

Heading One Content

Links...

- [Link 1](#)
- [Link 2](#)
- [Link 3](#)

Right Column

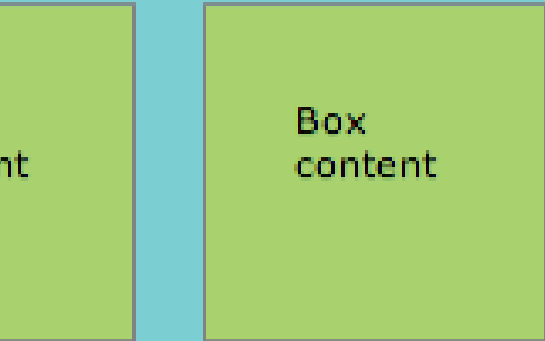
Right column content here

Box
content

Box
content

Contact info here

Heading Here



CHAPTER 1

Structure and
Design with HTML5
and CSS3

Realizing the Magic of HTML5 and
CSS39

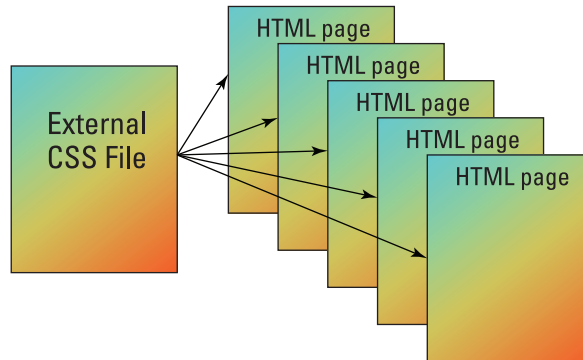
Understanding HTML Foundations.....15

Breaking Down Basic CSS27

Moving Forward with HTML5 and
CSS333

In This Chapter

- Essential new elements of HTML5
- Dynamic new design options with CSS3
- A crash course in structuring web page content with HTML
- An introduction to contemporary CSS styling techniques



In this chapter, I pull back the lens and survey the scope and range of new web design tools provided by HTML5 and CSS3. In the remainder of this book, I'll dive deeply into specific features of both HTML5 and CSS3. But here at the beginning, it will be valuable to step back from the trees to appreciate the forest.

HTML5 is a breakthrough in structuring web page content. There are all kinds of cool new features, ranging from pop-up calendars that go with input forms (see Figure 1-1) to native video that doesn't require plugins. But the big picture is a cleaner, more logical way to organize and present content. This cleaner way to organize content is concentrated in many ways in the new semantic page elements like `<article>`, `<header>`, and `<footer>`. Similarly, CSS3 provides a dynamic and fun set of new styling tools — like gradient backgrounds and irregularly shaped boxes. But the sum of CSS3 is greater than the parts. CSS3 expands and stretches what designers can do with web pages in a qualitative way.

In this chapter, I give you a sweeping, bird's-eye view of HTML5 and CSS3, in part by comparing and contrasting how pages were built in the pre-HTML5/CSS3 era, and how they can and should be built now. So, buckle up your seat belts and let's start on our journey.

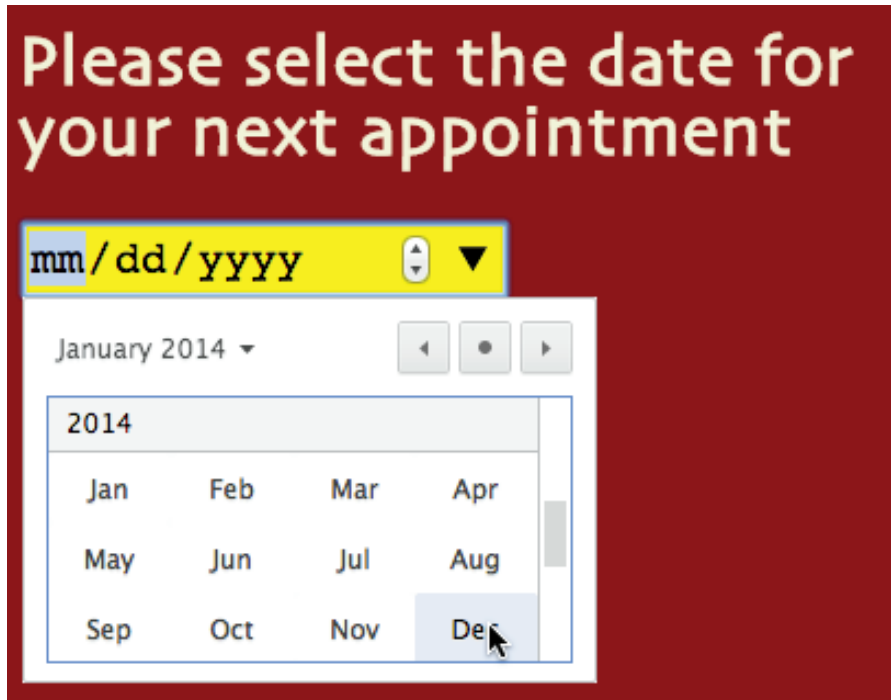


Figure 1-1

Realizing the Magic of HTML5 and CSS3

HTML5 and CSS3 open the door to designing really exciting, vibrant, and dynamic web pages. In different ways throughout this book, I will contrast new elements in HTML5 and new styling tools in CSS3 with previous versions of HTML and CSS. Here, in a compressed way, I want to quickly identify what these new features are.

Earlier versions of HTML had no systematic, universally applied set of elements for basic page content — like articles, sections, asides, and so on. HTML5 introduces a rational way to structure page content with *semantic tags* (see Figure 1-2), which describe the nature of content that they contain.

