element's font.
- **px**: pixel
- **mm**:millimeter
- **cm**: centimeter
- **pt** : point (1/72 inch)
- **pc**: pica (12 points = 1/6 inch)
- **in**: inch
- **%:** percentage

Note that:

- The *absolute* units mm, cm, pt, pc and in do not work well on screen and cause problems in many older browsers. It is safest to only use them for print media. Even in print media they may interact badly with users' preferences.
- The *relative* units em, ex and px do not specify a fixed length. Instead they scale relative to some other quantity. In the case of em and ex they scale relative to the font size of some element.
- The *flexible* unit percent lets the page interact gracefully with visitor's preferences.

CS Shorthand

CSS shorthand is a great way to speed up your development time and make your pages download faster by specifying several properties while using only one.

Grouping Selector Values: CS permits grouping values and giving them a shared declaration on one line with commas as separators. This only works if the values belong to the same selector. For instance, when styling font attributes, you can shorten the following:

From:

```
font-family: Verdana, Arial, sans-serif; font-size:110%; font-style: italic; font-variant: small-caps; font-weight:bold;
```

To:

```
font:Verdana, Arial, sans-serif 110% italic small-caps bold;
```

From:

```
background-color: #fff;
background-image: url(images/bg.gif);
background-repeat: repeat-y;
background-position: top left;
background-attachment: scroll;
```

To:

```
background: #fff url(images/bg.gif) repeat-y top left scroll;
```

Measurements

When using measurements for attributes like margin and padding, there is a Clockwise Rule for how the measurements may be shorthanded: top, right, bottom, left. For instance, margin: 1px 3px 2px 10px; is a shortcut for:

- **margin-top**:1px;
- **margin-right:**3px;
- **margin-bottom**:1px;
- **margin-left**:10px;

Inline Cascading Styles Quick Examples

These are not exhaustive, but give a good sampling. You will want to download a current CSS Cheatsheet for your own use: http://www.cheat-sheets.org/#CSS

- **Body**: <body style="background-image:url(images/filename.ext);"></body>
- **Text**: <h2 style="font-family:arial; color:#333333; margin:15px;" ></h2>
- **Text**:
- **Text**:
- **Text**:
- **Image**:
- **Image /text wrapping**: <img style="float:right;"> for a left image wrap, and float:left for a right image wrap.
- **Table**:<table style="align:left; border: 1px black solid; background-color:#ccc;" >
- **Table**: <table style=" border:2px #003 solid; border-collapse:collapse;">
- **Table**:<td style="width:50%; valign:top; background-image:url('path/to/img.jpg');"> </td>
- **Table**, tr, and td tags allow vertical alignment, background color, vertical alignment
- **Text** tags allow text alignment, color, and font variations
- **Font color**: style="color:black;"
- **Text alignment**: style="text-align:center;"
- **Vertical alignment** for tables: style="vertical-align:top;"
- **Border**: style="border:1px;"
- **Background Color**: style="background-color:green;"

Inline Cascading Styles

Inlining style is far more inflexible than the other methods. To use inline style, one must declare a single style sheet language for the entire document using the Content-Style-Type HTTP header extension. With inlined CS, an author must send text/css as the Content-Style-Type HTTP header or include the following tag in the head section:

```
<meta http-equiv="content-style-type" content="text/css">
```

Inlining style loses many of the advantages of style sheets by mixing content with presentation. As well, inlined styles implicitly apply to all media, since there is no mechanism for specifying the intended medium for an inlined style. Normally this method should be used sparingly, such as when a style is to be applied on all media to a single occurrence of

an element. If the style should be applied to a single element instance but only with certain media, use the id attribute instead of the style attribute.

However, for the purposes of learning cascading style rules, we will focus on using simple CS inline on web pages, and touch on a little CS use in a web page's head section. We will look at more complex CS in a later chapter when we build a cascading style sheet.

Style may be inlined using the style attribute, which may be applied to any body element (including body itself) except for param, and script. The attribute takes as its value any number of CS declarations, where each declaration is separated by a semicolon. An example follows:

```
<p style="color:#ccc;">I am gray paragraph text.</p>
```

Embedded Cascading Styles

Embedded head section style rules are another method you use in CS to create a styles to apply to your HTML code. The benefit comes from creating a style once that can be used for the same element/tag throughout the web page. They should be used when a single document has a unique style. If the same style sheet is used in multiple documents, then an external style sheet would be more appropriate.

Head section CS style rules:

- Are coded in the head section, not the body section of your web page.
- Can be styled once in a page so you can cut down code and be more consistent.
- Follow the strict CS style rules.
- Require you to reference the styles inline in the body section of your page.
- Use a different commenting style than regular HTML code: /* comment */

Head section style rules have the CS standard format which is organized a little differently than inline:

- **Code style**: selector, brackets around property/value pairs, semi-colon concluder for each property/value pairs.
  p {color:red; }
- **All** rules need to have the property/value paired style info inside brackets **{}**
- **Selector**: Determines which elements are affected - **p** {color:red;}
- **Declaration**: One or more property/value pairs - p **{color:red;}**

- **Attribute**: The part of the declaration that is the attribute of the HTML tag: p {**color**:red;}
- **Value**: The style choice made for the selector/tag and its attribute: p {color:**red**;}
- **All** property/value pairs need a semi colon to 'finish' it. p {color:red**;**}

CS Embedding in Head Section

In order to create cascading styles through the web page, you need to include the meta tag:

```
<head>
<meta http-equiv="content-style-type" content="text/css">
</head>
```

You also need to add a style opening and closing tag in the head section. Between these tags you would then code the style rules you would like to impact the entire page.

```
<head>
<style type="text/css">
/* This is the CSS style for commenting - use only inside <style></style> section. */
body {background-image:url(folder/imagename.ext);}
p {font-family:arial;}
td {vertical-align:top;}
</style>
</head>
```

CS Results in Body Section

```
<body>
<!– Your website's body will use the imagename.ext as a pattern –>
<p>This text will appear as Arial</p>
</body>
```

Head Section CS Quick Examples

Again, these are not exhaustive, but give a good sampling. You will want to download a current CSS Cheatsheet for your own use: http://www.cheat-sheets.org/#CSS. Here are examples of how they can be styled, with commenting to explain:

- body {background-image:url(folder/imagename.ext); color:#000; margin:0; padding:0; } /* Default white background color, black font, no margin or padding, background image */
- body {background-color:#fff; color: #000; margin:0; padding:0; } /* Default white background, black font, no margin or padding */
- h1, h2, h3, h4, h5, h6, body, p {font-family: Verdana, Arial, sans-serif; font-weight: normal;}
- h3 {color:#336699;} /* can use #369, Header 3 is a lighter blue */
- p {font-size:80%;} /* 80% of default html paragraph size */
- img {border: 1px solid #000;} /* 1px black border around all images */
- img {border-style: none;} /* hides border around all images, especially good when using image as a link too */
- table {border: 1px #ccc solid; vertical-align:top; padding:3px; margin:6px;} /* can use #ccc, instead of #cccccc */
- td {background-image:url(folder/imagename.ext); background-repeat:norepeat;} /* Non repeating background image. To repeat downward only, use repeat-y. To repeat to the right, use repeat-x. */

Pseudo Classes

Some selectors, like the link <a> tag, have several variations on how it can be styled, depending on whether it is active, hover, visited, etc. These can work regardless of browser. They are known as pseudo classes, for adding special effects to some selectors.

For instance, the linking tag <a> has four default appearances when a visitor sees a link on a web page:

- **Link**: The appearance of an unclicked link. Default color is blue.
- **Hover**: The appearance of a link when the mouse hovers over it. No default color change.
- **Focus**: The appearance of a link when the mouse is pressed down on it. No default color change.
- **Visited**: The appearance of a link that has been visited. Default color is purple.

However, all those happen to the same selector, <a>, which has no real separated attributes. Therefore, pseudo classes act as those attribute separations. Here are examples of how they can be styled, with commenting to explain:

- a:link {color: #990000; text-decoration:none;} /* can use #383, unvisited link is a maroon color, no underline. Note: link must come before hover, visited, and focus */
- a:hover {font-size:120%; color: #000000;} /* can use #000, mouseover link makes text bigger. Note: hover must come after link and before visited and focus */
- a:visited, a:focus {color: #666666;} /* can use #666, visited link is a medium gray. Note: visited and focus must come after link and hover. */

CS Classes and ID

If you want a property to work throughout a site, you can set it as a class or id, like .red {color:red;} turns any text red you use it as a class for.

If you want to affect only a specific selector, such as all text in a specific table cell, you would list the selector then class or id: td.red {color:red;} This will be covered in more detail in a later chapter about external style sheets.

Colors for Styling

We haven't discussed the use of color in styling, which bears a closer look. One thing every web author should get is a link to the web-safe HEX colors – just plug that into a search engine and you'll get plenty of examples.

Web-safe colors are colors used in designing web pages, and the methods for describing and specifying those colors. These colors were chosen specifically because they matched the palettes selected by the then leading browser applications. However, as devices have 16-bit to 24-bit color, web-safe colors are no longer required. Better results are obtained by providing an image with a larger range of colors and allowing the browser to quantize the color space if needed.

Authors of web pages have a variety of options available for specifying colors for elements of web documents. Colors may be specified as an RGB triplet in hexadecimal format (a hex triplet); they may also be specified according to their common English names in some cases. Often a color tool or other graphics software is used to generate color values. Professionally, HEX color expression seems to be common and using names like red, green, and blue is not.

HEX colors are hex triplets - six-digit, three-byte hexadecimal number used in HTML, CSS, SVG, and other computing applications, to represent colors. The bytes represent the red, green and blue components of the color. Hexadecimal color codes begin with a hash (#) . They are expected to render accurately regardless of the browser they are displayed on.

```
<p style="#000;">Black text here </p>
```

Color Styling Tips

- **Use HEX colors**: Always use the # in front of the color: #ffffff.
- **Background-color**: Background colors can be used for <table>, <tr>, <td>, <body>.
- **Border-color** : Can be only used for border attributes of selectors that allow them.
- **Color**: Text coloring only. There **is no** font-color style.
- **Use small letters** whenever you have them in the HEX color code: #cc0000
- When the HEX numbers have 3 pairs of repeating digits, you can also shorthand it: #ff3344 = #f34, #cccccc = #ccc

Color Palettes

Not everyone who produces websites has a strong sense of design, and that coverage is part of other books. However, you can find some color palette tools on the Internet to help you look at how colors complement and contract with each other. For instance, if you determine that your audience for recreational gear is mostly interested in outdoor and back-to-nature solutions, you might want to use a color palette that emphasizes earthy and natural colors. But, what looks good together?

Website links will change, but you can look up palette information on a search engine: Color Combination Palettes, Color Scheme Designer, Color Scheme Chooser, etc.

When you find one, you will likely see an interface that allows you to choose a base color, such as a dark green, and set filters to see palettes for a complementary, contrasting, monochromatic, triadic, and/or analogic color schemes.

What We Learned

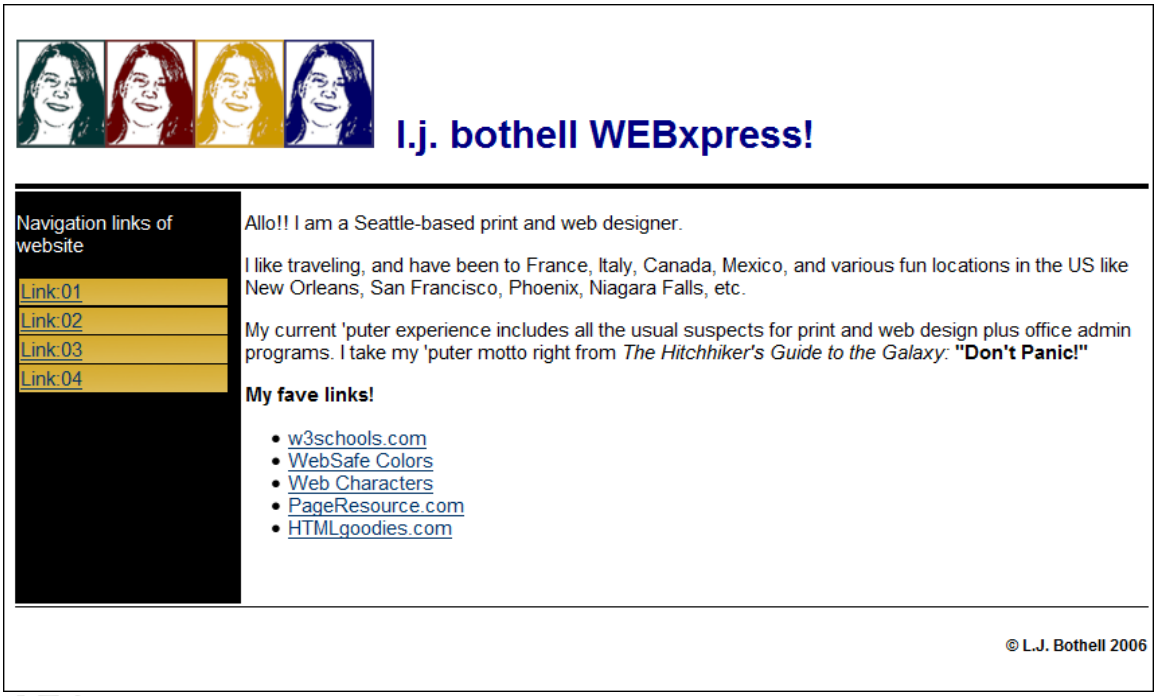
Example Cascading Styles

```
<!– This page is produced by Your Name–>
<!– http://yourdomain.freehostia.com/ –>
<!– FTP: yourdomain.freehostia.com/ Login:yourlogin –>
<!– To validate - http://validator.w3.org/#validate_by_input –>

<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN”
“http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”><html
xmlns=”http://www.w3.org/1999/xhtml” xml:lang=”en” lang=”en”> <!–
Opening HTML tag must abut against end of Doctype –>

<!–THEN HEAD SECTION –>
<head>

<!– Required personalized title  and meta tags - GET FROM HERE: http://
www.studiobast.com/sb_students/grabit/grabit01.php –>
<title>Basic XHTML Table Template Page </title>
<meta http-equiv=”Content-Type” content=”text/html;charset=utf-8” />
<meta name=”author” content=”YourName” />
<meta name=”copyright” content=”YourName and/or Company “ />
<meta name=”Description” content=”Web design and development, Your-
Name.” />
<meta name=”Keywords” content=”web design, xhtml, web page, pick some
relevant words “ />

<!– Using Embedded Cascading Styles –>
<style type=”text/css”>
h1 {font-family:arial, sans-serif; color:#000080;}
td {vertical-align:top;}
a:link {color: #003366;}
a:visited {color: #990000;}
a:active {color: #FF0000;}
</style>

</head>
```

```
<!–THEN BODY SECTION –>
<body style="font-family:arial,sans-serif; background-color:#fff;">

<!–PAGE LAYOUT USING TABLE STRUCTURE BEGINS HERE–>
<table style="width:100%;">

<!– BEGIN 2 COLSPAN BANNER SEGMENT–>
<tr>
<td colspan="2" style="border-bottom:5px #000 solid;">
<h1><img src="images/ljb01.gif" alt="lisa01" /><img src="images/
ljb02.gif" alt="lisa02" /><img src="images/ljb03.gif" alt="lisa03" /><img
src="images/ljb04.gif" alt="lisa04" />  l.j. bothell WEBxpress!
</h1>
</td>
</tr>
<!– END 2 COLSPAN BANNER SEGMENT –>

<!– BEGIN 2 COLUMN NAVIGATION AND CONTENT ROW–>
<tr>

<!– NAVIGATION SEGMENT –>
<td style="width:20%;background-color:#000;">
<p style="color:#fff;">Navigation links of website</p>
    <!– Table for navigation cell of bigger table–>
    <table style="width:95%;border:0;" cellspacing="2" cellpadding="2">
    <tr>
      <td style="background-image:url(images/mainnav.gif);width:209px;">
      <a href="filename.html" title="Link 1">Link:01</a>
      </td>
    </tr>
    <tr>
      <td style="background-image:url(images/mainnav.gif);width:209px;">
      <a href="filename.html" title="Link 2">Link:02</a>
      </td>
    </tr>
```

```
     <tr>
      <td style="background-image:url(images/mainnav.gif);width:209px;">
      <a href="filename.html" title="Link 3">Link:03</a>
      </td>
     </tr>

     <tr>
      <td style="background-image:url(images/mainnav.gif);width:209px;">
      <a href="filename.html" title="Link 4">Link:04</a>
      </td>
     </tr>
   </table>

</td>

<!– CONTENT SEGMENT –>
<td style="width:80%;">
   <p>Allo!! I am a Seattle-based print and web designer. </p>
    <p>I like traveling, and have been to France, Italy, Canada, Mexico, and
various fun locations in the US like New Orleans, San Francisco, Phoenix,
Niagara Falls, etc.</p>
     <p>My current 'puter experience includes all the usual suspects for print
and web design plus office admin programs. I take my 'puter motto right
from <em>The Hitchhiker's Guide to the Galaxy:</em> <strong>&amp;quot;Don't
Panic!&amp;quot;</strong></p>
     <p><strong>My fave links!</strong> </p>
     <ul>
      <li><a href="http://www.w3schools.com" target="_blank"
title="W3schools.com">w3schools.com</a></li>
      <li><a href="http://www.w3schools.com/html/html_colors.asp" tar-
get="_blank"  title="W3schools.com Colors">WebSafe Colors</a></li>
      <li><a href="http://www.intuitive.com/coolweb/entities.html" target="_
blank"  title="W3schools.com HTML Characters">Web Characters</a></li>
      <li><a href="http://pageresource.com/" target="_blank"
title="PageResource.com.com">PageResource.com</a></li>
      <li><a href="http://www.htmlgoodies.com/" target="_blank"
title="HTMLGoodies.com">HTMLgoodies.com</a></li>
     </ul>
```

```
        <br /><br />

    </td>

</tr>
<!– END 2 COLUMN NAVIGATION AND CONTENT ROW –>

<!– BEGIN 2 COLSPAN FOOTER SEGMENT –>
<tr>
<td colspan=”2” style=”border-top:2px #000 solid; text-align:right;”>
<h5>&copy; L.J. Bothell 2006</h5></td>
</tr>
<!– END 2 COLSPAN FOOTER SEGMENT –>

</table>
<!–PAGE LAYOUT USING TABLE STRUCTURE ENDS HERE–>

</body>

</html>
```



16 Template Page

We've looked at a lot of bits and pieces, so let's start putting them together. We'll do this by creating a simple table-structured template page which will use a combination of the things we've covered. This way you can see XHTML and CS in action. In this case, we'll look at template code for the Adopt-A-Furry-Friends concept mentioned earlier.

When you start building your web project, a template page is a seriously great first page. While it's tempting to jump in and start coding your index.html home page, starting with a template page is more effective. This allows you to work through your comp layout and coding kinds in a "safe" file, without the distractions of the content you will really need to insert for the website.

Use a basic template page to structure your site, prelink your navigation, set up your working footer, add head section styles and/or attach CSS links, add head section scripting, and set placeholders for your real text. Use it also to code all your basic text and link elements, so you can see in one place how all your headings, paragraph, list, and other text will look. This way you can make changes that affect this one file, get it set up the way you really want it, then use it as a template for your real website pages table data cell <td> borders so you can see how this looks.

Navigation Pre-linking

As you'll see, we will set up some links for the navigation to be used on the web pages in this web project. These links would be based on the pages you anticipate building for the website project, based on the graphic site map you would have created in your project planning stage.

The reason you will do this in a template page is so that when you replicate the template for actual page building, you will have the links already in place. You can pre-link even though the pages to be linked to have not yet been created. You can do this for your main navigation links to your website pages, your footer navigation, your logo, and your Back to Top links.

Opening Comments

A useful thing to do before you start coding is to build in some HTML comments to yourself at the very top of your page:

- The URL where you will see the live page.
- Your FTP information (minus password).
- The URL of the W3C Validator.
- Comments about the website's purpose, audience, and core considerations.
- Style and design comments, like color palette choices

```
<!– This page is produced by YourName–>
<!– http://yourdomain.com/ –>
<!– FTP: yourdomain.com/ Login:yourlogin –>
<!– To validate - http://validator.w3.org/#validate_by_input –>
<!– Local animal adoption group, audience includes people who want to
adopt housepets. Use vibrant colors that pop but have good contrast, –>
<!– Style/Design:Want rollover nav, earthy/natural colors with some pop
but that coordinate with images rather than overwhelm images, need good
contrast for readable areas, potential base color: #999966, or #339933,
#006600 –>
```

HTML Structure

Next, set up the HTML sections: HTML, head, and body code. We'll also comment the web page segments all the way through.

```
<!–REQUIRED DOCTYPE INFO FIRST –>
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://
www.w3.org/1999/xhtml" xml:lang="en" lang="en"> <!– Opening HTML tag
must abut against end of doctype –>
<!–THEN HEAD SECTION –>
<head>
<!– Required title and meta tags –>
<title>Adopt-A-Furry-Friend: Template Page </title>
<meta http-equiv="content-type" content="text/html;charset=utf-8" />
<meta http-equiv="content-style-type" content="text/css">
<meta name="author" content="YourName" />
<meta name="copyright" content="YourName " />
```

```
<meta name="description" content="Addopt-A-Furry-Friend: Adopt a dog, cat,
rabbit, or other furry friend for your family and home!." />
<meta name="keywords" content="YourName, adopt, veterinary, cats, dogs,
ferrets, rats, mouse, mice, rabbits, guinea pigs  " />
</head>
<!–THEN BODY SECTION –>
<body>
</body>
</html>
```

Page Table Structure

Next, let's focus on the body section, and add a table for page layout. We'll do a banner segment, a top horizontal navigation segment, a content segment with main content and a right-side sidebar, and a footer segment.

```
<body>
<!–START PAGE LAYOUT TABLE –>
<table>
<!– 1 WIDE COLUMN BANNER ID ROW –>
<tr>
<td colspan="2"><h1>Banner Segment</h1></td>
</tr>
<!– 1 WIDE COLUMN NAVIGATION ROW –>
<tr>
<td colspan="2">Navigation segment</td>
</tr>
<!– 2 COLUMN CONTENT AND SIDEBAR ROW –>
<tr>
<td>Content Segment</td>
<td>Sidebar Segment</td>
</tr>
<!– 1 WIDE COLUMN FOOTER ROW –>
<tr>
<td colspan="2">Footer Segment</td>
</tr>
</table>
<!–END PAGE LAYOUT TABLE –>
</body>
```

Page Table Styling

Next, let's focus on styling the layout table.

```
<body>
<!–START PAGE LAYOUT TABLE –>
<table style="width:850px; border:1px black solid; text-align:left; background-color:#fff;"><!–Makes the contents of the whole web page structural table left-aligned, and sets a white table background color –>
<!– 1 WIDE COLUMN BANNER ID ROW –>
<tr>
<td colspan="2" style="border-bottom: 3px #8c5803 solid; background-color:#274908;"><h1 style="color:#fff;">Banner Segment – Adopt-A-Furry-Friend!</h1></td><!—Adds a bottom border to the table cell, and gives the cell a background color. Also styles the H1 tag to be readable on a dark background. –>
</tr>
<!– 1 WIDE COLUMN NAVIGATION ROW –>
<tr>
<td colspan="2">Navigation segment</td>
</tr>
<!– 2 COLUMN CONTENT AND SIDEBAR ROW –>
<tr>
<td>Content Segment</td>
<td>Sidebar Segment</td>
</tr>
<!– 1 WIDE COLUMN FOOTER ROW –>
<tr>
<td colspan="2" style="background-color:#ffffcc">Footer Segment</td> <!–Sets a footer segment background color –>
</tr>
</table>
<!–END PAGE LAYOUT TABLE –>
</body>
```

Banner/ID Segment

Since we already styled the first wide column for the web page banner segment, let's move to the second wide column – the horizontal navigation. We will add some pre-linking here, based on a reasonable guess of what likely pages will be in the website and simple file naming conventions. Ideally, you would have this navigation information from your project planning Site Map.

```
<!– THIS IS 1 WIDE COLUMN NAVIGATION ROW –>
<tr>
<td colspan="2">
<p>
<a href="about.html" title="About Us">About Us</a> |
<a href="dogs.html" title="Dogs">Run with Dogs!</a>|
<a href="cats.html" title="Cats">Play with Cats!</a> |
<a href="other.html" title="Other">Other Furry Friends</a> |
<a href="faqs.html" title="FAQs">FAQs</a> |
<a href="contact.html" title="Contact Us">Contact Us</a> |
<a href="http://www.google.com" title="Home">Home</a></p>
</td>
</tr>
```

Content Segment – Main Area

Next, let's work on the main content segment of the web page. This is where you will build your main content for visitors to read – text, images, links, lists, media, etc. We are going to use this section, for this template example, to list various tags the website will likely use for text, images, and links. We will also build a small nested table. The idea is that when we add styles in the head section later, this section will show how the various headings and other page elements will look with that styling.

```
<!– THIS IS 2 COLUMN CONTENT AND SIDEBAR ROW –>
<tr>
<!– THIS IS MAIN CONTENT COLUMN –>
<td>
<h1>Header 1</h1>
<h2 style="font-family:Times New Roman, times, serif;">Header 2</h2> <!–
We will be styling the remaining fonts as sans serif later. –>
<h3>Header 3</h3>
<h4>Header 4</h4>
<h5>Header 5</h5>
<h6>Header 6</h6>
<p>Paragraph example</p>
<ul>
<li>Unordered list item</li>
<li>Unordered list item</li>
<li>Unordered list item</li>
</ul>
<ol>
<li>Ordered/Numbered list item</li>
<li>Ordered/Numbered list item</li>
<li>Ordered/Numbered list item</li>
</ol>
<dl>
<dt>Definition list item</dt>
<dt>Definition list item</dt>
<dt>Definition list item</dt>
</dl>
<hr /> <!– Example horizontal rule –>
<!– Nested table – must begin and end INSIDE the content table cells <td></
td> tags. –>
<table style="border:1px black solid">
<tr>
<td> Line item here </td>
</tr>
</table>
<p><img src="" alt="imagename" title="imagename" /></p> <!– Example
image –>
```

```
<p><a href="http://www.aaff.com" title="AAFF Home"
target="blank">Home</a> </p><!– Example link –>
</td>
```

Content Segment – Sidebar Area

The sidebar of the web page is a column in the same row as the main content column. At this time, let's just set up an example sidebar with a subhead, paragraph, and a caption tag.

```
<!– THIS IS SIDEBAR CONTENT COLUMN –>
<td style="border-left: 3px #8c5803 solid; width:20%;">
<h3>Sidebar Subtitle</h3>
<p>Sidebar Text</p>
<br />
<h6>Sidebar Caption</h6>
</td>
</tr>
```

Footer Segment

The footer segment is where you would add your copyright information, plus links to utility pages, like disclaimer, links sitemap, email link, etc.

```
<!– THIS IS 1 WIDE COLUMN FOOTER ROW –>
<tr>
<td colspan="2" style="background-color:#ffffcc">
<h5>&copy; 2010 YourName|
<a href="disclaimer.html" title="Disclaimer Page">Disclaimer</a> |
<a href="index.html" title="Home Page">Home</a> |
<a href="mailto:youremail@domain.com" title="Email Us">Email Us</a>
</h5>
</td>
</tr>
```

Head Section Cascading Styles

Finally, let's take another look at the Head Section code of the page, so that we can add some cascading styles. These added styles will allow all of the elements in this page that the styles refer to to automatically show that styling.

```
<!–THEN HEAD SECTION –>
<head>
<!– Required title and meta tags –>
<style type="text/css">
/* This is a Cascading Style comment here. Do NOT use HTML commenting
inside the style tags, or the styles will not work. */
td {vertical-align:top; padding:10px;}  /* Affects all table data cells <td> in
the page */
body /* Internet Explorer */ {font-family:verdana,arial,helvetica,sans-serif;
color:#000; text-align:center; background-color:#8C530E; } /*centers the
entire page in IE only and sets a golden brown body background color */
body /* Firefox */ {font-family:verdana,arial,helvetica,sans-serif; color:#000;
text-align:-moz-center; background-color:#8C530E; } /*centers the entire
page in IE only and sets a golden brown body background color */
h1 {color:#cc9;}  /* Affects all heading 1 <h1> in the page */
h2, h4 {color:#274908; font-weight:normal;}  /* Affects all heading 2 and
heading 4 <h2>, <h4> use in the page */
h3, h5 {color:#5D4425;} /* Affects all heading 3 and heading 5 <h3>, <h5>
use in the page */
a:link {color:#8C530E; font-size:90%; font-weight:bold;} /*golden brown color
*/
a:hover {color:#5D4425; font-size:90%; font-weight:bold;} /*medium brown
color */
a:visited, a:focus {color:#274908; font-size:90%; font-weight:bold;} /*dark
green color. Note: Focus does not work in IE */
</style>
</head>
```

What We Learned

Let's look at all the pieces put together:

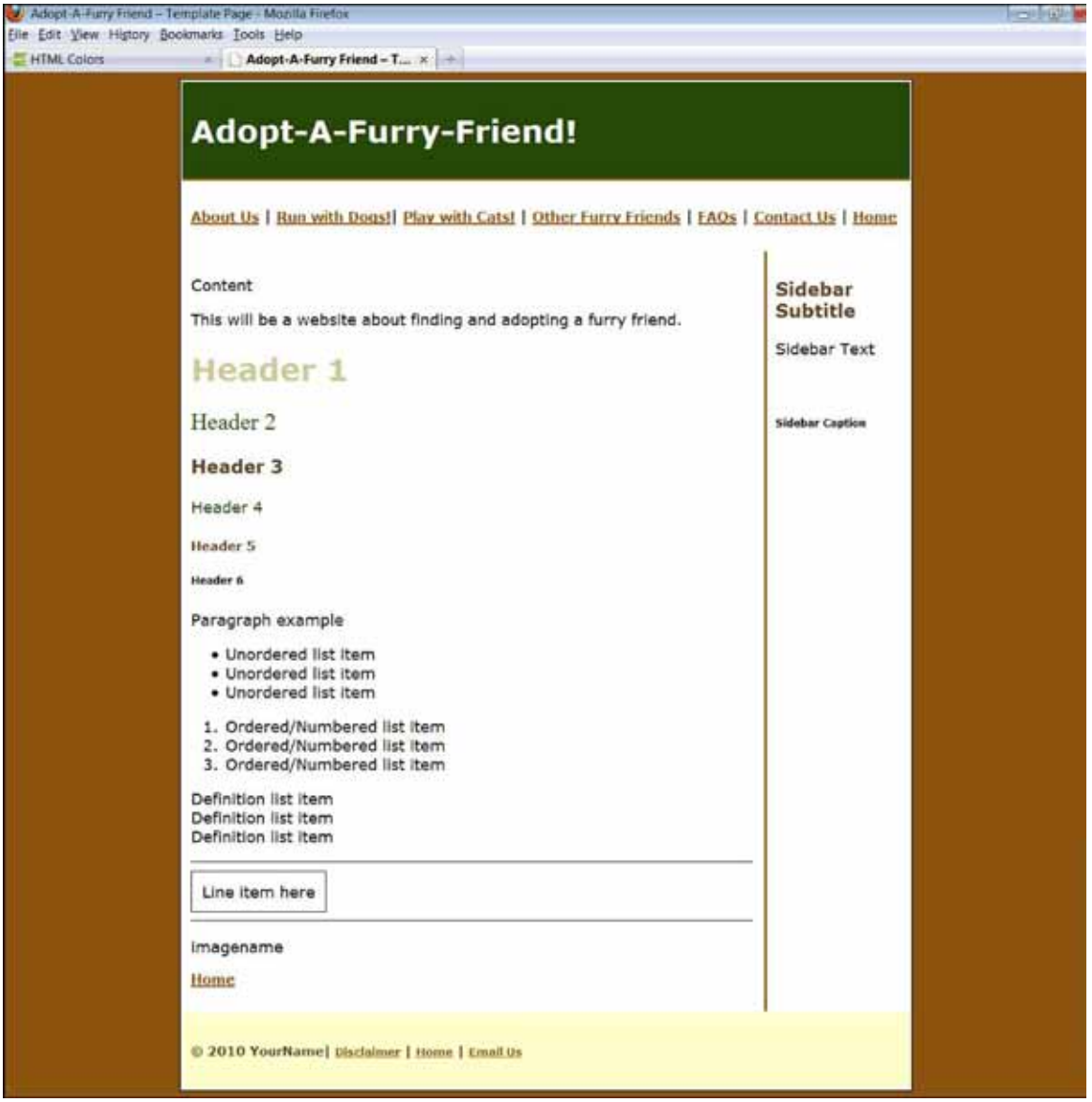
```
<!– This page is produced by YourName–>
<!– http://yourdomain.com/ –>
<!– FTP: yourdomain.com/ Login:yourlogin –>
<!– To validate - http://validator.w3.org/#validate_by_input –>
<!– Local animal adoption group, audience includes people who want to
adopt housepets. Use vibrant colors that pop but have good contrast, –>
<!– Style/Design:Want rollover nav, earthy/natural colors with some pop
but that coordinate with images rather than overwhelm images, need good
contrast for readable areas, potential base color: #999966, or #339933,
#006600  –>
<!–REQUIRED DOCTYPE INFO FIRST  –>
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://
www.w3.org/1999/xhtml" xml:lang="en" lang="en"> <!– Opening HTML tag
must abut against end of doctype –>
<!–THEN HEAD SECTION –>
<head>
<!– Required title and meta tags –>
<title>Adopt-A-Furry Friend – Template Page </title>
<meta http-equiv="content-type" content="text/html;charset=utf-8" />
<meta http-equiv="content-style-type" content="text/css">
<meta name="author" content="YourName" />
<meta name="copyright" content="YourName " />
<meta name="description" content="Addopt-A-Furry-Friend – Adopt a dog,
cat, rabbit, or other furry friend for your family and home!." />
<meta name="keywords" content="YourName, adopt, veterinary, cats, dogs,
ferrets, rats, mouse, mice, rabbits, guinea pigs  " />
<style type="text/css">
/* This is a Cascading Style comment here. Do NOT use HTML commenting
inside the style tags, or the styles will not work. */
td {vertical-align:top; padding:10px;}  /* Affects all table data cells <td> in
the page */
```

```
body /* Internet Explorer */ {font-family:verdana,arial,helvetica,sans-serif;
color:#000; text-align:center; background-color:#8C530E; } /*centers the
entire page in IE only and sets a golden brown body background color */
body /* Firefox */ {font-family:verdana,arial,helvetica,sans-serif; color:#000;
text-align:-moz-center; background-color:#8C530E; } /*centers the entire
page in IE only and sets a golden brown body background color */
h1 {color:#cc9;}  /* Affects all heading 1 <h1> in the page */
h2, h4 {color:#274908; font-weight:normal;}  /* Affects all heading 2 and
heading 4 <h2>, <h4> use in the page */
h3, h5 {color:#5D4425;} /* Affects all heading 3 and heading 5 <h3>, <h5>
use in the page */
a:link {color:#8C530E; font-size:90%; font-weight:bold;} /*golden brown color
*/
a:hover {color:#5D4425; font-size:90%; font-weight:bold;} /*medium brown
color */
a:visited, a:focus {color:#274908; font-size:90%; font-weight:bold;} /*dark
green color. Note: Focus does not work in IE */
</style>
</head>
<!–THEN BODY SECTION –>
<body>
<!–THEN PAGE LAYOUT STRUCTURE TABLE –>
<table style=”width:850px; border:1px black solid; text-align:left; back-
ground-color:#fff;”><!–Makes the contents of the whole web page structural
table left-aligned, and sets a white table background color –>
<!– THIS IS 1 WIDE COLUMN BANNER ID ROW –>
<tr>
<td colspan=”2” style=”border-bottom: 3px #8c5803 solid; background-
color:#274908;”><h1 style=”color:#fff;”>Adopt-A-Furry-Friend!</h1></td>
</tr>
<!– THIS IS 1 WIDE COLUMN NAVIGATION ROW –>
<tr>
<td colspan=”2”>
<p>
<a href=”about.html” title=”About Us”>About Us</a> |
<a href=”dogs.html” title=”Dogs”>Run with Dogs!</a>|
<a href=”cats.html” title=”Cats”>Play with Cats!</a> |
<a href=”other.html” title=”Other”>Other Furry Friends</a> |
```

```
<a href="faqs.html" title="FAQs">FAQs</a> |
<a href="contact.html" title="Contact Us">Contact Us</a> |
<a href="http://www.google.com" title="Home">Home</a></p>
</td>
</tr>
<!– THIS IS 2 COLUMN CONTENT AND SIDEBAR ROW –>
<tr>
<!– THIS IS MAIN CONTENT COLUMN –>
<td><p>Content</p>
<p>This will be a website about finding and adopting a furry friend. </p>
<h1>Header 1</h1>
<h2 style="font-family:Times New Roman, times, serif;">Header 2</h2>
<h3>Header 3</h3>
<h4>Header 4</h4>
<h5>Header 5</h5>
<h6>Header 6</h6>
<p>Paragraph example</p>
<!– NO paragraph or text tags inside of a unordered, ordered, or definition
list, and no text tags surrounding the lists –>
<ul>
<li>Unordered list item</li>
<li>Unordered list item</li>
<li>Unordered list item</li>
</ul>
<ol>
<li>Ordered/Numbered list item</li>
<li>Ordered/Numbered list item</li>
<li>Ordered/Numbered list item</li>
</ol>
<dl>
<dt>Definition list item</dt>
<dt>Definition list item</dt>
<dt>Definition list item</dt>
</dl>
<hr />
<table style="border:1px black solid">
<tr><td> Line item here </td>
</tr>
```

```
</table>
<hr />
<p><img src="" alt="imagename" title="imagename" /></p>
<p><a href="http://www.aaff.com" title="AAFF Home"
target="blank">Home</a> </p>
</td>
<!– THIS IS SIDEBAR CONTENT COLUMN –>
<td style="border-left: 3px #8c5803 solid; width:20%;">
<h3>Sidebar Subtitle</h3>
<p>Sidebar Text</p>
<br />
<h6>Sidebar Caption</h6>
</td>
</tr>
<!– THIS IS 1 WIDE COLUMN FOOTER ROW –>
<tr>
<td colspan="2" style="background-color:#ffffcc">
<h5>&copy; 2010 YourName|
<a href="disclaimer.html" title="Disclaimer Page">Disclaimer</a> |
<a href="index.html" title="Home Page">Home</a> |
<a href="mailto:youremail@domain.com" title="Email Us">Email Us</a>
</h5>
</td>
</tr>
</table>
</body>
</html>
```

The template example is on the next page:



Template Example

17 HTML Forms

HTML forms are used to pass data to a server, or to collect data to send in an email. You can use them, for instance, to get contact information from your visitors, and to elicit comments and opinions.

A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. It can also contain select lists, textarea, fieldset, legend, and label elements. Users generally complete a form by modifying its controls (entering text, selecting menu items, etc.), before submitting the form to an agent for processing (e.g., a Web server, a mail server, etc.) Some form of scripts (server-side, client-side, or both) must be used to process the visitor's input once it is submitted.

Users interact with forms through named controls. A control's *control name* is given by its name attribute. The scope of the name attribute for a control within a form element is the form element. These elements are either block or inline elements, but are collected here as their use is more restricted than other inline or block elements.

New HTML5 form attributes include autofocus, autocomplete, list, required, multiple, pattern, min and max, step, placeholder, numbers, times, telephone and color. At this time most are not supported at all or only by one or two browsers and versions, and none of the new form selectors are supported. I will mention useful HTML5 things in the context of the rest of the forms as appropriate.

Form Methods

In HTML, one can specify two different submission methods for a form using the method attribute. The two methods are post and get.

- Post: Form data is to appear within a message body, and/or for anything like storing or updating data, or ordering a product, or sending email.
- Get: Form data is to be encoded (by a browser) into a URL, and for getting (retrieving) data.

Form Handlers

In order to get a form working, it is not enough to just design the form itself. You also

have to create or find an HTML form handler, which is a program that runs on the web server. The form handler takes the information entered in the HTML form by the user and does something with it.

What it does depends on the nature of the HTML form. For a simple contact form, the form handler might do nothing except send the information entered to an email address. For a more sophisticated application, the form handler might be a highly complex program with ties in with databases and performs a variety of functions.

The form handler program is not part of the HTML form, but is a separate program which must be installed on your web host server. The form tag specifies the form handler URL in the action attribute:

```
<form name="MyForm" action="URL">
```

You access a form handler, you can:

- Find, modify, and use an existing form handler script.
- Have a developer create one for you.
- Use a remote form handling service.

Form Framework

The framework of a form is the structure. The tags used wrap around items, and so have an opening and closing tag.

Form

Creates a form. The form element specifies and operates the overall action of a form area. All contents of a form go between the opening and closing tag.
<form>…</form>

Action

A required attribute for the form to work. The action needs a URL to a form handler.
<form **action**="url">…</form>

Fieldset

A container for adding structure to forms. For example, a series of related controls can be grouped within a fieldset, which can then have a legend added in order to identify their function.
<fieldset>…</fieldset>

Legend

Describes a group of items in a fieldset as a caption.
<legend>…</legend>

Label

Creates a label for a form input (e.g. radio button). Clicking on the label fires a click on the matching input.
<label for="id">…</label>

Name

The name attribute specifies the name of a form or a form element, and provides a way to reference the element in a script.

```
<form>
Type of beverage: <input type="text" name="beverage type" /><br />
</form>
```

Value

Specifies the value of an input element, and is required for radio buttons and checkboxes to give the value of the control when it is checked.

Form Input Elements

The most important form element is the input element, which is used to select user information. An input element can vary in many ways, depending on the type attribute. An input element can be of type text field, checkbox, password, radio button, submit button, and more.

The most used input attributes are described below.

Text Fields

Defines a one-line input field that a user can enter text into, and the default width of a text field is 20 characters.

```
<form>
First name: <input type="text" name="firstname" /><br />
Last name: <input type="text" name="lastname" />
</form>
```

Text Area

Defines a multi-line input area a visitor can enter text into. The height is defined by the number of rows, and the width but the number of cols. The space between the opening and closing textarea tags can be left blank, which will create an empty text area for the visitor to type into. If you do add text, it will appear in the text box and visitors can replace it with their own text.

```
<form>
<textarea rows="6" cols="50"> Enter text here.</textarea>
</form>
```

Password Field

Defines a password field. The characters in a password field are masked (shown as asterisks or circles).

```
<form>
Password: <input type="password" name="pwd" />
</form>
```

Radio Buttons

Defines a radio button. Radio buttons let a user select only one of a limited number of choices:

```
<form>
<input type="radio" name="sex" value="male" /> Male<br />
<input type="radio" name="sex" value="female" /> Female
</form>
```

Checkboxes

Defines a checkbox. Checkboxes let a user select one or more options of a limited number of choices.

```
<form>
<input type="checkbox" name="beverage" value="Mocha" /> I like
mochas.<br />
<input type="checkbox" name="beverage" value="Latte" /> I like lattes.
<input type="checkbox" name="beverage" value="Cocoa" /> I like cocoa.
</form>
```

Dropdown List

The dropdown list uses <select>…</select> to allow users to choose from a list provided by <option>…</option>. The list can be full choice as below, or have a preselected option by adding *selected=”selected”* after one of the option values.

```
<form>
<select name=”beverages”>
<option value=”mocha” selected=”selected” >Mocha</option>
<option value=”latte”>Latte</option>
<option value=”cocoa”>Cocoa</option>
</select>
</form>
```

Email

HTML5 only, partly supported. The good news is that all browsers that don’t support it see input type=”email” the same way they see input type=”text”, so there is no error in using this. However, on the iPhone, when the email attribute is used, some devices will respond by offering an alternate keyboard on the screen which is optimized for typing email addresses. Good for your visitors. Also, interestingly, browsers that support HTML5 validation automatically look at any input type=”email” as something that needs the email to be in an appropriate format before the form is sent.

```
<form>
Email: <input type=”email” name=”emailaddress” />
</form>
```

URLs

HTML5 only, partly supported. Like the email attribute all browsers that don’t support it see input type=”url” the same way they see input type=”text” while the iPhone responds by offering an alternate keyboard on the screen which is optimized for typing web addresses. Good for your visitors.

```
<form>
Website: <input type=”url” name=”webaddress” />
</form>
```

Submit Button

A submit button is used to send form data to a server. The data is sent to the page specified in the form’s action attribute. The file defined in the action attribute usually does

something with the received input:

```
<form name="input" action="html_form_action.asp" method="get">
Username: <input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

If you type some characters in the text field above, and click the "Submit" button, the browser will send your input to a page called "html_form_action.asp". The page will show you the received input.

Reset Button

The reset button works like the submit button, except that it clears the completed form instead of sending the information.