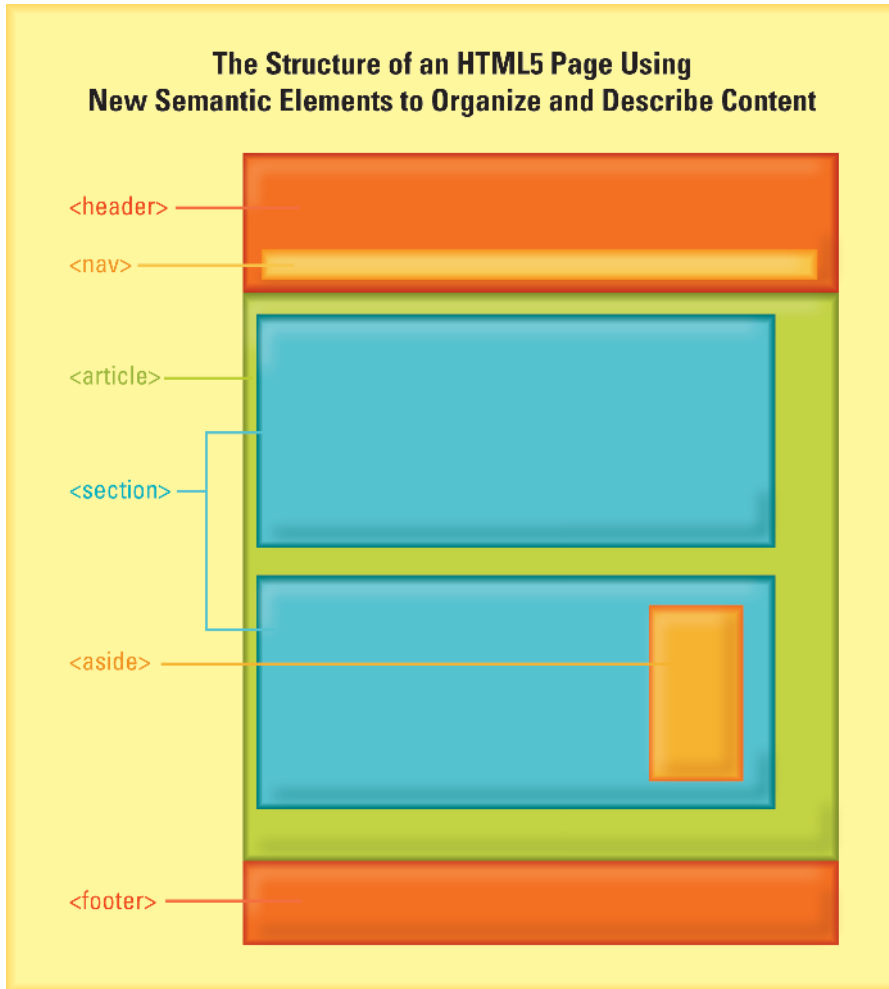


Figure 1-2

Until the advent of HTML5, plugins (like Windows Media Player or QuickTime Player) were necessary to present video. HTML5 approaches audio and video in a whole new way, freed of plugins.

And HTML5 provides form-field prompting and validation (testing), as shown in Figure 1-3.

The figure shows a form with two input fields. The first field is labeled "Name:" and contains the text "Full name go here". The second field is labeled "Email address" and contains the text "here". A tooltip message "Please fill out this field." is displayed over the second field, indicating a validation error.

Figure 1-3

CSS3 hands designers a toolbox filled with tools that makes it easy to rotate, skew, scale, and overlap page elements. Web designers can now apply rounded corners to content — even turning squares into circles.

In addition, CSS3 effects make rich graphical content available without graphics. You can, for example, easily define highly complex gradient blend backgrounds (see Figure 1-4) without requiring users to download any image file.



Figure 1-4

And I'm just scratching the surface here. By combining these different features, web designers can blaze new trails in creating websites that people will enjoy spending time visiting.

Not all new . . . but different

In two ways, all the exciting and dynamic design options made possible with HTML5 and CSS3 aren't exactly new.

First, almost all the features I rave about in the previous section have been available to web designers for some time. However, using them required complicated tools and/or high-level programming skills. In addition, they placed demands on computer and mobile resources that are no longer acceptable — particularly in an era when mobile viewing is such a critical dimension of reaching an audience. Here are a few examples to make the point(s):

▶ **Video:** Video has always been available (see Figure 1-5), but — as noted earlier — it required plugins like Windows Media Player, QuickTime Player, or Flash Player. These video players had to be updated regularly, they didn't support each other's formats (at least not without configuration), and they created an unpredictable viewing experience. With HTML5, users don't need to install

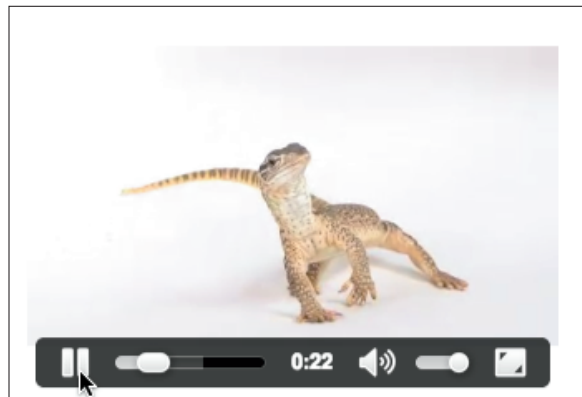


Figure 1-5

(or update) any media player plugins; all the software needed to view videos (or listen to audio) is built into browsers and accessed directly with HTML5 code.

- ▶ **Interactivity and animation:** Interactivity and animation (see Figure 1-6) have always been available, but they often required complex programming in Flash or JavaScript. Although HTML5 and CSS3 don't fully replicate the feature sets of Flash and JavaScript animation, they do make much of that feature set available at a far lower cost in terms of software, Flash or JavaScript coding skills on the part of designers, and download time for users.