```
<form>
<input type="reset" value="Reset" />
</form>
```

Button

A general-purpose button. The element <button> is preferred if possible (i.e. if the client supports it) as it provides richer possibilities.

```
<form>
<input type="button">
</form>
```

Image

Defines an image button. The image URL may be specified with the src attribute.

```
<form>
<input type="image" src="images/filename.ext">
</form>
```

File

A file select field (for uploading files to a server).

```
<form>
<input type="file" src="filename.ext">
</form>
```

Hidden

Hidden inputs are not visible in the rendered page, but allow a designer to maintain a copy of data that needs to be submitted to the server as part of the form. This may, for example, be data that this web visitor entered or selected on a previous form that needs to be processed in conjunction with the current form.

```
<form>
<input type="hidden">
</form>
```

Planning Forms

Using forms on your website can be useful for several reasons. You can have a form act as your email contact, so that you can get information from your visitor in an organized fashion. You can use a simple form as a poll or survey. Web forms can be used to enter shipping or credit card data to order a product or can be used to retrieve data.

HTML forms are fairly simple to build and use. For more complex forms , you can combine HTML with various scripting languages to allow developers to create dynamic websites. This includes both client-side and/or server-side languages. We will focus on simple HTML forms here.

Use forms only when you feel they are necessary or useful – to the visitor, not just the client. Decide form information that is needed in order to successfully craft the questions. Determine if the form will post or get data, and if the form needs to work with a database. For database-related forms, you'll want more extensive tools. For simple forms, you can aim for:

- Contact information: first name, last name, email address.
- Demographic information: gender, location.
- Choices information: visitor needs, preferences, actions needed.
- Comment information: for items not queried about in the form.

Form Redirect Pages

When your visitor submits a form, it is common practice for you to redirect them to a page that informs him/her that the form was submitted. This can be a thank-you page, or a redirect to a page on your site that offers more contact information or useful information. If there is no redirect, you visitor might not know if the form was submitted, which is poor usability and accessibility practice.

When you create a redirect page, the input value to that form must be an absolute URL, even of the page is part of your own website. This is because a form handler will redirect a visitor to a 'live' web page, and a visitor cannot be redirected to a URL that exists only on your own computer.

Here are a few tips for building your redirect page:

- Use same design 'template' as the rest of the website, or at least the same template you used for the form page itself.
- Have a thank-you/submission acknowledgement message on the page.
- Have links to other areas on your website if you are not using the full website temple (with the site navigation).

Testing the Form

You have a couple of ways you can test your form.

- Run your form through the W3C Validator to so it can check your nesting and proper form elements use.
- Complete and submit the form, and see if you are redirected properly to the page you created for a redirect location.
- Check the source where you expect the form information to arrive to verify it has arrived. For instance, if you are having your form send information to an email, you would check your email program for the arriving email, and verify that it provides the information you expect.

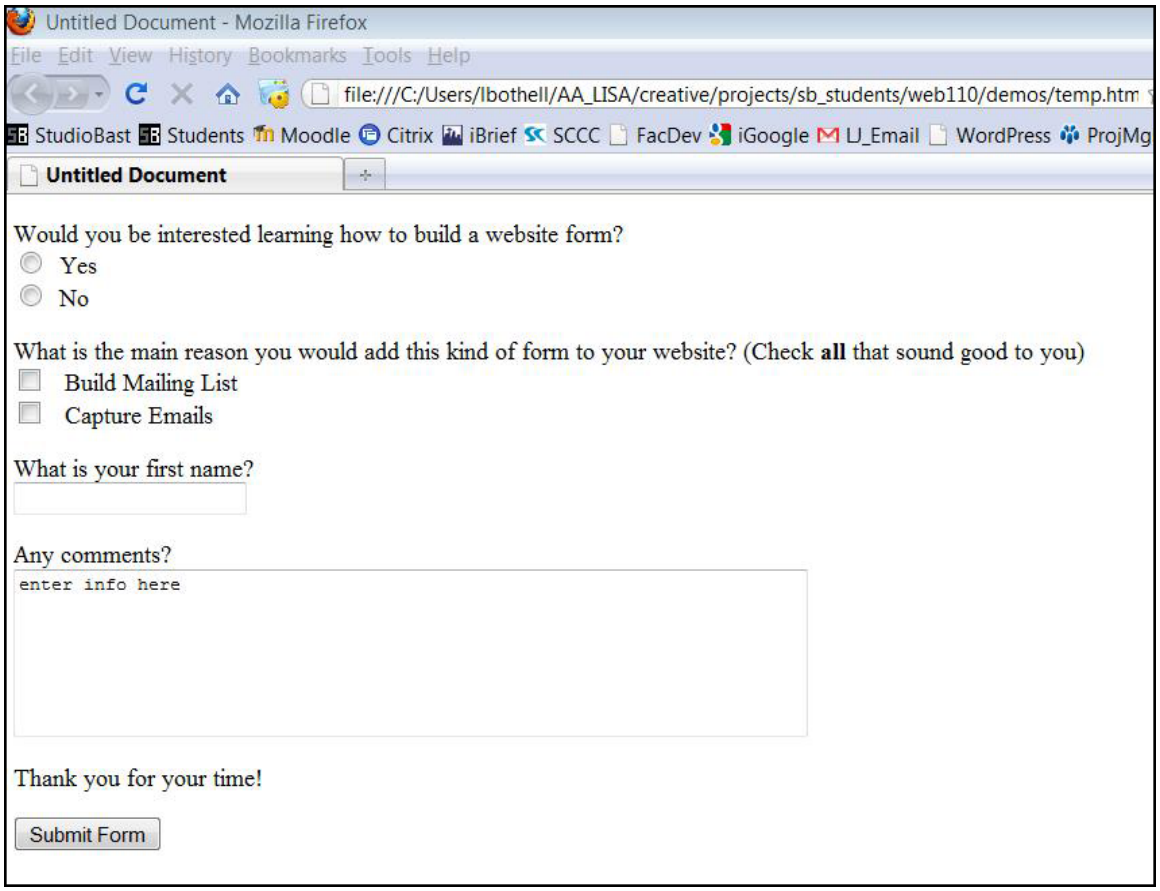
What We Learned

Here is a simple example of a form that lets you send information by email. This particular form was designed to work with a specific form handler, so the hidden input information you might use may be different.

```
<body>
<form name="basicform" method="post" action="URL">
<p>Would you be interested learning how to build a website form?<br />
<input type="radio" name="Interested in learning how to build a website
form?" value="yes" />  Yes <br />
<input type="radio" name=" Interested in learning how to build a website
form?" value="no" />  No
</p>
<p>What is the main reason you would add this kind of form to your web-
```

```
site? (Check <strong>all</strong> that sound good to you)<br />
<input type="checkbox" name="Why use form" value="Build Mailing List"
/>&amp;nbsp;&amp;nbsp; Build Mailing List<br />
<input type="checkbox" name="Why use form " value=" Capture Emails"
/>&amp;nbsp;&amp;nbsp; Capture Emails
</p>
<p>What is your first name?<br />
<input type="text" name="first name" />
</p>
<p>Any comments?<br />
<textarea name="Comments" cols="70" rows="6">enter info here</tex-
tarea>
</p>
<p>Thank you for your time!</p>
<!– This is my submit area –>
<input type="submit" name="submit" value="Submit Form" />
<!– These are the hidden required elements to make a submit button work
for this specific form handler –>
<input type="hidden" name="Thanks" value="AbsoluteURL"  />
<input type="hidden" name="Subject" value="Contacting YourName about
Forms for Website" />
<input type="hidden" name="ToFirst" value="FirstName" />
<input type="hidden" name="ToLast" value="LastName" />
<input type="hidden" name="ToEmail" value="YourName@domain.com" />
<input type="hidden" name="RedirectThanks" value="RelativeFileName.
html" />
</form>
</body>
```

Example Form



18 Frames

Frames allow a visual HTML Browser window to be split into segments, each of which can show a different document. This allows for lower bandwidth use, as repeating parts of a layout can be used in one frame, while variable content is displayed in another. This comes at a significant usability cost, especially in non-visual visitor agents. Because of this cost, frames (excluding the iframe element) are only allowed in the HTML 4.01 Frameset.

In HTML 4.01, a document may contain a head and a body or a head and a frameset, but not both a body and a frameset. However, iframe can be used in a normal document body. In HTML 5, frames will be obsolete.

<frameset>…</frameset>
Contains the frameset. The frames layout is given by comma separated lists in the rows and cols attributes.

<frame> or <frame/>
Delimits a single frame, or region, within the frameset. A separate document linked with the src attribute appears inside.

<noframes>…</noframes>
Contains normal HTML content for visitor agents that don't support frames.

<iframe>…</iframe>
An inline frame places another HTML document in a frame. Unlike an object element, an inline frame can be the "target" frame for links defined by other elements and it can be selected by the visitor agent as the focus for printing, viewing its source, etc. The content of the element is used as alternative text to be displayed if the browser does not support iFrames.

Frames Usage

In general web design, frames are not used much anymore, since other web languages like PHP and ASP allow web authors to create server-side includes instead. Occasionally iFrames are still in use, but not all browsers support them nor are they attractive for visi-

tors. However, in some websites still in use by various employers, in the form of content management systems, frames are still in use.

Frames usage made sense when internet connection speeds were much slower. You could use a frameset where one frame displayed your banner segment, another our navigation segment, and a third your footer segment – and kept the same content of these three frames displayed whenever the content segment frame was updated. Only the content frame changed, which made web page loading faster.

Server-side includes now in use do this same thing, but allowing you to create separate files for your banner segment, navigation segment, footer segment, and repeating sidebar segments, and linking those files to your main web page where you change only the content. You can learn more about this when you study the Web languages that support them, but HTML/XHTML does not go this far.

In frames you use one main page and swap parts for other pages. Each coded page needs all head section information, however.

Frameset Page

Your main frames page, index.html, is known as a Frameset page in frames use. It is the pager where you set the "structure" of how many frames will load into it, and it uses <frameset> instead of <body> tags.

This frameset page also requires to use the frameset doctype:<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

In the frameset you create links to separate pages that will be opened inside the frameset, such as your banner, nav, guts, and footer pages. Each of these web pages needs to use the standard doctype and body tags so that they display properly. Only the frameset page uses the frameset doctype and frameset tags in order to pull in these other pages for viewing.

The pages that are pulled in need to be named so that you can direct the frameset where the file opening locations will be.

Browser/Frame Window Types

- **_self:** Default. The new page will load in the same window that the link is in. If the link is in a frame, it will load inside the frame.
- **_blank**: The Web linked page loads in a new window and does not name the window.
- **_top**: The new page will load into the top window no matter how many framesets deep the link is embedded.
- **_parent**: The new page will load in the parent frame or window.
- **Named (top, left, main, header, nav, guts, etc.)**: used in frames to direct file opening locations.

Controlling Frames

Some of the original frame formatting has become deprecated, like margins.

- Borders and spacing: shows or doesn't show borders of frames. Ex. frameborder=NO framespacing=0 border=0
- Scrolling: shows or limits scrolling on sides of frame(s). Yes-absolute, no-none, auto-as needed, default.
- Resize: Allow or disallow resizing of frame area. For non-resizable windows, add the parameter "noresize".
- Noframe: Add the <noframes> tag for browsers that don't support frames.

Examples

Frameset Page

```
<?xml version="1.0" encoding="utf-8"?>
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"  "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd"><html xmlns="http://
www.w3.org/1999/xhtml">
<head>
<title>Basic HTML Frames Beginner Page </title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="content-style-type" content="text/css">
<meta name="author" content="YourName/>
<meta name="copyright" content="YourName" />
<meta name="description" content="Frames Page." />
<meta name="keywords" content="YourName, etc." />
</head>
```

```
<!–PAGE LAYOUT BEGINS HERE–>
<!–START FRAMESET HERE - index.html–>
<!–Note: To create 2 or more COLUMNS, you need multiple framesets. Not
doing in this demo. –>
<frameset rows=”70,45,*” frameborder=”0” > <!– 70px high, 45px high,
remaining height open –>
<frame src=”header.html” name=”header” id=”header” scrolling=”no”
noresize=”noresize” marginwidth=”0” marginheight=”0” />
<frame src=”nav.html” name=”footer” id=”footer” frameborder=”0”
scrolling=”no” noresize=”noresize” />
<frame src=”guts.html” name=”guts” id=”guts” frameborder=”0”
scrolling=”yes” noresize=”noresize” />

<noframes>
 <body>
  <p>To view this website, your browser must support “frame” technology</
p>
 </body>
</noframes>
</frameset>
<!–END FRAMESET HERE - index.html–>
</html>
```

Header Page

```
<!–Using UTF 8 encoding since this works with the quirky framesets and
frame files –>
<?xml version=”1.0” encoding=”utf-8”?>
<!doctype html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”><html xmlns=”http://
www.w3.org/1999/xhtml”>
<head>
<title>Basic HTML Frames HEADER Page </title>
<meta http-equiv=”Content-Type” content=”text/html; charset=utf-8” />
<meta http-equiv=”content-style-type” content=”text/css”>
<meta name=”author” content=”YourName/>
<meta name=”copyright” content=”YourName” />
```

```
<meta name="description" content="Frames Page." />
<meta name="keywords" content="YourName, etc." />
<!–HEAD SECTION CSS STYLE INFORMATION HERE –>
<style type="text/css">
body {font-family:Verdana, Futura, Arial, Helvetica, sans-serif; color:#000;}
h1 {font-weight:normal; color:#930;}
h3 {color:#033;}
h4 {color:#333;}
a:link {color:#033; text-decoration:none;}
a:hover {font-size:120%; color:#333; text-decoration:none;}
a:visited, a:focus {color: #666; text-decoration:none;}
</style>
</head>
<body style="background-color:#ccff66;">
<!–PAGE LAYOUT BEGINS HERE–>
<!–START BANNER AND WEBSITE IDENTIFICATION HERE - header.html–>
<h1>I LUV COFFEE</h1>
<!–END BANNER AND WEBSITE IDENTIFICATION HERE - header.html–>
</body>
</html>
```

Navigation Page

```
<!–Using UTF 8 encoding since this works with the quirky framesets and
frame files –>
<?xml version="1.0" encoding="utf-8"?>
<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://
www.w3.org/1999/xhtml">
<head>
<title>Basic HTML Frames NAV Page </title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="content-style-type" content="text/css">
<meta name="author" content="YourName/>
<meta name="copyright" content="YourName" />
<meta name="description" content="Frames Page." />
<meta name="keywords" content="YourName, etc." />
```

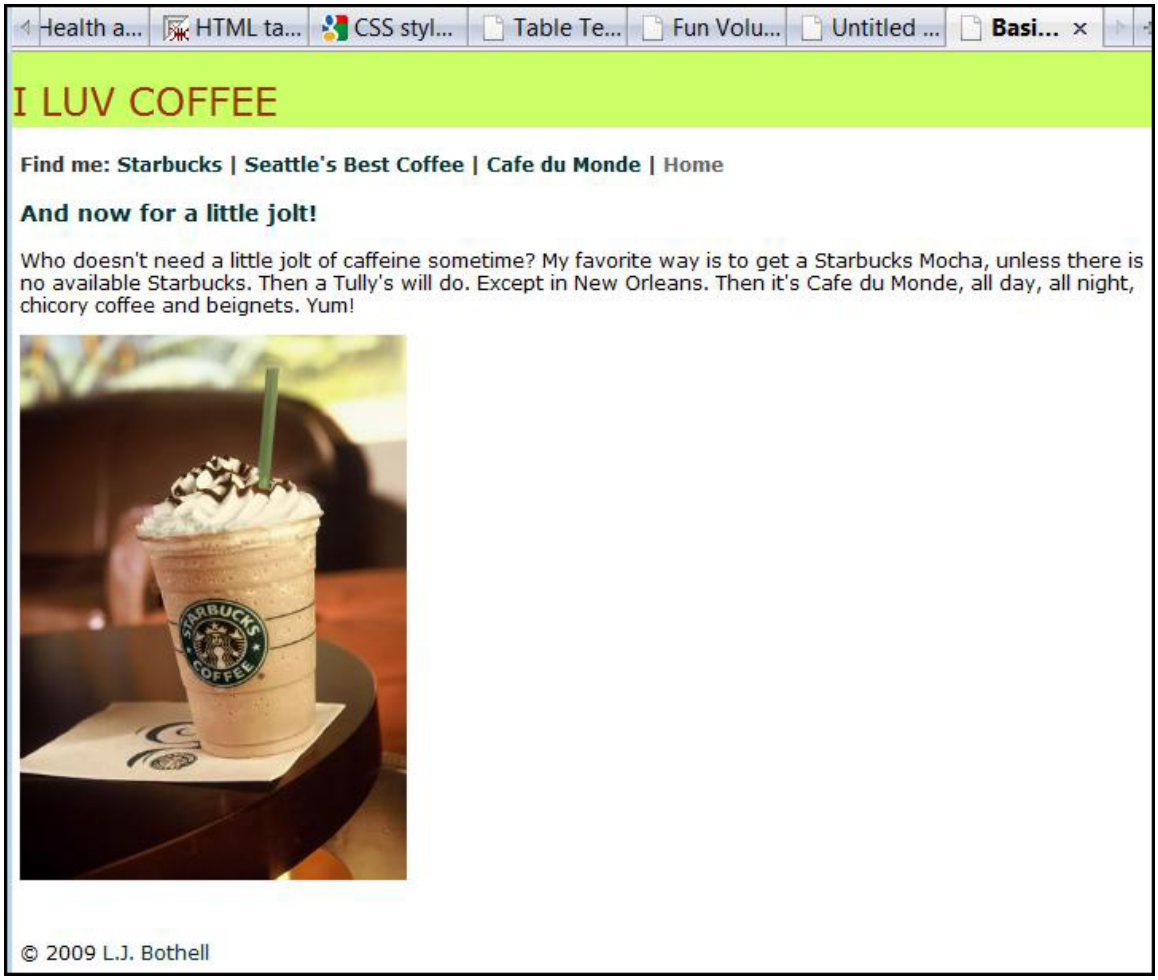
```
<!–HEAD SECTION CSS STYLE INFORMATION HERE –>
<style type=”text/css”>
/*USE THIS STYLE OF COMMENT FOR CSS SECTION*/
body {font-family:Verdana, Futura, Arial, Helvetica, sans-serif; color:#000;}
h1 {font-weight:normal; color:#930;}
h3 {color:#033;}
h4 {color:#333;}
a:link {color:#033; text-decoration:none;}
a:hover {font-size:120%; color:#333; text-decoration:none;}
a:visited, a:focus {color: #666; text-decoration:none;}
</style>
</head>
<body>
<!–START NAVIGATION HERE - nav.html–>
<h4>Find me:
<a href=”http://www.starbucks.com/” target=”guts”>Starbucks</a> |
<a href=”http://www.seattlesbest.com/#/” target=”guts”>Seattle’s Best Cof-
fee</a> |
<a href=”http://www.cafedumonde.com/” target=”guts”>Cafe du Monde</
a> |
<a href=”guts.html” target=”guts”>Home</a>
</h4>
<!–END NAVIGATION HERE - nav.html–>
</body>
</html>
```

Guts Page

```
<!–Using UTF 8 encoding since this works with the quirky framesets and
frame files –>
<?xml version=”1.0” encoding=”utf-8”?>
<!doctype html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”><html xmlns=”http://
www.w3.org/1999/xhtml”>
<head>
<title>Basic HTML Frames GUTS Page </title>
<meta http-equiv=”Content-Type” content=”text/html; charset=utf-8” />
```

```
<meta http-equiv="content-style-type" content="text/css">
<meta name="author" content="YourName/>
<meta name="copyright" content="YourName" />
<meta name="description" content="Frames Page." />
<meta name="keywords" content="YourName, etc." />
<!–HEAD SECTION CSS STYLE INFORMATION HERE –>
<style type="text/css">
/*USE THIS STYLE OF COMMENT FOR CSS SECTION*/
body {font-family:Verdana, Futura, Arial, Helvetica, sans-serif; color:#000;}
h1 {font-weight:normal; color:#930;}
h3 {color:#033;}
h4 {color:#333;}
a:link {color:#033; text-decoration:none;}
a:hover {font-size:120%; color:#333; text-decoration:none;}
a:visited, a:focus {color: #666; text-decoration:none;}
</style>
</head>
<body>
<!–PAGE LAYOUT BEGINS HERE–>
<!–START CONTENT TEXT AND IMAGES HERE - guts.html–>
<!–Replace this entire comment line with your website content information -
text, images, nested table, etc.–>
<h3>And now for a little jolt!</h3>
<p>Who doesn't need a little jolt of caffeine sometime? My favorite way is to
get a Starbucks Mocha, unless there is no available Starbucks. Then a Tully's
will do. Except in New Orleans. Then it's Cafe du Monde, all day, all night,
chicory coffee and beignets. Yum!</p>
<p><img src="../images/mocha.jpg" width="360" height="505"
alt="mocha" title="mocha" /></p>
<br />
<p> &copy; 2009 <a href="mailto:ljbtrainings@studiobast.com">L.J. Both-
ell</a></p>
<!–END CONTENT TEXT AND IMAGES HERE - guts.html–>
<!–PAGE LAYOUT ENDS HERE–>
</body>
</html>
```

Frames Example



More Information

For more information about frames, check out the http://www.w3schools.com/html/html_frames.asp which gives several frames code examples.

19 Website Launch

Client websites go live on a web host server, which the client pays for and maintains. We've already discussed file transfer protocol (FTP) which you need to use to test our website live, but you can do this on your own free or low cost server. However, your server will generally not be the server used for your client's final website, unless you get into the business of providing web hosting services yourself – a topic for another book.

Web Hosting Service

A web hosting service is a type of Internet hosting service that allows individuals and organizations to make their own website accessible via the Web. Web hosts are companies that provide space on a server they own or lease for use by their clients as well as providing Internet connectivity, typically in a data center. Web hosts can also provide data center space and connectivity to the Internet for servers they do not own to be located in their data center, called colocation.

The scope of hosting services varies. The most basic is web page and small-scale file hosting, where files can be uploaded by FTP or a web interface provided by the web host. The files are usually delivered to the Web as is or with little processing. Personal website hosting is typically free, advertisement-sponsored, or inexpensive. Business website hosting often has a higher expense, and can offer shared or dedicated servers.

Single page hosting is generally sufficient only for personal web pages, and these may come with a series of templates for you to choose from. This is not the option for web authors, since it does not allow much control over the design.

Free website hosting allows you to personalize your own site, upload as you need, etc. However, you may be stuck with advertising on your site, may have serious space, FTP, and content limits. You also don't use your own domain name, but rather get a free subdomain while using the Web host's domain name. Finally, you will likely get little and/or slow technical support if you experience issues.

Fee-based web hosts offer comprehensive packages and support that provides database support and application development platforms (like PHP, Java, ColdFusion, and ASP.net). They usually also provide an interface or control panel for managing the Web server

and installing scripts as well as other services like e-mail. Some hosts specialize in certain software or services (like e-commerce). They are commonly used by larger companies to outsource network infrastructure to a hosting company. They may also provide domain registration and annual renewal of your domain.

You will want to know the hosting uptime of the web host you choose. Hosting uptime refers to the percentage of time the hosted websites (like yours) is accessible via the Internet. Many providers state that they aim for at least 99.9% uptime (roughly equivalent to 45 minutes of downtime a month, or less), but there may be server restarts and planned (or unplanned) maintenance in any hosting environment, which may or may not be considered part of the official uptime promise. Many providers tie uptime and accessibility into their own service level agreement (SLA). SLAs sometimes include refunds or reduced costs if performance goals are not met.

Many large companies who are not internet service providers also need a computer permanently connected to the Web so they can send email, files, etc. to other sites. They may also use the computer as a website host so they can provide details of their goods and services to anyone interested. Additionally these people may decide to place online orders.

Types of Hosting

- **Free web hosting service**: offered by different companies with limited services, sometimes supported by advertisements, and often limited when compared to paid hosting.
- **Shared web hosting service**: one's website is placed on the same server as many other sites, ranging from a few to hundreds or thousands. Typically, all domains may share a common pool of server resources, such as RAM and the CPU. The features available with this type of service can be quite extensive. A shared website may be hosted with a reseller.
- **Reseller web hosting:** allows clients to become web hosts themselves. Resellers could function, for individual domains, under any combination of these listed types of hosting, depending on who they are affiliated with as a provider. Resellers' accounts may vary tremendously in size: they may have their own virtual dedicated server to a collocated server. Many resellers provide a nearly identical service to their provider's shared hosting plan and provide the technical support themselves.
- **Virtual Dedicated Server**: also known as a Virtual Private Server (VPS), divides server resources into virtual servers, where resources can be allocated in a way that does not directly reflect the underlying hardware. VPS will often be allocated

resources based on a one server to many VPSs relationship, however virtualization may be done for a number of reasons, including the ability to move a VPS container between servers. The visitors may have root access to their own virtual space. Customers are sometimes responsible for patching and maintaining the server.

- **Dedicated hosting service**: the visitor gets his or her own Web server and gains full control over it (root access for Linux/administrator access for Windows); however, the visitor typically does not own the server. Another type of Dedicated hosting is Self-Managed or Unmanaged. This is usually the least expensive for Dedicated plans. The visitor has full administrative access to the box, which means the client is responsible for the security and maintenance of his own dedicated box.
- **Managed hosting service**: the visitor gets his or her own Web server but is not allowed full control over it (root access for Linux/administrator access for Windows); however, they are allowed to manage their data via FTP or other remote management tools. The visitor is disallowed full control so that the provider can guarantee quality of service by not allowing the visitor to modify the server or potentially create configuration problems. The visitor typically does not own the server. The server is leased to the client.
- **Colocation web hosting service**: similar to the dedicated web hosting service, but the visitor owns the colo server; the hosting company provides physical space that the server takes up and takes care of the server. This is the most powerful and expensive type of web hosting service. In most cases, the colocation provider may provide little to no support directly for their client's machine, providing only the electrical, Internet access, and storage facilities for the server. In most cases for colo, the client would have his own administrator visit the data center on site to do any hardware upgrades or changes.
- **Cloud Hosting**: is a new type of hosting platform that allows customers powerful, scalable and reliable hosting based on clustered load-balanced servers and utility billing. Removing single-point of failures and allowing customers to pay for only what they use versus what they could use.
- **Clustered hosting**: having multiple servers hosting the same content for better resource utilization. Clustered Servers are a perfect solution for high-availability dedicated hosting, or creating a scalable web hosting solution. A cluster may separate web serving from database hosting capability.
- **Grid hosting:** this form of distributed hosting is when a server cluster acts like a grid and is composed of multiple nodes.

- **Home server**: usually a single machine placed in a private residence can be used to host one or more websites from a usually consumer-grade broadband connection. These can be purpose-built machines or more commonly old PCs. Some ISPs actively attempt to block home servers by disallowing incoming requests to TCP port 80 of the visitor's connection and by refusing to provide static IP addresses. A common way to attain a reliable DNS hostname is by creating an account with a dynamic DNS service. A dynamic DNS service will automatically change the IP address that a URL points to when the IP address changes.

Domains

A domain name is an identification label that defines a realm of administrative autonomy, authority, or control on the Internet, based on the Domain Name System (DNS). When you or your client creates a new website, the name of the domain, which will appear in the URL, is very important. You will want a top-level domain, which is the highest level of domain names of the Internet: http://www.**yourdomain**.com.

Professional/SEO Naming

The website domain name, and the impression it makes, is crucial. An important purpose of domain names is to provide easily recognizable and memorable names for visitors. You need something that is related to the client's business, client's name, product, etc.

Do:

- Use a domain name that is also the website name when possible.
- Create a variation of your preferred domain name if the one you want is already chosen. For instance, if you want *smithdesigns*, and it is already chosen, consider something like *designsmithy*, *designsbysmith*, *smithdesignninja*, etc.
- Consider using .net or .biz if your preferred name isn't available as .com, *but only* if the website is a noncompetitive one. If you are competing against other businesses with similar names, stick with .com.
- Brainstorm the websites top keywords, if needed. You want an intuitive name, not one that doesn't reveal anything the visitor can expect.
- Choose a name that is relevant to the client's business or name, or the website purpose.
- Go for a name that is part of the website branding, rather than something generic.
- Consider how the name can be useful in SEO that is relevant to how your site should be indexed.
- Use clean language and spelling (unless off-spelling is part of the charm).

- Be willing to create a longer name if it gives you the necessary options, but keep it professional.

Don't:

- Use a confusing domain name - think about how the domain name looks. For instance, a therapists' network is named therapistfinder.com. The rapist finder?
- Choose misleading names that make a visitor think they are going something else.
- Decide on a name that is completely unrelated to what the website is about.
- Select a name that is similar to a popular business/person, since this can border on infringement.
- Create a name with a mix of odd characters, unnecessary hyphens, or odd capitalizations.
- Create a really long domain name that no one can remember or spell.
- Be trendy. Trends end fast and the website should be around for awhile.

Getting a Domain Name

The right to use a domain name is delegated by domain name registrars which are accredited by the Internet Corporation for Assigned Names and Numbers (ICANN), the organization charged with overseeing the name and number systems of the Internet. In addition to ICANN, each top-level domain (TLD) is maintained and serviced technically by an administrative organization operating a registry. A registry is responsible for maintaining the database of names registered within the TLD it administers. The registry receives registration information from each domain name registrar authorized to assign names in the corresponding TLD and publishes the information using a special service, the whois protocol.

Registries and registrars usually charge an annual fee for the service of delegating a domain name to a user and providing a default set of name servers. Often this transaction is termed a sale or lease of the domain name, and the registrant may sometimes be called an owner, but no such legal relationship is actually associated with the transaction, only the exclusive right to use the domain name. More correctly, authorized users are known as registrants or as domain holders.

You can register a new domain yourself, but of you do, and you later decide to use a web host which provides domain name services, you will have to transfer your domain name to them. It can be more efficient for you to decide where you will host your website first, get that web host set up, then register your domain name through them. The Web host will have a process and explanation of how to do this.

Domain Renewals

You can renew or extend your domain name at any time. You don't have to wait until you get a notice that a domain is close to expiration, and if you have your domain name registered/handled through your web host, you should get expiration reminders so you know.

Renewing is required because the registration is for a finite period of time, usually one year. However, you can renew for one year or up to 10 years.

Domain Hijacking

Domain hijacking or theft is the process by which registration of a currently registered domain name is transferred without the permission of its original registrant (you), generally by exploiting a vulnerability in the domain name registration system.

Domain names expire after a fixed period of time (usually a year) and become available to the public. If their original owner allows them to expire - even momentarily - they may be immediately purchased by another party. The difficulty here is that many of these domain hijackers will then be happy to resell your own domain name back to you, for a very inflated price. Don't let this happen – know when you annual domain renewal schedule is and stay on top of it, or renew early.

Listing Websites

Back in the day, before search engines became so sophisticated, websites didn't just get found. You needed to register them with search engines. Nowadays the biggies like Google, Bing, and Yahoo tend to find websites sooner rather than later, so technically you do not need to register them, especially if they are personal, social networking, and blog sites.

However, clients like and need their professional websites to be indexed and available as soon as possible after they launch, so registering websites for launch still doesn't hurt. You can go to Google, Bing, and Yahoo, and do a search on registering your site with them; they will have an easy online form where you can give site information.

You can also find free website listing options, where you complete a form to get your site listed and the form will be sent to a bunch of smaller search engines to get indexed/listed. This can be useful if you need a site to get onto special research/scientific engines and such.

Whichever you choose, make sure you have solid meta tags: description, title, author, niche keywords - this, along with good SEO content, is what the indexing depends on.

and can also establish the placement on the search engine results more accurately.

Web Sitemaps

Web sitemaps are different than the design sitemap you may create while planning the layout of your website. Web sitemaps exist for audience and search engine use, and there are two types you will want to create for your website.

Website Sitemap

The website sitemap is simply another page of your website, following the same template, navigation, and content structure. However, in this page, you list all (or many) of your pages so that a visitor can simply click and go. You would list your main pages, sub pages, and your utility pages. You can choose to list pages:

- By their order in your main navigation, with sub pages listed below each.
- Alphabetically
- By subject
- By purpose

Whichever you choose, make sure to make the sitemap very easy and intuitive for your visitors, and to link it to all pages of your website – in the footer, for instance.

Indexing/SEO Sitemap

The indexing sitemap is an XML file you create solely for search engines to use to index your website. Sitemaps are a way to tell a search engine about pages on your site it might not otherwise discover in the normal crawling process. You will want to search for "search engine sitemap" in a search engine to get the most current protocols, but it's pretty easy.

Indexing sitemaps are particularly helpful if:

- Your site has dynamic content.
- Your site has pages that aren't easily discovered by bots during the crawl process—for example, pages featuring rich AJAX or images.
- Your site is new and has few links to it. Bots crawl the Web by following links from one page to another, so if your site isn't well linked, it may be hard for the search engine to discover it.
- Your site has a large archive of content pages that are not well linked to each other, or are not linked at all.

Web Analytics

Web analytics is the measurement, collection, analysis and reporting of internet data for purposes of understanding and optimizing web usage. It measures website traffic and can also be used as a tool for business research and market research, by providing data on the number of visitors, page views, etc to gauge the traffic and popularity trends which helps doing the market research. There are two categories of web analytics:

- Off-site web analytics refers to web measurement and analysis regardless of whether you own or maintain a website. It includes the measurement of a website's potential audience (opportunity), share of voice (visibility), and buzz (comments) that is happening on the Internet as a whole.
- On-site web analytics measure a visitor's journey once on your website. This includes its drivers and conversions; for example, which landing pages encourage people to make a purchase. On-site web analytics measures the performance of your website in a commercial context. This data is typically compared against key performance indicators for performance, and used to improve a website or marketing campaign's audience response.

Web analytics is a specialty unto itself, and can't be covered here. However, if you focus on clients who need sales and marketing support as part of your web projects, this will bear research to determine what analytics tools and skills you can add to your toolbox.

Online Website Tools

You can access a variety of tools for webmasters online. These can give you the bigger picture on current linking and indexing practices, search engine optimization, analytics, and more. Check them out:

- Google Webmaster Central
- Bing Webmaster Tools
- Backlink checkers
- Keyword density checkers
- Rank checkers
- URL redirect checkers
- HTML optimizers
- CSS optimizers
- Reciprocal link checkers
- Domain Whois

Web Authoring **Boot Camp**

STEP IT UP!

section 3

Intermediate code and styling

In this section, we're going to look at more advanced styling for your websites, including Cascading Style Sheets and a few dynamic bits you can add to your site.

20 Styling Code, Part 2

We discussed Cascading Styles (CS) earlier, with a focus on inline style rules and embedded head section cascading styles. However, web authors typically link their web page to an external file, called a Cascading Style Sheet (CSS). A CSS can also allow the same web page to be presented in different styles for different rendering methods, such as on-screen, in print, on mobile devices, by voice (when read out by a speech-based browser or screen reader) and on Braille-based, tactile devices.

As a result your future HTML code production should focus on CSS with XHTML. CSS files are easy-to-organize and able to be parsed by every browser with as much uniformity in interpretation as possible. They also allow you to store your styles separately from your XHTML code and to make changes to one file that radiate throughout all pages in your site. Need to change a style? It *cascades* throughout all the documents that use it.

This book is for beginning web authors, so while I will cover some CSS styling in this and the next chapter, you will benefit from picking up one or more CSS-specific books, like those by Eric Meyer, the *Sam's Teach Yourself* books, *Peachpit Press' Visual Quick-Start* books, and *Sitepoint's* books. Just make sure to look for one or more that offers plenty of examples of the kind of CSS layout and styling that you see as current on the WEB. You can also find excellent resources on the Web, like *Smashing Magazine* and *A List Apart*.

Until now, I have been referring to styling as CS – Cascading Styles. However, since CSS (Cascading Style Sheets) is the preferred and professional web authoring method for web page styling, I will now refer to all cascading styling as CSS. I will also swap out *CSS* with *style sheet* as needed. Not to confuse you or anything.

Planning Styles With Templates

Essentially, you should plan your styles with your web project's template page, but with CSS your styles would be built in a separate style sheet file instead of inside the template. Your template page would link to the CSS file in the head section, so that as you modify styles in the style sheet, they will show up in your template page. Then, once you are satisfied with your template, you can use it, already linked to the style sheet(s), to build out your entire website.

There are many ways to link style sheets to HTML, each carrying its own advantages and disadvantages. New HTML elements and attributes have been introduced to allow easy incorporation of style sheets into HTML documents. We'll look at the two most commonly used.

Linking to CSS

A CSS file may be linked to an HTML page through the link element in the page's head section.

The type attribute is used to specify a media type - text/css for a Cascading Style Sheet. This alows browsers to ignore style sheet types that they do not support. You also need to configuring your web host server to send text/css as the content-style-type for CSS.

```
<head>
<meta http-equiv="content-style-type" content="text/css">
<link rel="style sheet" media="screen" type="text/css" title="Website Screen
Styles" href="screenstyles.css" />
</head>
```

Media values include:

- **Screen**: For presentation on non-paged computer screens.
- **Print**: For output to a printer.
- **Handheld**: For presentation on handheld mobile devices.
- **Projection**: For projected presentations.
- **Aural**: For speech synthesizers.
- **Braille**: For presentation on Braille tactile feedback devices.
- **TTY**: For character cell displays (using a fixed-pitch font).
- **TV**: For televisions.
- **All**: For all output devices.

Importing CSS

A style sheet may instead be imported with CSS's @import statement. This statement may be used inside the style element:

```
<style type="text/css" media="screen, projection">
<!–
 @import url(styles/screenstyles.css);
–>
</style>
```

Local Page Styles

Other CSS rules that apply to only the single page may also be added in the page's style element, but all @import statements must occur at the start of the style sheet. Any rules specified in the page's embedded cascading styles override conflicting rules in the imported CSS files. For example, even if one of the imported style sheets contained body {background: aqua}, and you told your specific page style to make the body background black, the page would have a black background.

```
<style type="text/css" media="screen, projection">
<!–
 @import url(styles/style.css);
–>
body {color:#000;"}
</style>
```

```
body {background-color: #ffffff;
    color: #000000;
    margin:0;
    padding:0; } /* White background, black font, no margin or padding */

h1, h2, h3, h4, h5, h6, body, p {
    font-family: verdana, arial, sans-serif;
    font-weight: normal;} /* Same font for all body elements and specified for all headers and
paragraph text. Font-weight normal instead of default bold for all headers. */

h2 {color:#003366; margin-bottom:0;} /* can use #036, Header 2 is a medium blue */
h3 {color:#336699;} /* can use #369, Header 3 is a lighter blue */
h3.red {color:#990000;} /* Header 3 style remains the same, but its class red is a red color instead
*/
h4 {font-weight:bold;} /* Header 4 is bold */

.intense {font-weight: bold;
    color:#000000;} /* This class of bold and black style can be used with any text tag */
.script {font-style: italic;} /* This class of italic style can be used with any text tag */
.smallfont {font-size: 9pt;}  /* This class of smaller font style can be used with any text tag */

/* STANDARD BODY NAVIGATION LINKS  - LVFHA*/
a:link {color: #990000; } /* can use #303, unvisited link is a maroon color */

a:visited, a:focus {color: #666666;} /* can use #666, visited link is a medium gray */

a:hover {font-size:120%; color: #000000;} /* can use #000, mouseover link makes text bigger */

a:active {color: #990000;} /* active/mouse-click link is red */

hr {border:none;
    background-color:red;
    color:red;
    height:1px;
    width:200px;}/*The allows you to color a Horizontal Rule so it doesn't look standard  */

/* IMAGES */
img {border: 1px solid #000000;} 1px black border around all images

/*img {border-width:1px; border-style:solid; border:#000000;}  1px black border around all images */

/* TABLE STYLES */
table.small {border-color:#333333; border-style:solid; border-width:5px; width:100%; } /* All tables'
 border is 2px solid grey, width 100% */
tr#banner {background-color:#CCFF99;}
td.small {border: #ccc solid 3px; width:50%;}  /*All table data cells (columns) are 50% wide
(assuming 2 column table), with only a right border */
```

CSS Style Sheet

Multiple Style Sheets

The order in which style sheets are imported determines how they cascade. The last imported style sheet has greater impact than earlier ones, so if you list three style sheets, and they have conflicting styles, the style in the last listed sheet would win.

Why would you use multiple style sheets? Several reasons:

- Styles for massive websites can be made more manageable by using style sheets for each major section.
- To modularize major element sections into several small sheets, such as by structure-related elements, text-related, and graphic-related elements.

- To have different style sheets for screen, mobile, and print output.
- To have variations of the same style sheet which allow for different browser interpretations.