Figure 1-6

- ▶ **Rich graphical backgrounds:** Rich graphical backgrounds have, until the advent of CSS3, been created by designing gradient artwork in programs like Adobe Illustrator and then saving them as web-accessible images that *tile* (repeat) in the background of a design element (like a layout box). With CSS3, these backgrounds (see Figure 1-7) can now be defined without any image files at all.
- ▶ **Forms:** Complex forms have required JavaScript, or server-side scripts, written in programs like Ruby or PHP. HTML5 provides prompts (see Figure 1-8) and validation tests without any scripting.

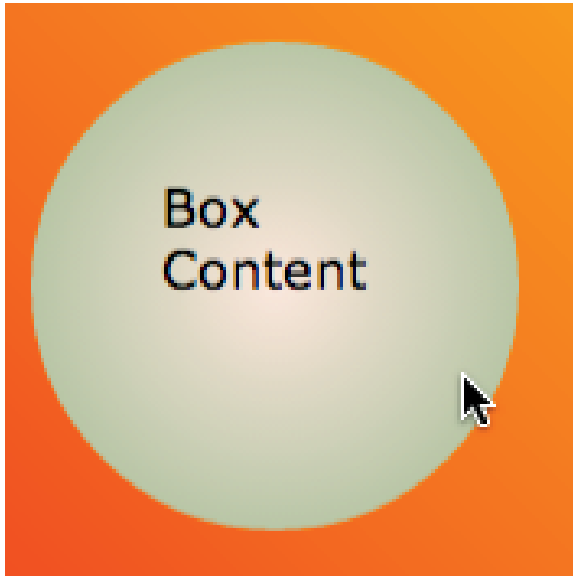


Figure 1-7

A form with two fields. The first field is labeled "Name:" and contains the text "Full name go". The second field is labeled "Email address:" and contains a warning message: "Please fill out this field."

Figure 1-8

My point here isn't to review the whole range of exciting new features I explore in this book, but rather to give examples of how HTML5 and CSS3 build on the history of web design. HTML5 and CSS3 make many features radically more accessible to designers — without having to resort to third-party products and plugins, as well as reducing download time for pages.

USE A CODE EDITOR!

Warning: Do *not* build HTML or CSS pages in a text editor. Doing so corrupts any code you create. For example, most text editors convert standard quotes (") into smart quotes (" ").

Plenty of very good free code editors are available. If you already have one, use it. If you don't already have a code editor, here are my recommendations. They aren't the most sophisticated code editors, but they're free, and easy to set up and work with:

- Windows: Notepad++, <http://notepad-plus-plus.org>).
- Mac: TextWrangler, www.barebones.com/products/textwrangler)

If you're comfortable working in another dedicated coding environment — say, Komodo Edit, Adobe Dreamweaver, Aptana Studio, or any other code editor — use that.

Second, HTML5 and CSS3 build on (and qualitatively augment) previous versions of HTML and CSS:

- ▶ **Simplification:** In some areas, new features in HTML5 and CSS3 replace older (which is a nice way of saying “clumsy, slower, problematic, and less-flexible”) techniques. For example, HTML5 simplifies audio and video embedding. (For more on using HTML5 to embed audio and video, see Chapter 7.) And with CSS3, you can easily rotate, rescale, move, or skew boxes of content without the need for positioned background images.
- ▶ **New features:** In other areas, new HTML5 and CSS3 open up completely new options. For example, HTML5 mobile tools facilitate app-like, highly mobile-friendly pages. And CSS3 effects provide options for defining transparency (allowing elements to “show through” elements above them) far more powerfully than the opacity tools in earlier CSS.

HTML5: Building on HTML techniques

I can’t emphasize this enough: HTML5 and CSS3 build on, and are extensions of, the most developed and current techniques for using HTML and CSS. Why am I so fixated on this? Because current techniques for HTML and CSS design are the launching pad from which you deploy HTML5 and CSS3.

HTML has evolved. The reason why this is important isn’t to belabor history, but rather to understand why you build pages in a certain way — and why you *don’t* build them in ways that don’t provide a solid-enough foundation to present page content in an engaging way that meets the standards of today’s demanding website visitor.

DO OLD BROWSERS SUPPORT HTML5?

With the exception of Internet Explorer (IE), all browsers prompt users to update frequently, so when I talk about “old browsers” I mean old versions of Internet Explorer that are installed in large corporate or institutional environments that, for reasons internal to the security and networking environments, can’t be updated. All other browsers (Firefox, Chrome, Safari, Opera, and so on, in their laptop/desktop and mobile versions) essentially update automatically.

But there is, and will continue to be, a substantial installed user base for old versions of Internet Explorer. What about that community? When someone asks, “Will my old version of IE support HTML5 and CSS?,” the short (and accurate) answer is yes. In fact, HTML5 pages are less problematic in old versions of IE than previous versions of HTML; the HTML5 document type declaration that identifies a page as HTML5 reduces error messages that old browsers sometimes display when they encounter minor errors in HTML coding. That said, old browsers (here, I mean Internet Explorer 8 and earlier) do not support all the new features in HTML5 and CSS3. But even in environments where support for old versions of IE is critical, this presents fewer issues than you might imagine. In most cases, new HTML5 elements and new CSS3 effects can be deployed in a way that *enhances* a visitor’s experience, but are not *essential* to that experience.

Take a very basic example: Eons ago (in “web design years” that is, which is more like 10 years ago in human time), web pages were designed with tables. Tables were included in HTML by the initiators of the web to display rows and columns of data. (Tables still play that role, by the way.)

But at a certain point in the development of the web, envelope-stretching designers figured out ways to use table cells to locate content on pages. This technique ushered in a whole new era of web design that broke out of single-column, boring pages full of text and unaligned images.

remember

Here is a critical point in implementing HTML5 and CSS3: Because both these programming languages build on previous versions of HTML and CSS, the better grounded you are in basic HTML and CSS approaches and techniques, the more you’ll get out of implementing HTML5 and CSS3.

Tables have been replaced in modern web design by using `<div>` (short for *divider*) tags combined with style rules that format `<div>` tags into boxes for layout.