```
<link rel="style sheet" media="screen" type="text/css" title="Website Screen Styles" href="screenstyles.css" />
<link rel="style sheet" media="print" type="text/css" title=" Website Print Styles" href="printstyles.css" />
```

Conditional Style Sheets

In a happy web authoring world, you could use only one style sheet that would make all browsers see your website properly. Unfortunately, older browsers have quirks, Internet Explorer interprets the layout *box model* differently, and even current browsers still have slight issues with what code they do and do not support. As a result, to get your web page to look correct in multiple browsers, you might have to "hack" by using conditional style sheets.

You can provide an alternative style sheet for the browser which gives your page the most trouble. In this second style sheet, you would adjust the layout measurements and other style quirks as needed. Then, you would tell whichever browser your web page is opened in to apply the default style sheet, unless that browser happens to be the trouble browser. Then it would look for the alternate style sheet:

- Tell the browser which sheet is the **default**
- Tell the browser that if it is the alternate browser, to use the **alternate** sheet, else
- Tell the browser that if it is **not** the alternate browser, to use the **default** sheet.

One example (you can research variations online):

```
<link rel="style sheet" media="screen" type="text/css" title="Website Screen Styles" href="screenstyles.css" />
<!–[if IE]> <link rel="style sheet" media="screen" type="text/css" title="IE Website Screen Styles" href="ie_screenstyles.css" /> <![endif]–>
<!–[if !IE]> <link rel="style sheet" media="screen" type="text/css" title="Website Screen Styles" href="screenstyles.css" /> <![endif]–>
```

CSS for Printing

You can use CSS to create amazing websites, but what if someone wants to print out information from the site? Some sites do provide pdfs for folks to print out, but many don't - they just assume folks might print from the screen. In many cases, parts of the screen

don't translate well to print - sometimes the sidebars or navigation don't print, or unique color schemes of the site look unreadable.

Knowing how to make a style sheet for printing can be very helpful - especially if you have critical take-away information for a visitor, like a table of contact information, a resume, a list of resources, etc.

First, you need to make a copy of your existing style sheet and give it another name, like printstyles.css. Then you need to make some changes in your print style sheet for it to be effective:

- Select appropriate fonts and font sizes, using points not pixels since points translate best for printing.
- Remove background colors and images, and use only colors that print well in black and white - with good contrast.
- Adjust page margins, if needed.
- Set page breaks to keep elements from being divided between 2 printed pages: p {page-break-inside:avoid;}
- Hide items that don't need to be printed, like sidebars, ads, rollover navigation, etc, like #leftnav {display:none}
- Change the display of the links so that theu are readable and do have an underline: a {color:black;}
- Show the URL of the links listed: a:after {content:"("attr(href)")";}. Unfortunately, this does not work with IE.

Finally, you need to use your <head> section's link to a style sheet to specify that you are linking to a print CSS:

```
<link rel="style sheet" media="print" type="text/css" title="Website Print Styles" href="printstyles.css" />
```

CSS for Mobile Apps

You'll need even more extreme CSS clean-up if you want website content to show up on mobile devices like cell phones, PDAs, and other handhelds. The problem is that this is still the early stage of this part of the styling field, so pretty much everything is inconsistent. You have two things to consider: miniaturization and mobilization.

- Miniaturization is trying to make an existing style sheet work, just much smaller.
- Mobilization is trying to redesign the spreadsheet so that only essential items are

styles/translated, but also small. The idea is to minimize the amount of clicking, scrolling, and download time.

First, you need to make a copy of your existing style sheet and give it another name, like mobilestyles.css. Then you need to make some changes in your mobile style sheet:

- Hide anything extra to minimize kbs downloaded to the phone - speed and low cost is essential.
- Make sure all images on your site have alt text so visitors can turn images off. Keep images very optimized/small.
- Avoid floats, multiple columns, absolute positioning, tables, pop-up windows, and large headers.
- Avoid background images and bad contrast colors.
- Adjust horizontal margins.
- Avoid fixed width design.
- Add more "back to top" links in your xhtml file, but give them a class name like "top". Then, in your normal style sheet, create the class top {display:none;} so your computer viewers don't see all the extra back to top links.
- Add border-bottoms between divisions to help visually organize content.
- Center content, and reduce the size of elements like the links.

Then, you need to use your <head> section's link to a style sheet to specify that you are linking to a mobile CSS:

```
<link rel="style sheet" media="handheld" type="text/css" title="Website Mobile Styles" href="mobilestyles.css" />
```

Many mobile browsers have implemented the handheld media type, but in different ways:

- Some read only the handheld style sheet.
- Some read only the handheld style sheet if there is one, but default to the screen style sheet otherwise.
- Some read both the handheld style sheet and the screen style sheet.
- Some read only the screen style sheet.

Testing CSS

CSS Validator

If you want to validate your CSS style sheet embedded in the head section of your HTML file, you should first check that the XHTML DAT you use is valid. Then, the W3C has a CSS Validator you can use to check your CSS by URI, by file upload, or by direct input. This works the same way as the W3C Validator for HTML.

CSS Clean-up

You can also use CSS cleanup tools, like CSSTidy, to clean up your style sheet code. These tools do not write your code – you need to build your own expertise on that. They can help catch repetitive styles, conflicts, and potential inheritance problems. This can help minimize the file size of your style sheet, which can help increase your site loading speed and load web pages faster.

Mobile Testing

You can use simulators on Windows system software to see how various small screen devices will view your site. Google Openwave Phone Simulator, Nokia simulator, and others from Palm, Blackberry, etc. For Mac, Opera has a View/Small Screen.

Cascading Style Sheets Styling

Style Rules Review

Style rules are the method you use in cascading styles to create a style to apply to your HTML code. Style rules have a standard format plus two basic parts:

- Code style: selector, brackets around property/value pairs, semi-colon concluder for each property/value pairs - **p {color:#000; }**
- Selector: Determines which elements are affected - **p** {**color**:#000;}
- Declaration: One or more property/value pairs - p {color:**#000**;}
- ALL rules need to have the property/value paired style info inside brackets p **{**color:#000; **}**
- All property/value pairs need a semi colon to 'finish' it - p {color:#000**;** }

In addition, you can create more efficient style rules by combining both selectors and property/value pairs, and even by having multiple values for a property:

```
h1, h3, p {font-family: Arial; color:#000;}
table {align:left; border:1px #000 solid;}
```

Finally, you can (and should) also use comments in your style sheets. There is a special format for this: /* COMMENT TEXT */

Classes and IDs

You can create styles for your page elements, like body, table, <p>, and more. However, what if you want to create a style to use only on some paragraphs, for instance, but don't want to repeatedly type inline style rules for every situation? Easy! CSS lets you create classes and ids that you can style and apply to only specific HTML elements when you want them.

Class Styles: A class is a style you can and will use repeatedly throughout a web page / website. This would be items like special styles for headers, text, tables, images, etc. The class is coded with a period in front of the class name: .selector.

With a class, you style using the standard style rules. To use the styled class, you would code the specific element (like paragrapg tag) to call upon the special class that is inside the stylesheet. For instance, the class *intense* will turn all text tags using the class to bold.

```
.intense {font-weight: bold;}
<p class="intense">I am intense text.</p>
```

ID Styles: An ID is a style that you can only use once in a web page. This you be for once-only special styles like a rollover navigation, a banner element, a footer element, and any division layout items. The ID is coded with a hash mark in fromt of the ID name: #selector.

```
#headbar
    {background-color: #000000;
    color: #ffffff;
    padding: 1em 2%;}
<div id="headbar">The DIV contents here </div>
```

It can be tempting to use only classes if you know that you will only use the ID-like item once. However, DO NOT! Follow CSS convention because the ID is commonly recog-

nized by other designers and the W3C as being a once-used item, and will be edited that way. Remember, other designers may need to see/adapt your style sheets.

Class/ID Coverage

Full Coverage: If you want a property to work throughout a site, you can set it as a Class or ID. For instance, **.red {color:red;}** turns all text red that you classify as red – whether paragraph, header, lists, etc.

Specific Coverage: If you want to affect only a specific selector, such as all text in a specific table cell, you would list the selector, then the class or id: td.red {color:red;} This means that every time you create a new table data cell <td>, the texxt inside will be red.

Here is a more complex example, where an ID has been created called catlinks in order to style a specific type of link on a web page. The catlinks ID can use both selectors and pseudo classes.

```
#catLinks h5 {text-align: left; font-weight: bold;}
#catLinks li a:link {color:#666; text-decoration: none; font-weight: bold;}
#catLinks li a:hover {color:#202020; text-decoration: none; font-weight: bold;
}
#catLinks li a:visited {color:#333; text-decoration: none; font-weight: bold;}
#catLinks li a:focus, a:active {color:#666; text-decoration: none; font-weight:
bold;}
```

Combination Coverage: You can combine multiple classes inside one element:

```
.gray {color:#666;}
.intense {font-weight:bold;}
<p class="gray intense">Gray bold text here</p>
```

Pseudo Classes

A pseudo-class is similar to a class in CSS, but it's not specified explicitly in the markup, like body, paragraph, and table elements are. Some pseudo-classes are also dynamic, and are applied as a result of user interaction with the document (like a form).

A pseudo-class starts with a colon (:). No whitespace may appear between a type selector or universal selector and the colon, nor can whitespace appear after the colon.

Some selectors, like the link <a> tag, have several variations on how it can be styled, depending on whether it is active, hover, visited, etc. These can work regardless of browser, and are not case-sensitive.

```
a:link {color:#333;}     /* unvisited link */
a:visited {color:#666;}  /* visited link */
a:hover {color:#000;}  /* mouse over link */
a:active {color:#333;}  /* selected link */
```

- a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!!
- a:active MUST come after a:hover in the CSS definition in order to be effective!!

Pseudo-classes can be combined with CSS classes, and use the element:pseudo {} style rule syntax.

```
a.red:visited {color:#ff0000;}
<a class="red" href=http://www.google.com" title="Google">Google red
link</a>
```

Most of the pseudo class elements are fairly straightforward by targeting elements in different situations and states. Some of them are a little trickier, so in the table below is a list of a few pseudo elements and their intended usage.

- **:link** - Apply something to an unvisited link
- **:visited** - Apply something to a visited link
- **:active** - Apply something to something that's active, for instance a link is active from when you click it until you release.
- **:hover** - Apply something to what the users hovering over
- **:focus** - When something is focused it's being used, for instance when a user is typing something into a input field its being focused on
- **:first-child** - Apply something to the first child of an element
- **:lang()** - Apply something to an element using a specific language, for instance lang(en) for English. You can then put lang="en" in an html tag.
- **:root** - Applies CSS to the root element, i.e. the <html> tag in an html document.
- **:target** - For instance, if you have a URL like index.html#header, this will allow you to apply CSS to the element with id="header"

- **:enabled** - Apply to all currently enabled inputs
- **:disabled** - Apply to all currently disabled inputs
- **:checked** - Apply to checked input boxes
- **:indeterminate** - For radio and check box elements, when they are neither selected or unselected
- **:default** - Apply to the default element, for instance the default submit button in a form

Pseudo classes can come in handy when trying to increase usability. For instance, *:hover* can make things besides links react to a users hover, letting them know they can do things with it. They can change color, change image, become larger or bolder, and more.

You can apply styling to your form input areas and textareas that only takes affect when a user has clicked into that area using the *:focus pseudo* class. For example, you could change the background and border color on those elements so that the visitor knows they are *selected*.

Graceful Degradation

When a browser doesn't support a class (or any CS), it will fall back on something else. For instance, the fact that older browser don't support rounded corners doesn't mean you can't implement it. It simply means that visitors won't see the rounded corners in older browsers, bu thatt won't affect your visitors in a site-crippling way.

The same goes for pseudo classes. If you're using an older browser which doesn't support *:focus* and the user clicks on an input with a focus CSS class, the visitors won't get the focus effect. It doesn't alter the usability of the input field. This means you can implement most pseudo classes on a basic level (as long as they're not being implemented in a way that'll effect the layout of a website) without worrying about altering user experience in older browsers.

QuickLook

Rollover Navigation Example

```
/* ROLLOVER HORIZONTAL NAVIGATION - GREEN */
ul#navigation
    {margin-left: 0;
    padding-left: .5em;
    list-style-type: none;
    background-color:#006600;
    float: left;
    width: 100%;}

ul#navigation li
    {display:inline;}

ul#navigation a
    {display: block;
    float: left;
    padding: .2em 1em;
    text-decoration: none;
    color: #ffffff;
    border-right: 1px solid #ffffff;}

ul#navigation a:hover
    {background-color: #000000;
    color: #cccccc;
    font-size:100%;}
```

CSS Style Rules Order

Creating style sheets is as personal as the designer. However, when you are getting started, think of ways you can keep your code organized so you can easily see, fix, and change it. Style the biggest/most encompassing items first, then narrow down to the smallest. What does this mean?

- First style items that appear throughout your site, over and over, like the body, h1-h6, p, tables, list starters, and basic links.
- Style sub items next, like table rows, link hover, etc.

- Style specific things last, like unnumbered list links that have a hover.
- Use CSS commenting between 'style areas' to tell yourself what the area is.

Basic styles: body, h1-h6, p
Table styles: table, tr, td, th
Basic links: a:link, a:visited, a:hover, and a:active
Layout items, like div layout styles

Special navigation formats, like rollovers.

Inheritance

CSS styles inherit values from parent styles, hence the *cascading* in the style sheet name. This means that if you designate the body text to be black, all text in your whole site will also be black. You can change that by assigning a different color to a specific selector, like gray text for a specific header:

```
body {color:#000;}
h3 {color:#ccc;}
h3.normal {font-weight: normal;}
```

This makes the header 3 gray instead of black. Then, the class .normal applied to header 3 makes only the <h3 class="normal"> turn to normal text rather than the default bold. The remaining header 3s will still be bold, and the non-bolded header 3 inherits the grey color from the h3 {color:#ccc;}.

So, when more than one style rules is applied to a web page element, how do you know which styles will be applied?

- Styles can be inherited from a parent. Some styles, like font family, text-alignment etc., are automatically inherited by child elements from their parent element (i.e. by an element contained inside another one).
- Later styles over-rule earlier styles. If you define a style property, and later define an alternative style property for the same thing, the later definition overrules the earlier one.
- Styles can combine from different sources. If the later styles or style sheets set any of the same properties as the previously-defined ones, they would overwrite the previous styles.

- Styles can apply to elements nested in a particular way. You can define styles to apply to particular kinds of elements nested within (or a child of) other particular kinds of elements. These would only affect the nested items, not other selectors of the same name elsewhere in the page.

Note: Inheritance can get to be a little too much fun and this makes CSS sometimes experimental in nature. In other words, you may find you have to do detective work to track down which style you create that is adversely affecting another style you are building. You'll learn a lot more by trial and error, but here are the basics:

- Inheritance: when all descending styles inherit the style of a parent, like table cells inheriting a align:left from the table selector
- Specificity: when you specifically tell an inherited style to do something different: table {align:left;} td {align:center;}
- Location: This rule breaks a tie when inheritance and specificity are duking it out. Rules that appear later have more weight (such as if you apply a style right inside your XHTML code.
- Override: You can also override the whole style system you create by declaring that a particular rule is more important than all the others: td {align:center !important;}
- Comment: Use CSS commenting to clearly note what you are doing, and why. This makes it easier for you to keep track of what styles might be the parent styles that a problem style is inheriting something from.

The Important Keyword

The important keyword makes a declaration take precedence over normal declarations, or those that are not styled with the important keyword. So "p {color: #000 ! important;}" takes precedence over "p {color: #ccc;}". The syntax for an important declaration is property: value ! important;

The relative importance of a rule also depends on its the source: whether it comes from a style sheet specified by the document author, the user or the user agent (browser). The order of declarations from least important to most important:

- User agent declarations
- User normal declarations
- Author normal declarations

- Author important declarations
- User important declarations

This keyword can only be used in style sheets and embedded header styles, but not in inline style rules.

CSS Positioning

CSS DIV layout can get really fun when you are trying to position things. We will discuss it more in the next chapter.

21
Page Layout Code – Part 2

While it's easy to layout a page with a table structure, this is no longer relevant for new sites and many CMS structures, especially as we evolve to HTML5 and CSS3 in the next couple of years. Cascading styles, with the ability to style positioning, sections, columns, and other layout information, replaces table layout with div (division) layout. For us to understand how this works, we need to look at the default model of the layout elements that browsers interpret.

The Box Model

The box model represents the default built-in margin, border, and padding possibilities of all elements - divs, images, paragraphs, links, etc. For instance, the default margins, borders and padding are all 0px, so when you use a element to "wrap" around other elements in your web page, there is no space between the divs edges and the elements being wrapped. Okay, layman's terms, please. . .

Think of an element of a web page, like some paragraph text . This paragraph element takes up space on the page, and part of this space includes default spacing around the text inside the paragraph tags. This space, plus the text itself, can be considered a "box", which is why it is considered a block element rather than an inline element.

The *content edge* surrounds the rectangle given by the width and height of the paragraph box, which often depends on the element's rendered content (the text inside the paragraph tags). The four content edges define the box's content box.

The *padding edge* surrounds the box padding. If the padding has 0 width, the padding edge is the same as the content edge. The four padding edges define the box's padding box.

The *border edge* surrounds the box's border. If the border has 0 width, the border edge is the same as the padding edge. The four border edges define the box's border box.

The *margin edge* surrounds the box margin. If the margin has 0 width, the margin edge is the same as the border edge. The four margin edges define the box's margin box.

Collapsing margins means that adjoining margins (no non-empty content, padding or bor-

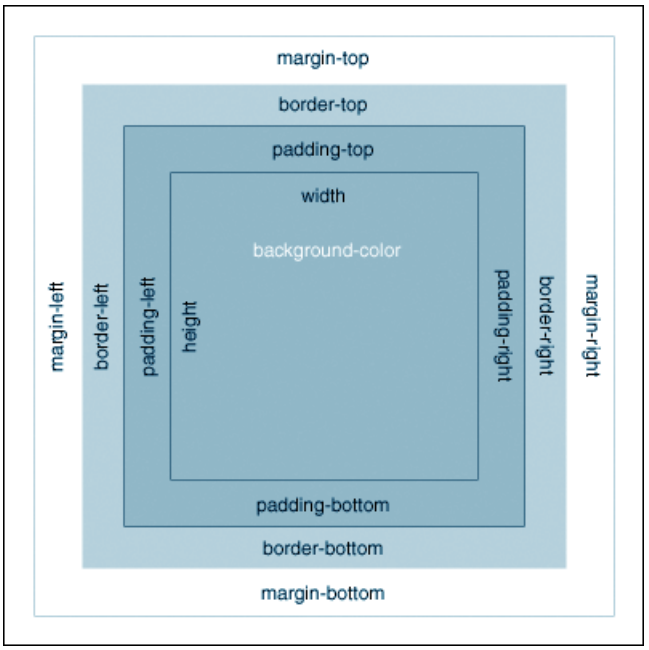
der areas or clearance separate them) of two or more boxes (which may be next to one another or nested) combine to form a single margin. There are margins that do not collapse:

- Horizontal margins never collapse,
- Vertical margins between a floated box and any other box
- Margins of absolutely positioned boxes
- Margins of inline-block elements
- An element that has had clearance applied to it never collapses its top margin with its parent block's bottom margin

Box Model Issues

It's crucial to understand the box model of web layout because different browsers actually have different default interpretations of the box model. Pre-Internet Explorer 6 and any Internet Explorer run in Quirks mode have issues. Internet 6 and later versions also seem to have some difficulties, because Microsoft developers interpreted the defaults of the box model differently than the W3C did.

According to the CSS1 specification, when a width or height is explicitly specified for any block-level element, it should determine only the width or height of the visible element, with the padding, borders, and margins applied afterward. However, Internet Explorer 5 includes the content, padding and borders within a specified width or height, which results in a narrower or shorter rendering of a box.



Box Model

In this example, the style rule would be:

```
div {width: 100px; padding: 10px; border: 5px solid black; margin: 10px; }
```

The top box is the standard - when this element is given a width of 100px, the width affects only the white portion, and the additional padding, border and margin affect the areas you see in gray.

Internet Explorer, however, interprets the width of 100px to be the "width" referred to the area within the outer edges of the black borders, but not including the margins. This has been corrected in IE 8, but the bad news is that web authors still need to design web pages that can be accurately viewed in earlier browsers, especially IE6 and 7.

As a result, you may find that pages which lay out properly in Firefox and IE 8 have troubles in IE 7 and earlier. How do you deal with this? Well, with much coffee, cursing, threats to the computer, and patience, you can do a couple of things.

- Create conditional style sheets. One would default to most browsers, and another would be specified for IE6 and older IE browsers, with style differences that affect the box model irregularities. This tends to be the simplest and most easy to maintain, plus minimizes side effects of CSS hacks and conditional code.
- Use conditional comments in HTML source code, which are conditional statements interpreted by Microsoft Internet Explorer. Conditional comments first appeared in Microsoft's Internet Explorer 5 browser and are supported through at least version 8.
- Use of CSS filters or hacks, which are a coding technique used to hide or show CSS markup depending on the browser, version number, or capabilities. Browsers have different interpretations of CSS behavior and different levels of support for the W3C standards. CSS filters are sometimes used to achieve consistent layout appearance in multiple browsers that do not have compatible rendering. Keep in mind that box model hacks have proven unreliable because they rely on bugs in browsers' CSS support that may have been fixed in later versions.

Since a web author's goal is to create and maintain as clean and pure of code as possible, you want to avoid CSS hacks and conditional code as much as possible.

DIV Page Layouts

The table structure for website layout offers much more control than simply designing a page from top to bottom (primeval HTML). However, tables have serious limits for posi-

tioning, as anyone who has tried to line up differently-sized elements with any control has experienced. As a result, CSS layout and positioning with div elements (divs), based on accurate box bodel understanding, allows the greatest control and flexibility for designing complex, attractive, and usable web pages.

The use of divs seems abstract at first, since unless you style them with borders they seem invisible. They also can have issues with "pushing off" a column that you style to be at the right but which appears below the other columns – until you understand the box model and how to style the "box" – or block-level – elements, while accounting for changes in their padding, margins, borders, etc.

This is why understanding table structured page layout first can be very helpful, especially if you use clear HTML commenting when opening and closing various table regions that represent your page segments. This commenting can help you replace the table tags with the appropriate replacements. Eventually you gain enough experience and or develop a natural spatial ability to "see" in div code, and the table starter becomes unnecessary. Note that:

- Div layout CSS styles can be added in the same external CSS style sheet as you do your element (text, table, image, and other) styling.
- Divs for page layout use **ID**, not classes: **#container**, not .container

Div Basics

Divs are essentially 'containers' for other page elements. For instance, your banner/identification segment of a web page would be a series of HTML elements you would wrap a around:

```
<div>Logo, website title, banner background color or image, website tagline, etc.</div>
```

Of course, if you simply use div elements to lay out your banner, navigation, content, and footer segments one after another, the web page will look unstructured. You need to set widths and positions for each of the divs so that the browser can display them as being above/below and/or next to other divs.

```
#banner {width:980px;}
<div id="banner"> Logo, website title, banner background color or image, website tagline, etc.</div>
```

You can also add background color and images to divs, just like table cells:

```
#banner {width:980px; background-image:url(retro.gif);}
<div id="banner"> Logo, website title, website tagline, etc.</div>
```

See how the background image code goes into the banner ID style, and therefore does not need to be added to any element inside the tags?

Div Page Structure

In HTML, the body is where all of the viewable elements go. The <body> acts as a web page structure boundary for what is viewable. It is also sufficient as a boundary to place a layout table inside.

However, with a div layout, you need to create a div container inside the <body>, before you structure the rest of the page divisions. Why? A container div is just a normal div, but you assign the properties which position the main 'bits' to this div. This div ID would be called container or wrapper – a wrapper for the whole page layout.

Inside this wrapper you would then assign a div for your banner segment, another for your navigation segment, etc. You need to code it into your HTML page before you add any other specific layout elements: container, then the banner, content, footer, etc.

```
<body>
<div class="container">
<div class="banner">Banner segment</div>
<div class="navigation">Navigation segment </div>
<div class="maincontent">Content segment </div>
<div class="footer">Footer segment </div>
</div>
</body>
```

The styles you assign to each of the segment div classes would determine their width, positioning, etc.

Basic div style classes for page layout would include, in this order

- body - standard body stuff
- #container (also called wrapper)
- #banner - your banner segment styling
- #navigation – your navigation segment styling

- #maincontent - your content segment styling
- #footer - your footer segment styling

Div Structure Variations

The div "maincontent" class of your website is commonly the non-banner and non-footer area. You can choose to put all navigation in the content area and not have a separate navigation div, or you can break up this "area" into columns:

- #container (also called wrapper)
- #banner - your banner segment styling
- #navigation – your navigation segment styling
- #centercontent - your content segment styling
- #rightside - your right-side sidebar segment styling
- #leftside - your left-side sidebar segment styling
- #footer - your footer segment styling

These are, of course, not all inclusive. You can assign div segments and style them in any way you see fit – your imagination and solid usability is the only limit for layout.

Div Elements

The div is not just for structuring a page layout – it can and often is used for positioning anything in a web page. For instance, inside your banner div, you might position your logo with a logo div.

Div Positioning

Normal Flow/Static Positioning

Normal flow is the default behavior of a web browser. You do not specify this in your style sheet since it is the default. With normal flow boxes will show up in the order that you placed them in your code, and each box level element is stacked on the next. Static positioning is applied by the declaration position: static. This places the element in the normal flow. Since normal flow is the default it is not normally necessary to explicitly use this. Where it is useful is over-riding another rule of lower specifity, such as:

```
div {position:absolute; }
#notAbs {position:static; }
```

Absolute Positioning

Absolute positioning positions a box relative to its containing block. However, unlike relative positioning the gap in the normal flow left by removing the element closes up. The containing block is the nearest ancestor with a position of absolute, relative or fixed.

You can use any one or combination of left, top, right, and bottom properties to position the box. The co-ordinates for absolute position have (0,0) at the top left of the containing block. Increasing the value of top moves the element down the page. Since absolutely positioned boxes are taken out of the normal flow they can be positioned anywhere on the page regardless of their position in the document's source order. This means its position, size, etc will not affect other elements on the page.

Relative Positioning

With relative positioning, the browser first lays out an element as though it was in the normal flow. The element is then displaced by the amount specified by the left or right properties and the top or bottom properties. A gap is left in the normal flow at the point the element should have appeared. Relative positioning does not allow an element to change size. If both left and right are specified, right will be ignored in languages which are written left to right such as English. Bottom is ignored if top is specified. For instance, you can take the original starting point of an image with the class of 'first' and push it 10px down and 10px to the right. This could cause it to overlap other elements.

Fixed Positioning

Fixed positioning is a subcategory of absolute positioning. The only difference is that for a fixed positioned box, the containing block is established by the browser window size. A fixed element does not move when a web page is scrolled as all other elements do. It is calculated in the same way as absolute positioning with respect to containing blocks in that it pulls the positioned box out of the normal flow. An example would be a navigation menu that is fixed at upper left position of the viewable browser window no matter how many scroll-downs the visitor does.

Floats

Elements can be made to float within the normal flow. Boxes are moved left or right as far as they can go. Elements after the float box will move up to fill any gap left behind thus flowing around the box with the float position. Float is not a position property, but it acts like one, and is applied with the float property, not the position property. You must

set the width property when floating block-level elements, or they will expand to fill the entire width of their container.

The way you get your div segments to display next to each other (like columns) rather than just below the one above it is to understand and use floats. When you float an element, like a div, it becomes a block box. This box can then be shifted to the left or right on the current line. The markup options are "float: left", "float: right" or "float: none".

You can put several floats **beside** each other, such as your

#navigation
#maincontent
#rightsidebar

Floated div areas will move to the left or right until their outer edge touches the containing block edge or the outer edge of another float. Always set a width on floated items (except if applied directly to an image - which has implicit width).

Elements in the code below a floated element will wrap around the floated element; they will appear to be right next to the previous div. You can avoid this by using the style rule c*lear:both* on the element that should be **below** the wrapping div. This tells the div to push the next element down.

```
#navigation {float: left; width: 150px;} /* This would be the left-hand navigation segment */
#maincontent {float: left; width: 400px;} /* This would be the main content text/graphics  segment of the page */
#sidebar {float: left; }/* This would be the right sidebar segment of the page*/
#footer {clear:both; /*forces the footer to float below the nav and content*/
```

Z-Index Layers

When elements overlap, you may want to adjust which elements are on top of other elements. This is also considered to be layering in CSS.

The z-index property specifies the stack order of an element. An element with greater stack order is always in front of an element with a lower stack order. However, z-index **only works** on positioned elements (position:absolute, position:relative, or position:fixed).

```
.putmehere { z-index: 1; } /* laid down first */
.layermeabove{ z-index: 2; } /* laid down on top of putmehere */
.putmeontop{ z-index: 99999;} /* laid on top of all */
```

Div Measurements

Div Widths

Back to the box model. Each block element, acting as a box, has default border, padding, margin, and content edges. These are 0, which means that each div you have next to another can look squashed, because it has no inherent padding or margin.

You can, of course, adjust this at will using your style rules:

#navigation {float: left; width:150px; padding-left:12px; padding-top:20px;} /* This would be the left-hand navigation segment */

Remember to add the width pixels to any width added by assigning padding, margin, and borders, because all of these measurements go into making up the full div element width. If you have three div segments alongside each other, their combined widths would in total need to be equal to the container div width you assigned for the width of the web page. If you calculate the individual div widths and their margin, padding, and border widths wrong, one or more of the div segments will get shoved down to the next line, below the other div segments instead of next to them.

Div Heights

The height of a div element is equal to the amount of content (text, images, etc.) you have in it. Setting a height does not work, any more than it does in a table cell. For instance, sometimes you place a background image inside a div element, and suddenly you see no height, and therefore cannot see the background image. This is because the background image has no real height, since it is in the background and not a block element inside the div

You can nest a second div inside the first one and add padding-top or padding-bottom around a blank character to 'push' the space down and reveal the background image.

Converting Table Layout to Div Layout

Okay, so you have a web page with a table defining the layout structure, and you want to move to a div layout structure. This isn't too hard, if you are exacting about the segments of your layout table and how they would relate to div elements.

First, you want to use HTML commenting to describe the segments of your table layout:

```
<!–OPEN PAGE LAYOUT STRUCTURE TABLE –>
<table>
<!–OPEN BANNER SEGMENT ROW –>
<tr>
<td colspan=”2”>Banner segment</td>
<tr>
<!–CLOSE BANNER SEGMENT ROW –>
</table>
<!–CLOSE PAGE LAYOUT STRUCTURE TABLE –>
Then, you can translate this into <div> elements:
<!–OPEN PAGE LAYOUT STRUCTURE TABLE –>
<div class=”container”>
<!–OPEN BANNER SEGMENT ROW –>
<div class=”banner”>
Banner segment
</div>
<!–CLOSE BANNER SEGMENT ROW –>
</div>
<!–CLOSE PAGE LAYOUT STRUCTURE TABLE –>
```

HTML5 Preparation

HTML5 is still not standardized in many ways, including the new structural tags that will replace several repetitively used Div classes. However, you can use the names of structural tags in your Div class naming conventions so that when you can reliably convert XHTML1.0 markup to HTML5 structure tags, you will have a minimum of effort.

<header>
The header tag is a container for a web page or an introductory information about a section within the web page. You can use for now.

<nav>
The nav element is for navigation elements. You can use .

<section>
The section is a thematic grouping of content, usually preceded by a header tag, and fol-

lowed by a footer tag. You can use div id=”section” for your content blocks, and clarify by section type: sectionmain, sectionside, etc.

<footer>
The footer element can be used as a footer segment of a web page. You can use .

This can and does work with much more complex page layout structure tables, but you get the idea.

What We Learned

Here is a basic div layout:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://www.w3.org/1999/xhtml">

<head>
<title>Fluid DIVs</title>
<meta http-equiv="content-type" content="text/html;charset=utf-8" />
<!-- Any images used in this template site are for learning purposes only. -->
</head>

<body>
<!-- Comment: Using CSS DIV Layout to organize a simple website.-->

<div id="container"> <!-- OPENS CONTAINER WRAPPING WHOLE SITE -->

    <div id="banner"><!-- OPENS SITE BANNER -->
    <h1>BANNER - left fluid layout </h1>
    </div>

    <div id="topnav"><!-- OPENS TOP NAVIGATION BAR -->
      <h2>TOP NAVIGATION BAR</h2>
    </div>

    <div id="leftnav"> <!-- OPENS NAVIGATION -->
    <h4>LEFT NAVIGATION </h4>
    </div>

    <div id="maintext"> <!-- OPENS MAIN TEXT SPACE FOR THE SITE -->
    <h2>CONTENT</h2>
    <p>Text</p>
    </div>

    <div id="rightsidebar"><!-- OPENS RIGHT SIDEBAR -->
    <h4>SIDEBAR</h4>
    <p>Text</p>
    </div>

    <div id="footer"> <!-- OPENS FOOTER SECTION OF SITE -->
    <p>FOOTER</p>
    </div>

</div> <!-- CLOSES CONTAINER -->

</body>
</html>
```

Basic div layout in HTML

22

Adding Dynamic Bits

You can add interesting bits on basic static web pages that can make them feel a bit more lively and provide modular information, like maps for directions. Often these items will not validate properly, so if you do use them, make sure to add a comment before the inserted code that doesn't validate so that you can recognize (and then ignore) the issue.

You also might have questions on other finishing touches, so I will list a few goodies here in the form of How Do I? questions/answers.

How do I:

Show the W3C Validator without the icon?

Place the Validator Icon code inside your page footer section, remove this code (), and replace it with the word Validate.

Make sure my site is ready for indexing?

- Use Google Webmaster Tools - you must have an existing Google/Gmail account to log in before you can add the website you want to check.
- You can also try the free tools on AddMe.com - meta tag generator, keyword density, broken link check.
- Will only work on a remotely loaded website (not your local computer or pen drive).
- Make sure to validate your live site too!

Offer plug-ins on my site?

By plug-ins, I mean adding the option for a visitor to download a program that will allow them to access something on your website. Just look up and provide these links (as needed) on your website to send the visitor to the download area, where s/he can download the program to install in his/her computer to view the related material you are publishing on your website.

- Acrobat Reader

- QuickTime
- Flash Player

Embed a Google Map?

It's easy, there are short simple samples of how to embed the map of your choice. You can either choose a map that you can manipulate (just like on the http://maps.google.com web page) or choose to use the Static Map.

Make sure to add the Google Maps API Key in your head section (a meta tag) for later troubleshooting.

Embed a Google Calendar?

Another easy item - you can embed a Google calendar once you (or your client) has a Gmail account and a Google calendar set up. There's an events publisher too.

Embed a Google-based external RSS feed?

It's easy too, and you can choose more than one feed source and from 3 different layouts.

Make sure to add the Google AJAX API Key in your head section (a meta tag) for later troubleshooting.

Add a Google Search engine to my website?

You can check with your web host (like DreamHost) to see if the host provides this option for free. The Google version is not free, but you may find a free or low cost search module to embed into your HTML code. These may be ad-sponsored, so double-check before you add one to your website and find you have ads popping up.

Add a social networking toolbar so folks can bookmark and share my site?

Just go to AddThis.com and choose your service (website) and format, then embed the

code into your web page as needed. “Help your visitors share your content everywhere. Generate traffic back to your site and increase your site’s search engine ranking.”

Embed a video/media?

YouTube has code on each video’s page that you can use to either link to the video on YouTube or to embed the video itself into your page.

You can also search online for an HTML generator for embedding different media.

Add a simple photo gallery?

You can add a simple gallery to your website without knowing much CSS or JavaScript. Lightbox 2 is one example, and Picasa is another.

Create a Favicon to show up in the tab of my website on the Internet?

- Go to favicon.cc to create a graphic
- Download when you are finished
- Put the favicon.ico file into your webserver directory - main directory where your index page is.

Explore other widgets to add to my website?

Visit Widgipedia for a variety of free open source widgets to embed/use. They are not all for beginners, so plan to have the time to explore, do trial-and-error, etc.

Sell items on my website?

If you really, really must have a basic purchase option on a website, you can use PayPal. Your client will have to set up a PayPal Account and do the financial heavy lifting before you can add any PayPal links/process to the website.

- DO NOT let yourself get pushed into doing the account yourself!!
- Make sure you have ample time to do this - at least 3-4 weeks for your first time

(including dealing with your client, your client setting up the account, setting up and testing the shopping cart and buttons, etc.).

- Go to PayPal for all the information on the database driven PayPal Shopping Cart.
- Check out the Website Payments Standard too. It is not a shopping cart.

Do Conditional Formatting for different IE and Firefox CSS style sheet files?

```
<head>
Title, metatags, AND then:
<!– STYLE SHEET IN APP_THEMES activating based on browser sniffed–>
<link rel=”style sheet” type=”text/css” href=”styles.css” />
<!–[if IE]> <link href=”iestyles.css” rel=”style sheet” type=”text/css”>
<![endif]–>
<!–[if !IE]>–> <link href=”styles.css” rel=”style sheet” type=”text/css”>
<!–<![endif]–>
</head>
```

Find simple things to add to my sites?

- DHTML Goodies
- Dynamic Drive

Get a fixed navigation when page is scrolled?

In your Cascading Styles, add the following attribute/value pairs inside your class for navigation:

```
{position:fixed; width:???; top:???; left:???;}
```

Add an alternate style sheet to my web page?

Add the following link to the alternate style sheet in your web page's <head> section:

```
<link media=”screen” rel=”alternate style sheet” title=”alternate style sheet”
type=”text/css” href=”altstyles.css” />
```

Add a style sheet for printing?

Add the following link to the print style sheet in your web page's <head> section:

```
<link media="print" rel="style sheet" type="text/css" href="printstyles.css" />
```

Add a style sheet for mobile devices?

Add the following link to the mobile devices style sheet in your web page's <head> section:

```
<link media="handheld" rel="style sheet" type="text/css"
href="mobilestyles.css" />
```

Create a Tag Cloud?

You can plug in your completed website at http://www.tagcloud-generator.com/, which will capture existing links. Then you can use the style options to set link colors, random link sizes, etc. **Note**: The tool builds inline, not CSS, styles. You can still get a lot out of it and create classes in CSS based on the inline styles.

What happens if I set a specific height or width on an element (like a for layout), and the content inside cannot fit?

This is called overflow, and you can learn more at the CSS Tricks - The CSS Overflow Property

Where can I learn about other CSS tricks?

You can go to a website called *CSS Tricks.* It covers all sorts of neat stuff, like CSS sprites, CSS transparency settings, and other CSS goodies. Keep in mind that CSS trick should be about mastering clean CSS with new and evolving ways of coding, nbot relying on old, out-of-date hacks from when CSS was new and browsers were. . . old.

Sitepoint also has 10 useful CSS tricks, such as dealing with minimum and maximum width, disappearing text or images in IE, a 3-d push button effect, and more.

Smashing Magazine has a great listing of many exciting CSS tips.

KEEP IT COMING!

Tools and resources for you

In this section, we've got a bunch of resources - glossary, troubleshooting tips, Q&A, JavaScript, and more.

23 HTML5 and CSS3

While XHTML with CSS is more robust than HTML alone is, it is evolving as we read. HTML5 seems to be the preferred leap for current web authors, and the W3C is focusing on standardizing it.

As of now, HTML5 and CSS3 are still in semi-development mode; the W3C Validator does validation for HTML 5, but the W3C CSS Validator is still mostly focused on CSS 2.1. In addition, not all browsers support all structural and semantic tags of HTML5, though this is changing. Internet Explorer 9 is expected to provide some support for them, and is now in beta mode. Other current version browsers (IE, Firefox, Opera, Safari, Chrome, etc.) only partly and/or experimentally support them. The consensus from various blogs and tutorial providers is that HTML5 will be widely standardized by/in 2012, and that many designers/developers are already integrating parts of it fairly consistently and pushing the envelope however they can.

These are also not likely to be recognized/used by most local employers for a minimum of 2-3 years (say 2013 or so), because employers are slow to change their IT systems, code, and products, which they have invested heavily in and usually repair and patch together until they explode. This generally means that while a designer might be ready, and employer might not use or plan to aim their web presence to the most current browsers. Therefore, HTML5 and CSS3 are still in the experimentation and leading-edge stage, where freelance designers and small studios are pushing the envelope and sharing tutorials while employers trail until they have become tried-and-true and can be supported easily with common tools.

HTML5

HTML5 has both a regular text/html serialization and an XML serialization, which is known as XHTML5. In addition to the markup language, the specification includes a number of application programming interfaces. The document object model is extended with APIs for editing, drag-and-drop, data storage and network communication.

The language is more compatible with HTML 4 and XHTML 1.x than XHTML 2.0, due to the decision to keep the existing HTML form elements and events model. It adds many

new elements not found in XHTML 1.x, however, such as section and aside. (The XHTML 1.2 equivalent (which XHTML5 replaces) of these structural elements would be and .)

Useful HTML5 resources include

- HTML5 Tags: http://www.w3schools.com/html5/html5_reference.asp
- HTML5 Rocks: http://www.html5rocks.com/

HTML5 Early Adoption

You can adopt certain stable parts of HTML 5 sooner, rather than later. These won’t change much as the standardization continues, and work with existing XHTML 1.0 information covered in this book and in current practice. You will want to search online for free tutorials and videos to show you techniques you can use. Benefits include:

- Cleaner code.
- Additional semantics of new elements like <header>, <nav>, and <time>.
- New form input types and attributes that will take the hassle out of scripting forms.
- Staying ahead of the curve before HTML5 becomes the mainstream markup language.

You can get started with HTML5 by:

- Simplifying your doctype.
- Simplifying your character set.
- Simplifying your <script> and <style> elements.
- Using block level linking.
- Get into the habit of using HTML5 class names in your HTML4 markup.
- Working in the new form input types, since they degrade gracefully.
- Using elements such as <time> to add semantics that will be interpreted by leading browsers.
- Using the <video> and <audio> elements, and then making them degrade gracefully.

HTML5 Doctype

Should you want to get started with very basic HTML5, note the new doctype and charset requirements:

```
<!doctype HTML> <!– new and simplified –>
<html> <!– simplified –>
<head>
<meta charset="utf-8" /> <!– new and simplified, must be right after <head>
opening tag –>
<!–Other meta tags –>
<title>Page Title</title>
</head>
<body>
</body>
</html>
```

HTML5 Semantic Tags

We did make note of several HTML5 semantic and general tags in Chapter 9. These were created for HTML5 only, and will not work properly in or validate for XHTML1.0 doctypes. However, once your adopt the HTML5 doctype, you can use the XHTML1.0 as usual and integrate these as well – insofar as browsers support them. Test well, my friends!

<dialog>…</dialog>
For marking up conversations — a chat transcript, an interview, a bit dialog from a screenplay, etc. Uses the definition list parts inside the dialog wrap.

<audio>…</audio>
Allows you to define defines sound, such as music or other audio streams. You can write text between the start and end tags to indicate that older browsers not support this tag.

<video>…</video>
Allows you to define video, such as a movie clip or other video streams. You can write text between the start and end tags to indicate that older browsers not support this tag.

<canvas>…</canvas>
Used to display graphics – but only allows you to create a "container" for the graphics. You must use a script to actually paint graphics inside of the canvas tagset.

<embed>…</embed>

Defines embedded content, such as a plug-in. Note: You can NOT write text between the start and end tags.

<source>…</source>
Specifies multiple media resources on media elements, such as <audio> and <video>. The browser should choose which file (if any) it has support for. You can write text between the start and end tags to indicate that older browsers not support this tag.

<figure>…</figure>
Make image embedding more descriptive and easier to recognize, and is also used for diagrams, code snippets and other visual elements. Used when referring to an image in the body of an article, not all images in general.

<time>…</time>
Defines a time or a date, or both, in a way that works with search engine spiders as well as human readers. The datetime attribute allows you to display a nice date format inside the tag.

<a>… link block
The anchor element was originally an inline element – you could link a image or bit of text inline. With HTML5 you can now link a block of text so that you can have a clickable heading and image all at once, rather than having to link each line separately. This removes duplication and creates a wider hit area to click.

HTML5 Structural Tags

We also took a brief look at HTML5 structural tags in Chapter 21. We considered these if you use the standard tag names for classifying your Divs in XHTML1.0, then you can more easily transition your web page structures to these same tags in HTML5. This allows you to replace some of a web page's standard div class names with actual HTML structure tags:

Note that when you apply styles to the new HTML5 tags, unknown tags have no default style in most browsers and are treated as inline elements. Because most of the new HTML 5 tags are structural, you want them be behave like block elements, so be sure that you include display:block; in your CSS styles for these elements.

<header>
The header tag is a container for a web page or an introductory information about a section within the web page. The <header> tag can act as your Banner segment with your typical logo/slogan that sits atop most pages. It can also act as a headline that introduces

a section of the web page. If you've been using in your pages, you can replace it with the new <header> tag.

<nav>
The nav element is for navigation elements. It can be used for your site's primary navigation, and again for your page-specific navigation, when you differentiate the two uses with CS IDs. If you've been using a tag to hold your page navigation, you can replace it with the new <nav> tag.

<section>
The section is a thematic grouping of content, usually preceded by a header tag, and followed by a footer tag. For instance, if you have been using div id="content", you can replace it with the new <section> tag. However, sections can also be nested inside of each other, if needed. You'll want to look at example HTML5 websites to see what works.

<article>
The article element should wrap a block of content that forms an independent part of a document or site, like a magazine or newspaper article, a blog entry. It is not a necessary part of a standard web page, but would be good for use in blog designs and online magazine sites.

<aside>
The aside element is for content that is related to the content that forms the main textual flow of a document. Inline footnotes, pull quotes (using the <q> inside the <aside> tags), or annotations are good examples. Note: A sidebar is a not good example – you would instead use another <section>, and give it a class="sidebar: for CS styling.

<footer>
The footer element can be used as a footer segment of a web page. It can, however, also be used as a footer piece inside an article (such as part of a blog entry).

Example use:

```
<!DOCTYPE html>
<html>
<head>
Chartype, title, metatags
</head>
<body>
<header><h1>Site Name</h1></header>
```

```
<nav>
<ul>
<li>Home</li>
<li>About</li>
<li>Contact</li>
</ul>
</nav>

<section id="main"><!–Giving an ID so you can use CSS to set position, etc.
–>
<h1>Content Area</h1>
<article>
<p>Main content items – text, images, links, etc.</p>
</article>
</section>

<section id="sidebar"><!– Giving an ID so you can use CSS to set position,
etc. –>
<h3>Sidebar Area</h3>
<article>
<p>Sidebar content</p>
</article>
</section>

<footer>
<p>Footer items – copyright, disclaimer link, sitemap link, etc.</p>
</footer>

</body>
</html>
```

Hacking for Internet Explorer

Sections are not fully supported by all browsers, and definitely not by older browsers like Internet Explorer 6 or Firefox 2. At this time, if you need to support legacy browsers like IE, there is a fix. IE 6 parses and displays the structural tags, but it won't apply any CSS to them. The fix is to use some JavaScript in the head section of your web page by using the createElement method so IE 6 can be 'informed' of the new tags.

```
<script> /*Remember that HTML5 already assumes the default script is
```

```
text/javascript */
document.createElement('header');
document.createElement('nav');
document.createElement('section');
document.createElement('article');
document.createElement('aside');
document.createElement('footer');
</script>
```

CSS 3

CSS1 and CSS2 does a lot to enhance XHTML, but it has limits which seem to require hacks to get around Internet Explorer and mobile device issues. It also has limits to styling, like the need to manually style rounded corners, an inability to add shading to text or objects, and other design needs that web authors use hacks to create.