To complete the example, unless your pages are built with `<div>` tags (as opposed to tables), a whole array of really fantastic new CSS3 effects, ranging from box-shadows to rounded corners to rotation, won’t be deployable.

Similarly, new CSS3 design tools can’t be used effectively unless your website is built with external style sheets. Older techniques of embedding styles within pages — or even older techniques of using HTML style properties (such as `"align=right"`) to style page content — will radically reduce a designer’s freedom to take full advantage of CSS3 styling.

Understanding HTML Foundations

Take a minute to see the essential, current HTML approaches and techniques on which HTML5 can soar. First, note that I said “approaches and techniques” — not “syntax and coding rules.” You do *not* need anything close to an encyclopedic knowledge of HTML to build sites with HTML5!

tip

I list a few helpful resources for HTML tutorials and syntax in the “Need More HTML Basics?” sidebar later in this chapter.

And of course, I walk you through essential HTML code and syntax in this book.

ORGANIZE SITE CONTENT IN A SINGLE FOLDER

One essential element of modern website building is that all the files for a website need to be organized in a single folder. If you're new to building modern sites, start by creating a folder on your computer to organize all your content. You can, of course, divide your root site folder into subfolders for images, video, and so on. Just remember that modern websites rely on multiple and complex linkages between files; a single page may rely on links to style sheets, links to scripts, navigation links to other pages in the site, embedded video and image files, and more.

remember

You should have a handle on a handful of very basic HTML approaches and techniques in order to make the most of HTML5. Most importantly, defining styling with external, linked CSS style sheets, not embedded styles in HTML, and designing pages with `<div>` elements, not tables.

Five things you need to know about HTML

I want to cut to the heart of things. There are a few basic things you need to know about HTML — in general — in order to get the most out of HTML5. The following list breaks these down:

1. HTML5 files are identified with a `doctype` declaration that tells browsers, “I’m an HTML file.” This declaration is the first line in an HTML file (see Figure 1-9) and looks like this for HTML5:

```
<!DOCTYPE HTML>
```

HTML5 files are identified with a `doctype` declaration that tells browsers “I’m an HTML file.” All page content is enclosed between an open `<html>` tag and a close `</html>` tag. The `<html>` element is divided into the `<head>` and `<body>`.

```
<html>
<head>
</head>
<body>

</body>
</html>
```

Figure 1-9

2. All page content is enclosed between an open `<html>` tag and a close `</html>` tag (see Figure 1-9).
3. Inside the `<html>` element, page content is divided into the `<head>` element (content that doesn't display in a browser window) and a `<body>` element (content that does display in a browser window). Once again, see Figure 1-9.
4. HTML elements within the `<body>` element define visible content, including headings, paragraphs, lists, links, images, and other key content (see Figure 1-10). `<div>` tags matched with class styles (that can be used more than once on a page) or ID styles (that can only be used once on a page) are the basic building blocks of page design.

```
<html>
<head>
</head>
<body>
  <div>
    <h1>Heading</h1>
    <p>Paragraph</p>
    <ul>unordered list</ul>
      <li>item 1</li>
      <li>item 2</li>
    </ul>
  </div>
</body>
</html>
```

Figure 1-10

5. Any web pages beyond the most primitive link to external style sheet (CSS) files that determine how page content looks (see Figure 1-11).

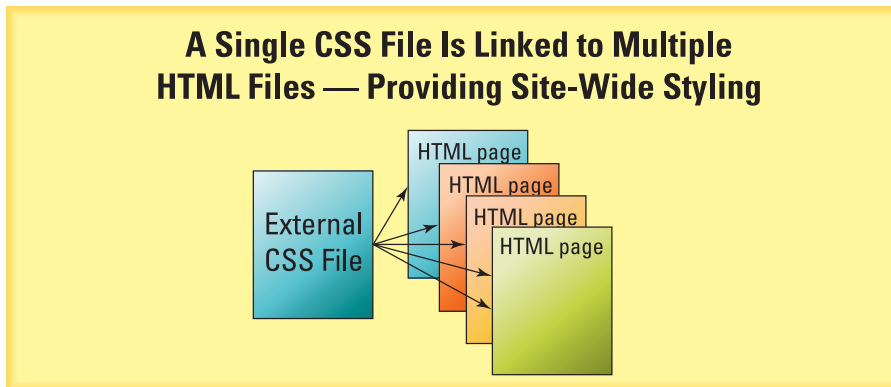


Figure 1-11

If all this is old news to you, good! You're in the right place. Bear with me while I review some basic concepts to help identify the key foundations from which you will implement radical new elements in HTML5.

If all this is new news to you, that's fine too. You'll be okay — I walk you through how all this works in a concise way in the rest of this chapter.

Getting started with a basic HTML template

One good (and quick) way to refresh/reground/learn basic HTML5 concepts is to start with an HTML page and break down the different components. Start with a basic HTML5 template.

You can copy the code in Listing 1-1 and paste it into your code editor, or just kick back, grab a snack, and keep this page handy in your book or e-reader. I spend the next few sections walking through Listing 1-1 more or less line-by-line, and ultimately come out the other end with an understanding of the essentials of HTML.

TAGS VERSUS ELEMENTS

What's the difference between an HTML *tag* and an HTML *element*? Not much, and sometimes the terms are used interchangeably. But it is worthwhile to identify the difference. Most HTML elements are enclosed inside an open tag (like `<body>`) and a close tag (like `</body>`). There are also some single-tag elements (like the `
` tag that forces a line break). In short, most elements are bookended by an open and close tag.

on the web

All the code listings used in this book are available for download from the Downloads tab on the book's companion website at www.dummies.com/extras/html5andcss3.

GOING LIVE

A full examination of how to contract for (and work with) remote web host providers is beyond the scope of this book. (You'll find an in-depth exploration of that challenge in *Building Websites All-in-One For Dummies*, 3rd Edition.) However, if you want to work through this book experimenting with a real, live, online site, you need two things:

- A remote hosting service: If you want to try a free one, head over to www.000webhost.com and sign up for this no-cost, ad-free service.
- An FTP program: You use this to transfer files from your computer to your remote site. (Your remote host will supply you with the information you need to log into your remote site.) You can download the free FileZilla FTP application from <https://filezilla-project.org/download.php>.