CSS3 is the next stage of CSS, and the W3C is working to make new agreements and standards that will be adopted by browsers. There is some partial support by current browsers, but it is touch-and-go. You can get current information at CSS3.com, which can lead you to tutorials and techniques.

CSS3 Early Adoption

Like for HTML5, you will want to search online for free tutorials and videos to show you new, reliable and standardized techniques you can use. A few techniques are currently available: box and text shadows, transparency, border radius, custom web fonts, showing hyperlink cues, and text wrapping.

Anticipated items include a multi column module, a template layout module, a multiple background images feature, box model handling, replaced content, new selectors and pseudo classes, and more.

Keep in mind that although CSS3 hasn't been released as a final specification by W3C, it has some limited validation available through the W3C CSS Validator. You can try to compensate by for holes creating more style sheets to work with CSS3 as well as whatever you plan to create, but this might not be an effective use of your design time this early in the game. You are a creative – you decide! A great resource is: CSS3.info: http://www.css3.info/

What's Working?

Opacity

Opacity is a method of making elements transparent in CSS 3. It is not supported in all browsers. You can change the opacity of your elements so that they fade in and out. Here is an example of how we would add 80% opacity to a div using additional lines to support IE and Mozilla Firefox versions:

```
div {
opacity: 0.8;
filter: alpha(opacity=80);
-moz-opacity: 0.80;
}
```

Multi-Column Layout

You can create multi-column layouts without having to use multiple divs. The browser interprets the properties and create the columns, giving the text a newspaper-like flow. You can define the number of columns (column-count), the width of each column (column-width) and the gap between columns (column-gap). If column-count is not set, the browser accommodates as many columns that fit in the available width. You need to add lines to incorporate for Safari/Chrome (-webkit), for Mozilla Firefox (-moz), etc.

```
.index #content div {
-webkit-column-count : 4;
-webkit-column-gap : 15px;
-moz-column-count : 4;
-moz-column-gap : 15px;
}
```

To add a vertical separator between columns, use the column-rule property, which acts like a border property:

```
div {column-rule: 1px solid #00000;}
```

Word Wrap

The word-wrap property is used to prevent long words from overflowing. It can have one of two values: normal and break-word. The default normal value breaks words only at

allowed break points, like hyphens. If break-word is used, a word can be broken where needed to fit the given space and prevent overflowing.

Rounded Corners

Border-radius adds curved corners to HTML elements without background images. You need to add lines to incorporate for Safari /Chrome(-webkit), for Mozilla Firefox (-moz), and Konqueror (khtml).

```
H1 span {
color: #990000;
padding: .5em;
-webkit-border-radius: 5px;
-moz-border-radius: 5px;
-khtml-border-radius: 5px;
border-radius: 5px;
}
```

Box Shadow

The box-shadow property adds shadows to HTML elements without extra markup or background images.

```
#shadow {
-moz-box-shadow: 8px 8px 4px #777;
-webkit-box-shadow: 8px 8px 4px #777;
box-shadow: 8px 8px 4px #777;
}
```

Special Fonts

Most common fonts are used on the web because most computers have them preinstalled. With CSS3, you can use unique fonts as they can be called from an internet-enabled location such as the server your website is hosted. @font-face feature.

```
@font-face {
  font-family: Frutiger;
  src: url('http://example.com/fonts/ Frutiger.ttf');
}
```

```
h1 {
  font-family: Frutiger;
  font-size: 3em;
  letter-spacing: 1px;
  text-align: left;
}
```

Pending Items

A number of exciting CSS3 items just are not ready for primetime yet – browser support is too spotty or still nonexistent. Also, some of these might be dropped or delayed in favor of items not noted here. However, over 2011 and 2012, more of these will become standard as the browsers continue upgrading:

- Border images: defines an image to be used instead of the normal border of an element
- Column-space-distribution: describes how to distribute space between the columns if there's rest space left.
- Media Queries: You would be able to change stylesheets based on, for instance, the width and height of the viewport/device.
- Box shadow: drop shadows for boxes.
- Multiple backgrounds: allows for multiple background images on one element.
- 2D and 3DTransforms: allows you to rotate, scale or skew elements in a page to a specified angle or measurement.
- Transitions: lets you add an effect when changing from one style to another.
- Animations: lets you create simple animations with the @keyframes rule.
- User Interface: Lets you add new user interface features like resizing elements, box sizing, and outlining.
- Text-stroke, text-fill-color
- Appearance: allows you to make an element look like a standard user interface element.
- Selectors for element children, elements that have specific values, and elements in specific places.
- Color names: 130 more color names besides the core 17 (like black, white, red, etc).
- Grids: properties like columns and rows for grids.
- Linebox and Flexible boxes
- Hyperlink properties: target information for where/how new windows /tabs should be opened.

- Paged Media
- Generated Content
- Marquee
- Speech properties: audio stream styling support.
- Ruby properties: for Ruby use.

Adopting HTML5 / CSS3

We've looked at only a few parts of HTML5 and CSS3 that are in growing use. Should you use them in your design and production yet? That's a personal choice, and you need to balance the browser support, clean validation/markup, and your audience and client needs.

Clients

You need to establish if your client is ready for HTML5 and CSS3 use. For instance, many clients work through Content Management Systems that do not add these updates quickly, and your client might be one or more versions behind. As a result, using new standards can actually break existing working code and cost time, money, and audience support.

Visitors

The audience can get a lot of pros out of sites done with HTML5 and CSS3 integrations. They will have access to smoother transitions from one device to another, for instance. With proper structure and styling, websites can only get more visually appealing and well-organized so that viewability and accessibility improves.

Designers/Developers

You are a pivot in the use of HTML5. While myou have to understand that a website will not be able to look the same across various browsers, you will still have more pieces to play with to innovate and organize your work. You can add more presentation, more media, more structure, and as time moves on, fewer hacks because these standards are continuously improving.

Getting Started

Wherever you are in the continuum of your training, you can get started using reliable elements HTML5 by:

- Simplifying your doctype.
- Simplifying your character set.
- Simplifying your <script> and <style> elements.
- Using block level linking.
- Get into the habit of using HTML5 structural tag names names in your HTML4 markup class names.
- Working in the new form input types, since they degrade gracefully for browsers that don't yet support them.
- Using elements such as <time> to add semantics that will be interpreted by leading browsers.
- Using the <video> and <audio> elements, and then making them degrade gracefully for browsers that don't support them.

You can also play with the aspects of CSS3 that have code that supports different browsers, like:

- Rounded corners
- Opacity
- Open source fonts

Make sure that you really understand your client's needs and capabilities, and the demographics of the audience population - so you can predict which browsers they will mostly use. Know which browsers support what you want to do, so that you can avoid hacks and special code markup that won't be useful later. Then validate as much as you are able, review your work in the the current and last version of all major browsers.

24 JavaScript

JavaScript is a simple scripting language created for use in web browsers to make websites more dynamic. On its own, HTML is capable of putting out static pages. Once you load them the visitor's view doesn't change much until they click a link to go to a new page. Adding JavaScript to your code allows you to change how the document looks completely, from changing text, changing colors, the options available in a drop-down list, and more.

JavaScript is a client-side language, which means all the action occurs on the visitor's side of things. No trips to the web server are required for scripts to work. Usually they work because the page is refreshed, or you have programmed in times or responses to a visitor's actions.

You can easily add a bit of interaction to your website with bits of JavaScript. For the real thing, you will want to study JavaScript, get a book, and do tutorials on it. You will also want to look into JavaScript libraries you can lean on in your website like JQuery, MooTools, or Yui. However, you can easily do a few simple things without being an expert and add that little dash to your website that can make an impact.

Let's just look at a few examples of simple javascript you can apply to make your site a bit more dynamic. You can find tons of javascript tutorials and scripts on the Web.

Add A Dynamic Date

```
<head>
<script type="text/javascript">
var myDate = new Date();
var myYear = myDate.getFullYear();
</script>
</head>
<body>
Use somewhere, like in the footer.
&copy;<script type="text/javascript">document.write(myYear);</script>
</body>
```

Add A Protected Email Address

```
<head>
<script type="text/javascript">
function getEmailAddy(displayName, emailName, emailHostName, ext)
{
return "<a href='" + "mail" + "to:" + emailName + "@" + emailHostName + "."
+ ext + "'>" + displayName + "</a>";
}
</script>
</head>
<body>
<p><script type="text/javascript">document.write( getEmailAddy("Email
L.J.", "lbothe01", "seattlecentral", "edu") );</script></p>
</body>
```

Add a Changing Phrase

```
<body>

<script type="text/javascript">

var Quotation=new Array()

Quotation[0] = "I want a mocha.";

Quotation[1] = "I want some sleep.";

Quotation[2] = "I want to be in the sun.";

var Q = Quotation.length;

var whichQuotation=Math.round(Math.random()*(Q-1));

function showQuotation(){document.write(Quotation[whichQuotation]);}

showQuotation();

</script>

</body>
```

Create a Rollover Image

```
<head>
<script type="text/javascript">
<!–
if (document.images) {
    button1 = new Image
    button2 = new Image
    button1.src = 'img1.gif'
    button2.src = 'img2.gif'
 }
//–>
</script>
</head>
<body>
<a href="http://domainname.com" onmouseover="document.rollover.
src=button2.src" onmouseout="document.rollover.src=button1.src"><img
src="img1.gif" border=0 name="rollover"></a>
</body>
```

Open a New Window

```
<body>
<script type="text/javascript">
<!–
window.open ('titlepage.html', 'newwindow', config='height=100,
width=400, toolbar=no, menubar=no, scrollbars=no, resizable=no,
location=no, directories=no, status=no')
–>
</script>
</body>
```

Print This Page

```
<body>
<a href="javascript:window.print()">Click to Print This Page</a>
</body>
```

OR

```
<body>
<form>
<input type="button" onClick="window.print()">
</ form >
</body>
```

Redirect Page

```
<head>
<script type="text/javascript">
<!–
var time = null
function move() {
window.location = 'http://www.yourdomain.com'
}
//–>
</script>
</head>
<body onload="timer=setTimeout('move()',2000)">
</body>
```

Back to Previous Page

```
<body>
<a href="javascript: history.go(-1)">Back</a>
</body>
```

Hide Javascript From Browsers

Older browsers that do not support javascript will tend to show the code in the visitor's view of your site. You can comment out your script so this does not happen.

```
<script language="JavaScript">
<!–
Put JS code here. Old browsers will treat it as an HTML comment.
–>
</script>
```

25

Building Web Authoring Skills

A career in the web specialties demands continuous training and skills upgrades. If you love to learn, this career is definitely a keeper. It is not inexpensive to attend professional trainings and conferences. However, you have many free and low-cost ways to keep boosting your skills.

Classes and Tutorials

- Community colleges often have short refresher courses you can take on a weekend or a few weekdays in continuing education.
- For Adobe program free trainings, check out Adobe Creative Suite Video Tutorials, especially since you can pick up techniques while viewing the use of the programs.
- Microsoft also has extensive video tutorial offerings for users and developers.
- Keep an eye on the W3C Schools for continuing information on HTML5 and CSS3.
- Search online for free tutorials in your area of interest, and you'll find plenty from experts and peers in the field.
- Check out Lynda Online Training - $25/month for unlimited tutorials.
- Also look at creativeLIVE! - often has free live online courses and inexpensive videos of past ones.
- Online design magazine sites, like Speckyboy, SmartWebby, A List Apart, Smashing Magazine, Web Designer Wall, and others also provide articles with links to outstanding tutorials and resources.

Get Experience

There is no replacement for getting current experience. Even with a tough job market, you have options to explore to keep adding work to your portfolio and expertise under your belt.

- Look for paid and unpaid internships which give you room to grow.
- Do probono work with nonprofits listed through VolunteerSolutions, VolunteerMatch, Idealist.org, Serve.gov.
- Help your local community center, senior center, or other neighborhood small

businesses and social groups with pro-bono or discounted web assistance. Treat them professionally to learn project management skills too.
- Check out local contract agencies for creatives to get on their list for contract jobs.
- Check out bit work at online freelancer market websites. Do keep in mind that these markets are international, and pricing for your work will be very low, but you can still get bits and pieces of experience this way if you want to try.

Meet Your Peers

The great thing about the web specialties is that you are in a most superb group of creative and chatty people. You can always find folks in your field to network with, share tips and ideas, and find out about gigs and job opportunities.

- Check out Meetup.com for local web career meetups.
- Find out if Biznik has local seminars and meetings you can attend (or lead).
- Find out when Adobe, Microsoft, and other software companies have local low-cost events and user group meetings you can attend.
- Search online for Unconferences in your area, which are usually low-cost and have a fairly unstructured but fun layout where you can meet and swap info. Look for BarCamp, Social Media Camp, and InfoCamp.
- Find out if AIGA, SPGA, and the Freelancer's Union have local chapters.
- Check out your locan creative/tech agencies to see if they host events, webcasts, and/or have e-newsletters listing events and trainings.

Build, Build, Build

Even if you can't find paid or pro bono work that suits you right away, you can and should build portfolio items for yourself. This is a great way to experiment with web specialist tools like Wordpress, Drupal, Joomla, HTML5, CSS3, and free/low-cost products and tutorials you find.

Multiple Templates

For instance, now that you know about templating, you might look at several web page designs that you like, and put together your own starter templates for them so you have a place to start when you begin a build.

You can include several basic HTML templates, and also create two or three CSS templates where you lay out your style coding the way you like, then modify, add, and subtract style rules for real projects.

Expand Portfolio and Self-Promotion

When you get started, your portfolio will be lean, often a small website with a few examples of your work. Keep in mind that your portfolio itself, and your own logo/logotype, are part of your professional "foot forward". The good news is that you can enhance your portfolio with small touches that expand your experience and reach.

- Link to your online profiles. If you are on LinkedIn, have a professional Facebook account, and/or belong to career-related organizations, make sure to link to these from your portfolio. If you are not on these, get on LinkedIn!!! Hustle!
- Create your professional profile on Google and link to it as well.
- Add your portfolio link in the footer of your email messages and in your mobile business card. Don't overdo it - just list a primary.
- Consider contributing to a good blog in your field of expertise, and linking to it on your portfolio.
- If you have time, drive, and enough good, solid tips and information to share, you can create your own blog for professional promotion.
- Get business cards and create mobile device business cards.

appendix 1

Glossary of Terms and Phrases

Here are some basic phrases, terms, and acronyms you will encounter in this book and your web authoring studies. This glossary is not exhaustive, though it looks like it.

Above the fold: The part of a webpage that is visible in the Web browser window when the page first loads is described as being 'above-the-fold'.

Accessibility: Design of websites that focuses on being accessible to people with different kinds of disabilities and the difficulty they can experience due to physical and or technological barriers.

Alt text: The alternative text (alt) attribute provides a text-only description of an image on a web page.

Anti-alias: Aliasing is the distortion of a continuous line due to the nature of screen display, which relies on a matrix of pixels. Anti-aliasing visually corrects this by introducing additional colored pixels to give the impression of a continuous line or curve.

Authoring: The process of designing and, at a technical level, building a website.

Back-end: The information structure or application part of web authoring – programming, database admin, etc.

Bandwidth: The amount of data that can be transferred over the network in a fixed amount of time, such as bits per second (bps). A hosting server will allocate your website a fixed amount of bandwidth usage within a regular period of time.

Blog (web log): A website comprised of date-related entries (or posts).

Browser: A web browser is an interface used to access the World Wide Web and to view web pages. Examples include Internet Explorer, Firefox, Chrome, Safari, and others.

Character Encoding: Unicode and its parallel standard, the ISO/IEC 10646 Universal Character Set together constitute a modern, unified character encoding (per Wikipedia). This is to establish a universal set of characters that can be encoded in a variety of ways, like readable HTML generated web pages.

Comps: In design, a comprehensive layout (usually shortened to comp), is the page layout of a proposed design as initially presented by the designer to a client. It shows the relative positions of text and illustrations as a rough draft of the final layout.

Content: The 'stuff' that makes up a web site – text, images, links, lists, navigation, media, etc.

Content Management System (CMS): A number of templates corresponding to standard 'types' of information and web pages in a website that are designed so that several users can update the pages though a simple web-browser-based interface.

Cascading Styles (CS): CS is a style language used to describe the presentation semantics (the look and formatting) of a document written in a markup language – HTML, XHTML, and XML. It is designed primarily to enable the separation of document content (written in HTML or a similar markup language) from document presentation, including elements such as the layout, colors, and fonts.

Cascading Style Sheet (CSS): A file with standardized style rules that add style (e.g. fonts, colors, spacing) to Web pages.

Database: Store and organize data for later retrieval, often used for managing accounts and user information in dynamic websites.

Deprecated: Code that has been superseded by the W3C by more efficient or accessible alternatives.

Domain Name System (DNS): Identifies each computer as a network point on the Internet using an internet protocol address systems to translate from domain name to IP and reverse.

Doctype: Determines a layout model used in a web page and lets a browser correctly interpret the coded tags.

Dynamic HTML (dHTML): Dynamic HTML is an extension to the standard HTML language introduced by version 4+ browsers (Netscape Navigator and Internet Explorer). "Dynamic" makes reference to changing the webpage display in response to user interaction.

Elements: When writing the code that creates a webpage (authoring), content (text, images, etc.), is marked-up into elements. Element types include: headings, tables, images and lists.

Fonts, system: Along with the typefaces installed with a browser, the typefaces used by a computer operating system (system fonts) are often considered web-safe.

Fonts, web: A web font is a typeface installed with a web browser, and typically (although not always) designed for screen-use. The term is also used to describe typefaces that can be used by designers when authoring web pages.

Forum: A discussion board on the Internet.

Frames: An HTML technique for combining two or more separate HTML documents within a single web browser screen. A web site using frames often causes great problems for search engines, and may not be spidered and indexed correctly.

Front-end: The interface part of a web page that allows a user to access the information or application. It is focused on design and usability.

File Transfer Protocol (FTP): The transfer of files from one computer to another by logging into the other computer through an interface program like Filezilla.

Graphical User Interface (GUI): Allows a user to interact with a computer without entering code. With a combination of an input device (such as a mouse or stylus) and visual representations of the workspace and tasks, a user is able to interact with the computer in a manner similar to the physical manipulations available in the real world.

Hex Colors: The websafe color palette consists of 216 colors that when displayed on a 256-colour (8 bit) monitor will be displayed as continuous flat colors. These colors are rendered as flat colors as to hue and saturation. Hex colors have 6 letter/number combinations to identify the color.

Home Page: The opening page, start page or main page of a website, which is technically be the index/default page of the website's root directory.

Host: A computer (or a network of servers) that stores the files of a website which has web server software running on it, connected to the Internet.

HyperText Markup Language (HTML): The predominant markup language for web pages. It is written in the form of HTML elements consisting of "tags" surrounded by angle brackets within the web page content, and is the building block language of all basic websites.

HTML Tags: Most elements in HTML coding are written with a start tag and an end tag, with the content in between. A rare few close themselves because they do not wrap around content - like a horizontal rule, an image, a meta tag, and a break tag.

Hyperlink: A link from one Web page to another, either on the same site or somewhere else on the Internet.

i-frame: An i-frame is an inline frame, and is treated as though it were an image or table. As the content page acts as its own frameset, the problem of bookmarking a specific navigation/content page is overcome.

Image Map: A graphic on a Web page that is divided into parts which link to different Web pages.

Importing: The ability to bring in assets, code, information, and/or artwork created in other applications and use them in a current program.

Interactive: A web page is interactive when it prompts a response from the user or in some way can interact with the user dynamically, like filling out a form or logging into an account.

Internet: The Internet is a global network connecting millions of computers. Each Internet computer, called a host, is independent. The Internet is not synonymous with World Wide Web. The Internet and the Web are two separate but related things.

Intranet: An intranet is an internal networked communication tool often accessed through a web browser.

IP Address: An IP (Internet Protocol) address is a unique identifier for a point or host connection on an IP network.

JavaScript: An object-based, client−side scripting language developed by Netscape. When embedded in the head section of a web page, It can produce interactivity to a web page.

Link, absolute: An absolute link is created when the href value is a fully qualified URL, such as http://www.studiobast.com.

Link, hyper: a link from one Web page to another, either on the same site or somewhere else on the Internet.

Link, relative: A relative link is created when the destination href value is relative to the location of the current web page or source anchor.

Link, root-relative: A root-relative link is created when the href value is relative to the website root.

Link rot: Link rot occurs when website navigation is compromised by broken hyperlinks.

Lossless: Compression of images discards (loses) some the data in order to compress the image, which can result in degradation / pixilation of images. Jpgs are lossless files.

Lossy: Compression for images compresses by reinterpreting the mathematically interpreted image as accurately as possible, with limited/no degradation or pixilation. Pngs and Gifs are lossy files.

Markup: Markup comprises a set of symbols, or a language, that can be used to provide instructions. The use of markup is supported by rules that define the symbols and how they should be used: a specification. A webpage is created by marking-up content (text, images, etc.) using the HyperText Markup Language (HTML).

Metadata: Information-about-information, such as classification by subject, format, author, etc.

Meta Tags: HTML page head section tags that are loosely defined as data about data. These give browsers and search engines information so that web pages can be found and indexed properly.

Navigation: Typically menus and a search tool that people can use to navigate around a website.

Navigation, top-level/global: 'Global', 'top-level', 'persistent', or 'primary' navigation is navigation that is displayed on every webpage, usually at the top or left of the screen. This type of navigation provides access to the main content sections or services.

Navigation, in page: When arriving at longer-length web page, a visitor may need to scroll down to get a sense of the topics covered. In-page navigation simplifies this orientation process by linking to major content headings from the top of a web page.

Navigation, footer: Footer navigation is usually displayed on every webpage and as such is global navigation. However, rather than linking to content sections, footer navigation typically links to administrative content; copyright statements, terms of use, legal disclaimers, website feedback links, etc. Footer navigation also tends to be presented as text links.

Open Source: A computer program where the underlying code is made available to the public is referred to as 'open source'. With the networking of computers enabled by the internet, what computer programs 'are doing' has become a cause for concern. Open source programs address this by making the program code available for general inspection. A significant byproduct of this transparency is the ability for programmers to actively contribute to the continued development of a shared code library.

Pixel: Refers to how monitors divide the display screen into thousands or millions of individual dots to display an image. A pixel is one dot.

Plug-in: A plug-in is third-party software that extends the capabilities of a computer program. In a web browser, a plug-in may be required to display non-standard (often multimedia) content formats. Common browser plug-ins include Flash, QuickTime and Shockwave.

Quirks Mode: In quirks mode, a web browser attempts to render code based on a 'best-guess', this includes a generous interpretation of code that may be non-standard or poorly-formed from browser to browser.

Ranking: The number that a web site is listed for a specific search term in a specific search engine. Search Engines utilize a ranking algorithm (mathematical formulas, variables, and set of weights) to determine a site's ranking for a particular keyword or keyword phrase.

Raster: Graphics are made up of a grid of pixels, or points of color, viewable via a monitor, paper, or other display medium. File formats include the lossy format tif, jpg, and bmp, which lose quality each time the image is resaved.

Resolution: How fine the dots are that make up an image. The more dots, the higher the resolution. When displayed on a monitor, the dots are called pixels. A 640 x 480 screen (resolution) is capable of displaying 640 distinct dots on each of its 480 lines, or about 300,000 pixels.

Rich Internet Applications (RIAs): Web applications that have many of the characteris-

tics of desktop applications, but are delivered by a site-specific browser, a browser plug-in, or independently via sandboxes or virtual machines.

Scripts: An executable list of commands created by a scripting language. Scripts that are executed on a web server are server−side scripts. Scripts that execute on a website visitor's own computer are client−side scripts. Scripts can be embedded within HTML to produce a web page with dynamic actions.

Search Engine: A web interface that lets a visitor search for information on the World Wide Web. Examples include Google, Bing, Yahoo, and others.

Search Engine Optimization (SEO): The process of improving ranking in search engine results.

Server: A computer connected to the Internet that stores and provides information. Web pages are stored on a server, which is a storage tool of a web hosting service.

Site: A particular 'place' on the Internet - a collection of Web pages.

Sitemap, design: A visual tool which displays the pages which will be designed for a website. Can be displayed in a flow-chart fashion, with main levels, sub levels, auxiliary levels, etc.

Sitemap, web: A sitemap provides an overview of website content in a manner similar to the contents page of a book. Sections and pages are typically listed according to narrative flow, if the author intends for pages to be read in a specific order; or alphabetically or by chronology if content pages are essentially unrelated. The later is the case in news websites, where articles are often archived by topic, and then by title or publication date.

Spider: An automated software robot that continuously crawls hyperlinks and pages on the Internet and collects data that is returned to its database for indexing.

Splash Page: A splash page is an introduction to a website, placed before the homepage, and are often animated and introduce the products, services or mission of the website.

Standards-compliant Mode: A web browser assumes that a web page has been authored to the Web content specification declared in the page's head section; code that does not conform to the declared standard may not display, or may display incorrectly.

Style Rules: Coding language used to describe the look and formatting of web pages in a more versatile manner than HTML coding is able to do.

Style sheet: CSS rules are added to a web page either by writing the code directly into the <head> of the webpage HTML, or by linking to a separate file. A separate file containing only CSS rules is commonly referred to as a 'style sheet', and has the extension .css.

Target: The browser window output of a link.

Template: Skeletal HTML pages with the main content left out in order to provide an effective solution in creating many pages with an identical look or navigational structure but with different content.

Thread: A group of messages or postings to an Internet forum on a single topic.

Traffic: Traffic on the Web is a measurement of the number of users who visit a Web site.

Upload: Uploading is the process of copying files from your computer to a remote location, such as to a webhost server.

Uniform Resource Identifier (URI): The generic term for all types of names and addresses that refer to objects on the World Wide Web. A URL is one kind of URI.

Uniform Resource Locator (URL): An Internet 'address' which is made up of two parts - the protocol (e.g. HTTP) followed by two slashes, and the domain name. This gives the exact location of a particular file on the Internet.

Usability: Refers to the level or degree of a web page's operating friendliness for the user. Usability is a measure of the quality of a person's experience in interacting with content or services.

USB Flash Drive: Memory data storage space which you can plug into a USB port on a computer. USB flash drives are usually removable and rewritable.

Validation: The process of checking data against a standard or requirement. Web page validation involves checking the code used to create a webpage (markup), against a computer-language specification.

Vector: Graphics which use geometrical primitives like points, lines, curves, and shapes or polygon(s) which are all based on mathematical equations. File formats include .ai, .gif, .eps. Uses lossless compression - generally does not lose quality each time the image is resaved.

World Wide Web Consortium (W3C): is the main international standards organization for the World Wide Web. The W3C determines the evolving standards for workable HTML coding and web page styling.

W3C Validator: A validation tool that allows web coders to check HTML documents for conformance to HTML or XHTML standards and for coding errors.

Web: Abbreviation for the World Wide Web (WWW).

Web Application: Web or real programs designed to be used on the Web site through a browser, such as e-commerce tools, games, polls, logins, etc.

Web Code Structure: The order and format of the HTML code used to make up a web page before it can be viewed on a browser like Internet Explorer. Includes HTML tags, the head section, and the body section.

Web Host: Web hosts are companies that provide space on a server they own or lease for use by their clients as well as providing Internet connectivity. This space is where one stores website pages so that visitors can access them by typing their address (URL).

Web Page Structure (basic): The structure of the viewable website that visitors can see. It includes an identifying banner area at the top, navigation near the top and/or one of the sides of the page, and the content area, which has the text, images, links, tables, footer, and other areas that make up the information of the web page.

Wireframe: A diagrammatic representation of the elements that need to be accounted for on each page of the website or screen of the program. Usually created in plain B&W before the design comps.

Workspace: The view of toolbars, panels, windows, etc. that male up the work area in a program used for creative design, production, illustration, coding, etc. In a personally owned copy of a program, you can personalize and save a copy of the workspace.

WYSIWYG (what you see is what you get): A graphical user interface (GUI) where the user appears to interact with an accurate, screen-based representation of an end-product. This form of software interface was pioneered by computer programs used for creating (print) documents, specifically newspapers.

Extensible Hypertext Markup Language (XHTML): An XML coding language that extends versions of HTML. It is more standardized than HTML so that browsers like Internet Explorer and Firefox can consistently interpret the coding correctly.

appendix 2
Problem-Solving Tips

My Code Doesn't Work!

When you are learning to code, you'll have a lot of situations where your code just doesn't work. Things look wrong, or don't validate, or look significantly different on different browsers.

Most of the time it's because of simple things that break your code's ability to work, especially depending on which doctype you chose. Here are a few things to keep an eye out for:

Placement

- Your web page must have open and closed html tags surrounding the head and body.
- Your head tags must open and close before you open the body tags: <head></head><body></body>
- NO tables, text tags, styles, etc. ever go inside the head section of your website.
- All your text, image, link, and styling tags need to be in the body of your web page

Validation Hint: If your page has several dozen errors, you likely have a problem with your doctype, opening html tag, and/or metatags.

Tags

- Make sure you have open and closing tags for tags that wrap around something: Ex. text here
- Make sure your closing tag has a slash in it: Ex.
- Make sure you open and close the same tags:
- NO <h1></h2>
- YES <h2></h2>
- Check that you did stand-alone tags right:
, <hr />, <img />

Validation hint: If you get an error mentioning some part of the , <td>, and like tags, look at both the line the error has hit on AND the lines above it. Often you will find you forgot to close another tag set, like ending a table row before starting a new table row.

Nesting

You need to nest tags properly:

NO:
YES:

> **Consider: <table><tr><td> text </td></tr></table>**

Validation hint: You might get a message stating that some tag or element 'can't be used in this location". Use your cursor to highlight your tag pairs to make sure everything nests properly.

Mistyping

- Close all quote marks you open: Ex.
- Check for missing carets: Ex. NO: </p NO: <!--comment--
- Use the = sign when needed: Ex. NO: width-"20%" YES: width="20%"
- Check coding spelling: NO: img scr
- If you see odd symbols showing up in your browser, you may have used a keyboard key to make a symbol, such as the &. You will need to instead use the HTML character code for that symbol.
- Use hex colors properly, with the # sign before the color code. Ex. NO: color:ccc; YES: color:#ccc;

Validation hint: You can get all sorts of weird little warnings, but check the noted line, and the line above and below, very carefully for mistyped tags, styles, carats, punctuation, etc.

Images

- Put the correct location of the image in your code
- Verify the image you need is actually in the location you code
- Type the correct image extension - .jpg, .gif
- Check to see that you have a space before you close your img tag: Ex.
- Make sure you added the alt tag and the title tag.

Validation hint: A main image error is that you have not added the mandatory alt tag. Always add both the alt and title information in the image for accessibility purposes.

Links

- Be sure to spell and space the link right: Ex. NO: <ahref="">, YES:
- Make sure to wrap your linking info around the item to be clicked: linktext,
- Make sure to always have the title tag in your link tag: Ex.

File Not Getting Found?

- Absolute urls - full length, used for external links, redirect pages
- Relative urls – the link name is relative to file location
- Moving down a level (foldername/filename)
- Moving up one level (../filename
- Moving up a level to another folder in that level (../foldername/filename)
- Moving up multiple levels (../../filename)

Validation hint: Usually any error with your link will be because you forgot to close it with the tag, or you did not nest the closing tag in the right place.

Styles

- Can only be used inside of a style="" attribute for another tag, such as .
- ALL rules need to have the attribute/value paired style shown after style=" and must close with another " (quote marks).
- General use and format: <tag style="ruleattribute1: value; ruleattribute2: value;"> </tag>
- All measurements need a qualifier - px, %, etc.
- Inside the style quote marks, do not use the equal = sign - use the colon : instead.
- Every attribute/value pair needs a semi colon ; to 'finish' it.
- Remember there are no more *font* colors. Use color:color;

Validation hint: Errors will usually be mistypes, using the colon instead of the semi-colon at the end of a value rule, using the = sign instead of the : inside the style="" space, and using deprecated code like font color.

Validation hint #2: Always use the inline style rules instead of old deprecated tags, since your validator will find tons of errors, like font face=, align=, valign=, font color=,

Seeing Code Changes

Sometimes you do the work and make code changes, and you can't see any difference.

- Save your file after changing code.
- Refresh your browser when testing changes to your code.
- If your browser seems "stuck", try closing it and opening it or another browser after 30 seconds.
- Make sure that you check any pages you upload to your web host using the full website url, so you can make sure it transferred properly.

Understanding the W3C Validator Error Messages

Unfortunately, the W3C validator messages are written for code developers, not normal designers trying to make sure their web pages will work properly for all browsers. Here are a couple tricks of the trade, plus I listed some Validation Hints above.

Don't panic.

- Focus on correcting only the first error or two in your original code, then revalidating the full code of the page you changed. Often you'll solve several errors that appear at one time when the first ones are corrected.
- Look for the line number the error is referring to and find it in your Text Editor.
- If you see several references to one specific line number, carefully go over that line to find out what is wrong: nesting, mistypes, missing characters, etc. This will usually solve all the repeat errors for that line in the Validator/
- Look at the error note for anything that looks familiar, and for any bit of code highlighted in red. Usually, this means the error has hit when the validator passed over this highlighted bit of code, which means something in the line, or within the lines immediately above it, is incorrect.
- If you really aren't getting the point of the error message, look over the referenced line in your text editor, and check the code coloring carefully to see if it is off (like all green, tags not being blue but purple, etc.) The incorrect color can also help you discover what is wrong.
- If you see errors stating 'deprecated' and/or with things like align=, valign=, font color=, etc., you'll know you are not using the inline style rules and that you should be.

appendix 3
How Do I Do That?

Of course you want extras. Websites would be boring without them. You can find many, many free resources on HTML, CSS, JavaScript, other scripting, media and other modules, images, and more online. Just choose a search engine and go for it.

I'll share a few of the favorites my students have requested. How do I. . .

Decide how wide my website should be?

The common resolution for computer monitors these days is 1024x768. This means you can:

- Use a flexible width table of 100%
- Set a fixed-width table of between 900-1000px

Create text wrap around an image?

 You can also choose float:right

Create a middle of the screen website with equal background on both sides (like Wordpress blogs)?

This is a two step process:

body {text-align:center;} Works for IE only, so you also need to relist it with a variation: body {text-align:-moz-center;}
<table style="text-align:left;"> You can also add a background color or non-repeating image (like a banner)

Get my banner or other background image to show up because only part of it does?

The background image in a table and td tag only shows up if there is other content inside the tags to push the cell "open". If you don't plan to put text in there, you can:

Add style="padding-top:____px;" or
Add a few
 or   tags to push down a bit.

Keep my text or images from being squashed against the side of a table cell?

You can use style="padding-left:____;" and/or style="padding-right:____;"

Keep one of my table columns (usually the right-hand one) from dropping down below the rest of the table?

Ah. . . if you have chosen a fixed-width table and calculated the widths of the columns, and have added margin measurements, you have actually added extra width on top of your calculated columns. Remember that your column widths and margin additions must together equal the width of the fixed width table.

Get rid of the spacing around my table cells so my borders fit seamlessly?

You need to do this inside the opening table tag by using the border-collapse option: <table style=" border-collapse:collapse;">

Note: your table can still have a border - the border-collapse simple affects the build-in spacing around table cells (<td></td>).

Change the font family and/or size through the whole web page?

You can do this in the body tag: <body style="font-family:_____; font-size: _____;">

Change the color, size, and/or font of just a small piece of text after I change the whole body text?

Easy as pie! Place a around the bit of text, with *yourchoice* being the font, color, size, and other changes.

Put the W3C Validator code in the footer of my web page without the ugly icon?

Piece o'cake! Place the Validator Icon code inside your page footer section, remove this code (), and replace it with the word Validate. Since it all falls inside a , you can even change that tag to be h5, h6, or style the to make the text more discreet.

appendix 4

Steps to Build a Web Page

Okay, you've read, marked up, skimmed, tried out, and thrown this book at your cat. While it should all fit seamlessly together for you, isn't there just a simplified set of steps to authoring a web page? Sure! Keep in mind you still need to plan your workflow, organization, timeline, and launch-related tasks. However, there are basic steps to just diving in and building a web page from scratch.

Step 1

Get your coding tools open and ready: text editor and browser to view your work

- In your text editor, make sure you have word wrap and line numbers visible.
- Choose where you will save your web page's file and images.
- Make a subfolder in this location called *images*.
- Save a blank file called index.html in this location.

Step 2

Code the basic web page structure:

- **HTML** wrap – open and closing html tags.
- One (1) **Head** open and close - inside the html wrap.
- One (1) **Body** open and close - inside the html wrap and *below* the opened/closed Head section.
- SAVE.

Step 3

- Choose and code your doctype just before your opening **HTML** wrap.
- Clarify your opening html tag: <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
- SAVE.

Step 4

- Add content to the **Head** section of the page:
- Title
- Various metadata tags
- Styles (if using)
- Useful comments
- SAVE

Step 5

Author the **Body** section of your website:

- Determine the width of your website.
- Use a table for layout - <table>, <tr>, <td>, colspans, rowspans, etc. Decide if you want flexible, fixed, or mixed widths. (Or use CSS Div layouts if you have progressed to that).
- Use HTML Commenting to announce the segments to yourself to keep things straight.
- Code an Identification Banner segment to identify your page, give a logo, graphic, and/or title text.
- Code your site's Navigation links segment (absolute and relative URLs).
- Code the Content segment for your reading and image-related content.
- Code your Footer segment – copyright information, contact email, disclaimer link, etc.
- SAVE.

Step 6

Fill in your Content segment with the reading and image-related content your visitors will view

- Headers
- Subheaders
- Paragraphs of text
- Tabular table information
- Lists (bulleted, numbered)
- Image(s)
- In-text links
- Whatever else
- Anchor "back to top" links for more than 2 screen scrolls down
- SAVE OFTEN.

Step 7

- Review your coded web page in at least 3 current browsers (Internet Explorer, Firefox, Google Chrome, Safari, etc.)
- Check your layout in a free multi-browser comparison tool.
- When you make changes and corrections to your code, refresh your testing browser view.
- Keep checking your code to clean it up so it works.
- SAVE OFTEN.

Step 8

Your web page is standards compliant when your code validates and you *can* add a W3C approval icon/link to your page. Therefore,

- Validate your code with the W3C validator – in the "Validate by Direct Input" tab.
- Don't panic.
- Look for and correct errors one at a time, since one correction can fix several apparent errors.

Step 9

Make a BACK UP of your web page(s), especially your index.html. **Always**.

Step 10

- Upload your web page **and** all related images to the Web host server you plan to view it from.
- Verify it works when "live" in the world.
- Test all the links.
- Fix whatever is not working right.
- Validate your code with the W3C validator by typing your URL in the "Validate by URI" tab.

HINT: If you build a page that works, make a backup of the project. Call this backup project directory Template. Then, in the page, remove the content, leaving just the shell code for you to reuse again, and again, and again.

appendix 5
Example Disclaimer

Use freely. Just a sample, not legally vetted.
ClientName has made reasonable efforts to ensure the accuracy of the information throughout this website. However, ClientName reserves the right to make appropriate changes in information found on this website.

Nothing contained in this website shall be construed as contractual rights.

ClientName has many links to other websites. These include links to websites operated by other government agencies, nonprofit organizations and private businesses.

YourName does not endorse any content or linked materials found on websites or web services provided by and/or developed by ClientName.

When linked to another site, you are no longer on the ClientName website and this Privacy Notice will not apply. When linked to another website, you are subject to the privacy policy of that new site.

YourName does not warrant the accuracy, reliability or timeliness of any information published by this system, nor endorses any content, viewpoints, products, or services linked from this system, and shall not be held liable for any losses caused by reliance on the accuracy, reliability or timeliness of such information. Portions of such information may be incorrect or not current. Any person or entity that relies on any information obtained from this system does so at their own risk.

YourName does occasionally offer and/or make use of open source code and content that has been stated by originators as not subject to copyright or ownership. When possible we have acknowledged such open source content in the code of this website. YourName acknowledges those who provide open source content and code used in the coding of this site, such as: (list and links here).

who matters

References/Acknowledgements

I feel blessed to have had the opportunity to teach. I am primarily a designer/producer – a front-end person, and a short teaching gig dropped in my lap. Then, I never left, just pestered the college program coordinator with my skills, drive, and enthusiasm, and have been very lucky to get to teach several classes. Beginning web authoring is one of my favorites, since we get to cover many aspects of web authoring, not just one program or area.

Lots of folks have been a big part of my growth in this industry, and have provided support and laughs when things were (and are) tough in the market. Family, friends, cat, you know who you are.

I wanted to offer special thanks to Gary Nelson, of Highline Community College. He has been a fabulous design advisor, who has a wandering, illustrative way of sharing anecdotes and design concepts that help me think of design as a lifestyle as well as a career aim. He is supportive of every student who aims for excellence and creative insight.

I also want to offer thanks to Bill and Sara Newman, who have been instructors, peers, and friends in my time teaching. They give so much information, support, kudos, and enthusiasm about design/development and the teaching of it, and I have learned tremendously from them.

Finally, every student I get to work with is a gift. I get to share information, open doors, prompt A-Ha moments, duel with challengers, share chocolate, and continuously evolve and (hopefully) better my curriculum and knowledge of the web specialties and tools. Thank you all!

This is an open-source book, built primarily from my own career expertise, industry skills, and teaching experience. I also have checked for clarifying information from the following open-source materials. I take full and complete credit for my own errors. The design and layout is also mine. This book will be a work in progress over the first couple of editions, so I thank everyone for their patience.

Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Wikiversity: http://en.wikiversity.org/. Text is available under the Creative Commons Attribution/Share-Alike License.

Wiktionary: http://www.wiktionary.org/. Text is available under the Creative Commons Attribution/Share-Alike License.

Wikibooks: http://en.wikibooks.org/. Text is available under the Creative Commons Attribution-ShareAlike License.

W3Schools Web References: http://www.w3schools.com/. This website was used for reference/clarification only, and while examples were useful, any used here were created based on L.J. Bothell's experience after being compared against the standards shared in the W3C schools tutorials.

Dive into HTML5: http://diveintohtml5.org/. This web reference was visited for reference only. Copyright MMIX–MMX Mark Pilgrim

CSS3.Info: http://www.css3.info/. This website was visited for reference only. 2009 WEBFLUX. All Rights Reserved.

I have also referred to some terrific books so that you know where to look for current and relevant information related to the topics in this book. So you know what they are:

Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition; New Riders Press, author Steve Krug, ISBN-10: 0321344758, ISBN-13: 978-0321344755.

The Design of Sites: Patterns for Creating Winning Web Sites, 2nd Edition; Prentice Hall, authors Douglas K. Van Duyne, Jason I. Hong, and James A. Landay, ISBN-10: 0131345559, ISBN-13: 978-0131345553

Eric Meyer Books: http://meyerweb.com/eric/writing.html

Peachpit Press: Visual QuickStart Guide Series (books): http://www.peachpit.com/

Sams Teach Yourself Books, http://www.informit.com/

SitePoint Books: http://products.sitepoint.com/

open source

Creative Commons Licensing

bio

Author Bio

L.J. Bothell is a Seattle, WA multidisciplinary web & print designer with project coordination and writing experience. L.J.'s web specialties include usability/customer orientation, project/traffic coordination, communications planning, and streamlined production skills. Past employment has included Boeing, Group Health, WA State Ferries, The Cobalt Group, TCS Expeditions, Bocada, and freelance clients. L.J. teachs web design classes at Seattle Central Community College and writes for publication.

L.J. Bothell
www.studiobast.com

FIND IT HERE

index

What's In Web Authoring Boot Camp

index

index