If you download Listing 1-1, save it as `template.html` so you can use it as you work through the remainder of this chapter.

Listing 1-1: `template.html`

```
<!--The HTML5 doctype (document type) declaration is very
      simple-->
<!DOCTYPE HTML>
<!--All page content is inside the HTML element-->
<html>
<!--Head element content is not visible in a browser window-->
<head>
<!--The UTF-8 character set supports all symbols and
      characters-->
<meta charset="UTF-8">
<title>HTML Template</title>
<!--The following line links to our style sheet file-->
<link rel="stylesheet" type="text/css" href="style.css">
</head>
<!--Content visible in the browser's window is inside the HTML
      element-->
<body>
<!--All our content is enclosed in the wrapper ID style-->
<div id="wrapper">
<h1>Heading One Content</h1>
<!--column-2 ID style is floated left and used for
      navigation-->
<div id="column-2">
<h2>Links...</h2>
<ul>
<li> <h3><a href="#">Link 1</a></h3></li>
<li> <h3><a href="#">Link 2</a></h3></li>
<li> <h3><a href="#">Link 3</a></h3></li>
</ul>
<!--column-1 ID style is floated right and used for content-->
</div>
<div id="column-1">
<h1>Right Column Heading Here </h1>
<p>Right column content here </p>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
</div>
<!--The clear class style clears (removes) float-->
<div class="clear"></div>
<h4>Contact info here </h4>
</div>
</body>
</html>
```

Identifying HTML document structure

The essence of the HTML document in Listing 1-1 is shown in Figure 1-12 and is as follows:

- ▶ **The `<!DOCTYPE HTML>` document declaration:** This tells browsers, “I’m an HTML page.” This works in any browser; see the sidebar “Doctype declarations new and old” for more information.
- ▶ **The `<html>` element:** This wraps the entire document in HTML.
- ▶ **The `<head>` element:** This element holds metadata associated with the page (like a description of the page or the text that appears in a browser title bar). This is information not displayed in a browser window.
- ▶ **The `<body>` element:** This element holds all the content displayed in a browser.

```
<html>  
  
  <head>  
  </head>  
  <body>  
  
  
  
  
  
  
  
  
  
  </body>  
  
</html>
```

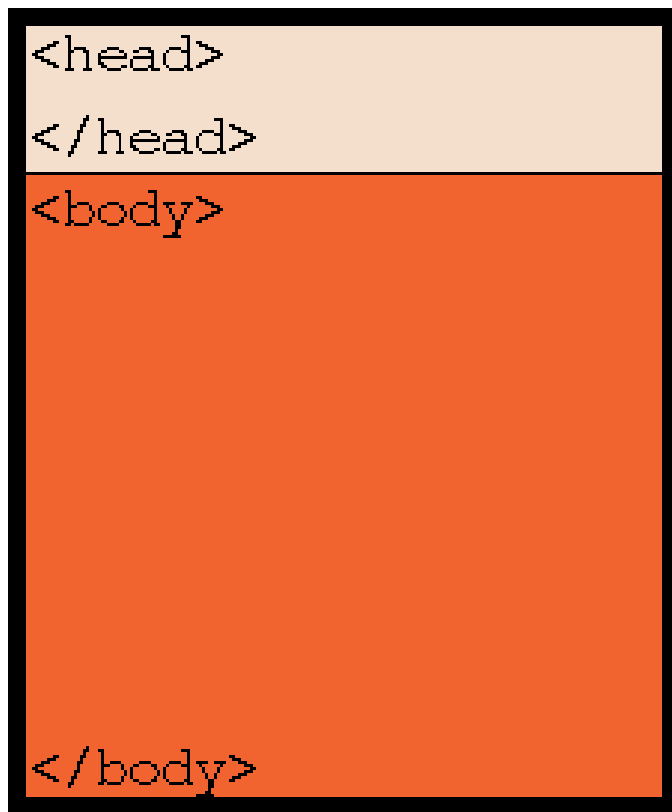
A diagram illustrating the structure of an HTML document. It is enclosed in a thick black rectangular border. The top section is a light beige rectangle containing the code <head> and </head>. The bottom section is a larger orange rectangle containing the code <body> and </body>. The entire structure is wrapped in a black border, with the opening <html> tag at the top and the closing </html> tag at the bottom.

Figure 1-12

HTML documents can include comments. The syntax for a comment is

```
<!--this is a comment-->
```

warning

All markup, including the head information and the comments, are easily viewable via “view source” options in browsers. Avoid including anything in markup language that you do not want anyone else to see.

DOCTYPE DECLARATIONS NEW AND OLD

Other `doctype` declarations are used for older versions of HTML. Although not all browsers support new HTML5 elements (something I cover in detail throughout this book as I examine specific HTML5 elements), that doesn’t matter when it comes to a `doctype` declaration because every browser can interpret the `<!DOCTYPE HTML>` tag to recognize that this is an HTML page.

Knowing basic element syntax

Here are a few basic rules to help understand how HTML elements are defined:

- ▶ Elements usually have open and closing tags, like `<p>` and `</p>`.
- ▶ Element parameters are defined with the open tag. For example, if a paragraph element is associated with a class style named “`hilite`”, (something I explore later in this chapter in the “Deploying class styles” section), the open tag might look like this: `<p class="hilite">`.
- ▶ Element content is usually enclosed between the open and close tags. So, a paragraph looks like this:

```
<p>Welcome to our site!</p>
```

remember

You don’t have to close all elements in HTML5 — you can get away with leaving a `<p>` tag unclosed. However, it is best practice to close tags.

Working with the <head> element

The <head> element is defined after the open <html> tag. Head element content includes

- ▶ **The `charset` meta tag:** A *meta tag* is a tag that defines everything in the document, and the `charset` meta tag defines the character set as the universally supported UTF-8.

tip

UTF-8 is an acronym for UCS Transformation Format—8-bit. It is an encoding system with the capacity to represent every symbol and character in the widest available set of alphabets including nearly every Latin-derived alphabet, along with the Greek, Cyrillic, Coptic, Armenian, Hebrew, Arabic, Syriac, and Tāna alphabets.

- ▶ **The <title> tag within the title element:** This tag defines the page title that appears in the title bar or tab bar of a browser.
- ▶ **A link to the CSS style sheet file:** This link element defines the relationship of the link to the page (`stylesheet`), the type of file (`text/css`), and the `href` (link to the page). (Later in this chapter, I dissect a CSS file for you.) The HTML link to a style sheet code looks like the one shown in Figure 1-13.

```
<head>
<meta charset="UTF-8">
<title>HTML Template</title>
<link rel="stylesheet" type="text/css" href="style.css">
</head>
```

Figure 1-13

Using the <body> element and <div> tags

The <body> defines content that appears in a entire browser window. However, the <body> tag isn't sufficient to define the contours of the page in which visitors will see content.

While the style associated with the <body> tag defines things like the background color for the entire web page, most designers use a separate element to define the box that contains all the page content. This is, generally, a <div> element.

remember

The <body> element holds everything you want to display in a visitor's browser.

FLUID DESIGN, RESPONSIVE DESIGN, AND THE 960 GRID

There are different approaches used to define the width of the `<div>` tag box that constrains the page content. Until the last few years, the prevailing doctrine was *fluid design*. Essentially, that meant creating pages that varied in width depending on how wide a user sized his or her browser window.

More recently, the concept of fluid design has been transcending with the concept of responsive design. *Responsive design* recognizes that the differences between laptop/desktop environments, tablets, and smart phones are so substantial that pages need to be completely redesigned for each environment, and that simply resizing page content is not sufficient.

In Chapter 9, I explore responsive design in depth and walk through new techniques in HTML5 for implementing responsive design with media queries and jQuery Mobile pages. Here, I will focus on building a desktop/laptop page.

For reasons related to aesthetics and accessibility, and to facilitate collaboration between designers in professional design workflows, the most widely applied approach to full-sized page design in professional design environments today is the 960 grid. Our model applies the *960 grid*, which in essence means constraining the page content that is displayed in full-sized browsers to 960 pixels wide.

The actual properties of the box that holds page content are defined in a linked style sheet. I provide you with a template CSS style sheet file that matches the HTML I'm creating here a bit later in this chapter in the section "Breaking Down Basic CSS." That style sheet includes an ID selector (style definition) named "wrapper". For now, note that everything inside the `<body>` tag is enclosed in a `<div>` tag.

In our template model, everything inside the `<body>` tag is enveloped within the main `<div>` tag (see Figure 1-14). The `<body>` tag defines how the page background looks, but the main `<div>` element defines the box that displays all the rest of the page content appears.

```
<body>  
<div id="wrapper"> ☰ </div>  
</body>
```

Figure 1-14

<DIV> TAGS: A HALLMARK OF FUNCTIONAL WEB DESIGN

Even with new page structure elements in HTML5, `<div>` tags are still a useful option that can be used together with HTML5 elements to create more semantic markup.

You might have heard that HTML5 replaces `<div>` tags for page layout. And that's understandable because HTML5 elements do replace *some* need for `<div>` tags. But the actual relationship is that HTML5 elements simplify and supplement the use of `<div>` tags, but `<div>` tags remain the essential building block for contemporary web design.

The `<div>` tag is used to create *divisions* within a page. ID and class selectors in style sheets define how those `<div>` tags look on a page. The main `<div>` tag is paired with an ID style named "wrapper". Take a look at the style associated with that ID selector ("wrapper") later in this chapter in the "Breaking Down Basic CSS" section, where I dissect the style sheet that goes with this page.

Using headings, lists, and links

Headings (h1, h2, and so on), lists (either numbered or bullet lists), and links are basic elements of HTML pages. While none of these elements has undergone any changes in HTML5 — compared to previous versions of HTML — I want to take a minute to review how they work.

The example page (refer to Listing 1-1) has a heading 1 (`<h1>`) element, right after the "wrapper" ID `<div>` tag opens. The heading 1 element content is surrounded by an open and a close tag (see Figure 1-15).

```
<div id="wrapper">  
<h1>Heading One Content</h1>  
<div id="left-column">
```

Figure 1-15

tip

There are six heading elements in HTML: h1 through h6. This has not changed in HTML5. By default (and in standard usage), `<h1>` elements are the most important, and `<h6>` elements are the least important. Styling for heading elements can be customized in a style sheet. And, of course, you don't need to use every one of the heading elements in web pages (our template example uses four heading elements).

A second `<div>` tag with an ID selector of "column-2", holds content displayed in the left column of the page (see Figure 1-16).

Inside the left column you see

- ▶ An `<h2>` heading element
- ▶ An unordered list (``) element
- ▶ Three list (``) elements within the unordered list
- ▶ Each `` element is further defined with paragraph (`<h3>`) elements

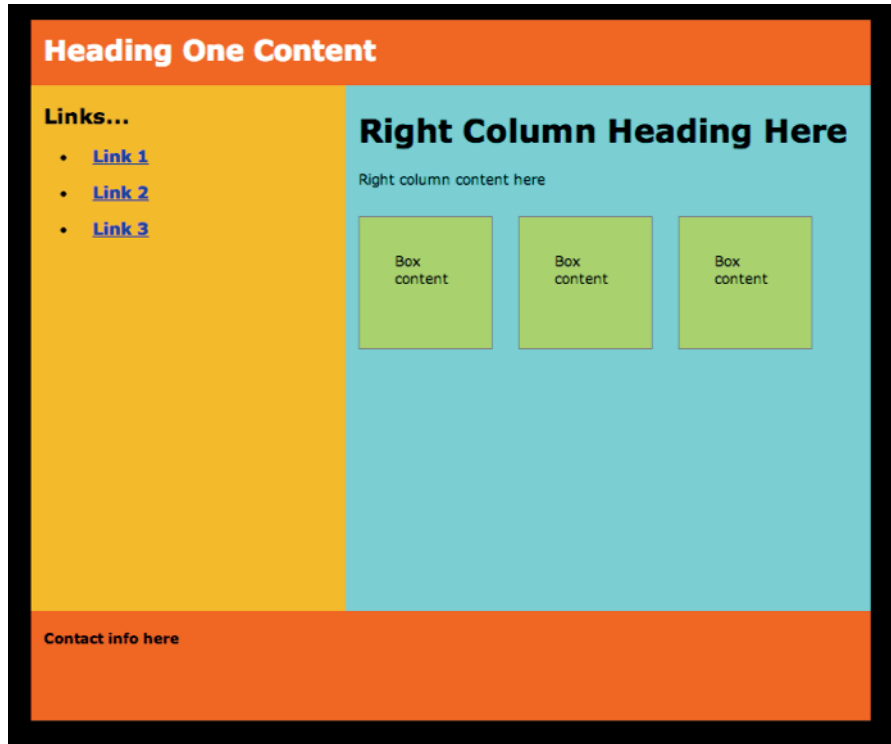


Figure 1-16

Here are a couple of things to note in relation to this section of code:

- ▶ **Local trumps global:** The “cascading” in Cascading Style Sheets refers to how styling is prioritized. The rule is that the inner-most element will override any styling inherited from the outer elements. In this example, the `<h3>` tag, with its large type and vertical spacing, trumps the size and spacing inherited from the list elements. I like to think of this as “local trumps global.”
- ▶ **Links:** Second, this section of code also includes links (see Figure 1-17). Link syntax opens and closes with `<a>` and `` tags, and includes the link URL and the text (or image) that displays in a browser.

```
<a href="#">Link 1</a>
```

open link tag link URL display text link close tag

Figure 1-17

Deploying class styles

Before wrapping up this crash course in current HTML approaches, see how class styles are deployed in the column-1 `<div>` tag (see Figure 1-18):

```
<div id="right-column">
<h1>Right Column Heading Here </h1>
<p>Right column content here </p>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
<div class="box"><p>Box content</p></div>
</div>
```

Figure 1-18

- ▶ Class styles are appended to `<div>` tags (or other elements) just like ID styles, with the same syntax. The difference is that you can use a class style with multiple `<div>` tags. Comparatively, ID styles can be applied to a single element only.
- ▶ Because I want all the boxes to have identical styling, I apply the same class style to them all.
- ▶ Finally, note the class style called "clear" (see Figure 1-19) after the box styles. This is handy because the style sheet applies *float* to these boxes, causing them to all display in the same row instead of separate rows (which is the default for most elements).

```
<div class="clear"></div>
```

Figure 1-19

The "clear" class style has been defined to wipe out the float property for everything that follows so that the float is not inherited.



For more exploration of using float in page design, see Chapter 5.

NEED MORE HTML BASICS?

The best resource for a basic grounding in HTML basics — beyond what I can cover here in this crash course — is *Beginning HTML5 and CSS3 For Dummies* by Ed Tittel and Jeff Noble.

Other online resources for HTML code help include:

- www.w3.org
- www.w3schools.com
- www.webplatform.org
- <http://stackoverflow.com>
- www.sitepoint.com

Breaking Down Basic CSS

Having crashed your way through HTML basics, it's time to tackle the basics of CSS! Listing 1-2 is a CSS file that matches the `template.html` file I dissect earlier in this chapter. Before you start, download Listing 1-2 and save it as `style.css` within the same folder on your computer that you used to save the `template.html` file.

on the web

You can download Listing 1-2 from the Downloads tab on the book's companion website (www.dummies.com/extras/html5andcss3).

I spend the next few sections working through this `style.css` file, and you can review and reinforce any skills that need strengthening by playing around with the `template.html` and `style.css` files. And of course, if you accidentally trash them, you can return to the book's companion website, download them again, and start fresh.

Listing 1-2: `style.css`

```
@charset "UTF-8";
/* CSS Document */

/* The body tag style applies to all elements on the page */
body {
  background-color: black;
  font-family: Verdana, Geneva, Arial, sans-serif;
```

continued

Listing 1-2 *(continued)*

```
padding:0px;
margin:0px;
}

/* The wrapper ID style is used with a div tag to provide a 960px wide
   page */
#wrapper {
width: 960px;
height: 800px;
margin-left: auto;
margin-right: auto;
background-color: #F25F29;
}

/* The column-1 ID style is floated right */
#column-1 {
float: right;
width: 600px;
height: 600px;
background: #55D9D9;
}

/* The column-2 ID style is floated left */
#column-2 {
float: left;
width: 360px;
height:600px;
background: #F2B544;
}

/* Selector for tags separated by commas applies the style to all tags */
h1,h2,h3,h4,h5,h6,p,li {
margin-left:15px;
}

h1 {
color: white;
padding-top:15px;
}

/* Selector for tags not separated by commas applies in specific
   circumstances*/
#column-1 h1 {
padding-top:5px;
color: black;
font-size:36px;
}

/* Advanced web design relies on class or ID style boxes*/
.box {
```

```
height: 100px;
width: 100px;
float: left;
margin: 15px;
padding: 25px;
background: #A8D977;
border: 2px solid gray;
}

/* The following pseudo-class applies to the box class when in a hovered
   state */
.box:hover {
background-color: #F2B544;
border-bottom: 2px solid black;
}

/* This clear class style terminates float */
.clear{
clear: both;
}

}
header, footer {
background-color: #F27830;
color: red;
padding-top: 5px;
padding-bottom: 5px;
}
```

DEFAULT BROWSER STYLING

Different browsers impose different default styles. For example, many full-sized browsers, by default, include ten pixels of padding at the top of the page in order to keep content from bumping into the top of the browser window. Mobile browsers, given their smaller viewports, might have less (or no) default margin. Another example: Many browsers have defined ways of presenting drop-down menus to maximize accessibility.

There are two basic approaches to address default browser styling. One way is for designers to do everything possible to override default styling in different browsers. Tools for doing this are found at the [normalize.css](http://necolas.github.io/normalize.css/) site (<http://necolas.github.io/normalize.css/>). The other approach is to appreciate the value of different default styling for different browsers in different devices, and design pages with enough flexibility to work well in different browsers with their default styling. I explore the wide range of browsers and how they handle styling in Chapter 8.

Creating a CSS document

Defining a document type is simple in CSS — you don't have to do anything. Nice, huh? Most designer, however, open CSS documents with a UTF-8 character set declaration like this:

```
@charset "UTF-8";
```

tip

By adding a UTF-8 character set declaration, if anything in the CSS style sheet code contains characters outside the standard Western alphabet and number characters, browsers can still interpret the page.

The sample `style.css` file also opens with a comment (see Figure 1-20). Comments open with `/*` and close with `*/`. Comments are notes by the coder that do not affect the code itself, but simply help document the purpose of code.

```
@charset "UTF-8";  
/* CSS Document */
```

Figure 1-20

Examining CSS style definitions

Each individual style within a style sheet is a *selector* (sometimes called a *rule*). Each selector has sets of properties (*declarations*) that consist of a property and a value (or set of values).

You can see how this works by examining the `body` style definition (see Figure 1-21):

- ▶ The `body` element is the selector.
- ▶ The four declarations for this style are `background-color`, `font-family`, `padding`, and `margin`.

```
body {  
  background-color: black;  
  font-family: Verdana, Geneva, Arial, sans-serif;  
  padding: 0px;  
  margin: 0px;  
}
```

Figure 1-21

tip

Declarations never have spaces within them. A widely-used technique is to separate words with underscore characters (`_`) or dashes (`-`) instead of spaces.

- ▶ Each declaration has a property (like `background-color`) followed by a colon (`:`) and a value (like `black`).
- ▶ Each declaration ends with a semicolon (`;`).

That's it. With these basic rules in mind, see what you can learn from the `style.css` sample CSS code file.

I could fill the remainder of this book (or at least a few chapters) with a comprehensive list of all the available CSS properties and values, but that would be environmentally irresponsible — seriously, and a waste of space as well. When I discuss new CSS3 properties in this book, I'll walk through them in detail.

For now, I want to point out a few key CSS properties as you examine techniques in the model style sheet. In that light, focus on a few implementations of critical CSS properties that you will build on when you dive into CSS3:

ONLINE RESOURCES FOR CSS RULES

Two online resources for CSS rules are:

- W3.orgs CSS documentation: www.w3.org/Style/CSS
- W3Schools CSS page: www.w3schools.com/css

- ▶ ID selector names begin with `#`, and class selector names begin with a period (`.`).

tip

Selector names can't have spaces. As with declarations, designers typically use underscore characters (`_`) or dashes (`-`) in place of spaces.

- ▶ The `#column-1` and `#column-2` selectors have `float` properties, aligning them right and left (respectively), and keeping them in the same horizontal row — side by side.
- ▶ Color and background color values are either in standard colors (for example, `"white"`) or hexadecimal color values. You can find a wide range of online resources for getting hexadecimal values for colors, ranging from Adobe's Kuler site (<https://kuler.adobe.com>) to a nice simple chart at Total Recall (<http://html-color-codes.com>).

tip

Hexadecimal colors are the standard for defining colors (although there are other options).

▶ Pseudo-class selectors are created by placing a colon (:) after a class name, followed by a state (see Figure 1-22). For example, the `.box:hover` selector defines how the box looks when a user hovers over the box with a cursor or when the box is tapped in a mobile device. Pseudo-classes define link states and also define different properties for any element acted on by a user.

States include

- `:link`: An unvisited link
- `:visited`: A visited link
- `:hover`: A hovered link (as in the sample page)
- `:active`: A style for an active element: for example, a link in the process of being opened
- `:focus`: An element “in focus,” such as a form field in which a user has inserted a cursor

```
.box:hover {
background-color:#F2B544;
border-bottom:2px solid black;
}
```

Figure 1-22

Combining style definitions

Before wrapping up this compressed overview of CSS, look at how style definitions can be combined. Style definitions for sets of selectors that are separated by commas apply to every selector in the list (see Figure 1-23).

```
h1,h2,h3,h4,h5,h6,p,li
{
margin-left:15px;
}
```

Figure 1-23

On the other hand, style definitions for sets of selectors *not* separated by commas apply only a specific set of circumstances, where the last element in the list is enclosed in every previous element in the list (see Figure 1-24). For example, the `#column-1 h1` style applies only to `<h1>` elements that are inside a `#column-1` ID element.

```
h1 {
color: white;
padding-top:15px;
}

#right-column h1 {
padding-top:5px;
color: black;
font-size:36px;
}
```

Figure 1-24

Note that the `h1` style applies to the `<h1>` content differently in the right column. The color in the right column is black instead of white (see Figure 1-25), which applies to normal `<h1>` elements, and the sizing is different as well.



Figure 1-25

Moving Forward with HTML5 and CSS3

Everything in in this crash course/review of basic HTML and CSS has two implications for building cutting-edge pages with HTML5 and CSS3:

- ▶ HTML5 and CSS3 *build on* HTML and CSS, especially current generation design techniques.
- ▶ You're gonna use it all!

Bearing those two points in mind (okay, maybe two ways of saying the same thing), you're ready to build pages in radically new ways that enhance web designer productivity and provide animation, interactivity, and a greatly enhanced experience for visitors.