Dreamweaver CS5.5 Mobile and Web Development with HTML5, CSS3, and jQuery

David Karlins

Chapter No.3
"Customizing HTML5 Layout Elements " 
In this package, you will find:

A Biography of the author of the book
A preview chapter from the book, Chapter NO.3 "Customizing HTML5 Layout Elements"
A synopsis of the book’s content
Information on where to buy this book

About the Author

David Karlins is a consultant, writer, and teacher on digital graphics and interactive design solutions. He has written or co-authored some fifty books, professional instruction videos, and apps on web design, vector graphic design, digital photography, sports photography, project management, digital video, and animation.

David Karlins' consulting clients have ranged from Hewlett Packard to the Himalayan Fair, from AAA Health Insurance to the Association of Alternative Newsweeklies.


Thanks to Wilson D'souza, Srimoyee Ghoshal, Priya Mukherji, and the entire management and staff at Packt Publishing. I would also like to thank my agent Margot Maley Hutchison.

For More Information:
Dreamweaver CS5.5 Mobile and Web Development with HTML5, CSS3, and jQuery

Dreamweaver is the most powerful and industry-leading web design software that utilizes innovative web technologies such as HTML5, CSS3, and jQuery for web and mobile development. These technologies have radically reconfigured the process of designing the web content and function in the widest possible range of browsing environments ranging from desktops to mobile devices.

For experienced Dreamweaver designers and for designers who are new to Dreamweaver, this book explains in detail how to take advantage of the new features available in the latest releases of Dreamweaver that add support for HTML5, CSS3, and jQuery. In addition to this, the book also contains detailed systematic directions for building mobile applications in Dreamweaver CS5.5.

This book starts off by teaching you to create web pages in Dreamweaver using the latest technology and approaches—HTML5, CSS3, and JavaScript. It demonstrates how to create or customize pages with HTML5 layouts and add multimedia to these pages with HTML5 elements. Then, you will learn to add various CSS3 effects to web pages. This book also covers different techniques of adding interactivity to web pages. The later chapters show how to optimize web pages with Dreamweaver for display in various browsing environments. You will also learn to build jQuery-based mobile apps from scratch in the later chapters. By the time you finish reading this book, you will have learned several techniques to use the latest features of Dreamweaver for web and mobile development.

For More Information:  
What This Book Covers

Chapter 1, Creating HTML5 Pages in Dreamweaver, begins with the exploration of creating HTML web pages with Dreamweaver CS5.

Chapter 2, Customizing HTML5 Layout—Content and Look, provides information about customizing both the content and the look of HTML5 pages generated with the HTML5 Layouts in Dreamweaver.

Chapter 3, Customizing HTML5 Layout Elements, provides an exploration of using new HTML5 layout elements, including <header>, <nav>, <article> and <section>, <aside>, and <footer>.

Chapter 4, Building HTML5 Pages from Scratch, guides you through the process of building modern standards-compliant pages relying entirely on HTML5 layout tags.

Chapter 5, Defining and Implementing Multiscreen Previews and Media Queries, guides you through the process of providing media-sensitive content for a variety of viewports, ranging from large-screen projections of websites to hand-held devices.

Chapter 6, Applying CSS3 Effects and Transforms, highlights the importance of using CSS3 to format effects such as drop-shadows, rounded box corners, and opacity (transparency) along with transforms that change the shape, location, rotation, and size of objects.

Chapter 7, Embedding HTML5 Audio in Dreamweaver, guides you through the process of embedding native (browser-based, not plugin-based) audio to web pages using Dreamweaver CS5.5 tools for HTML5 media.

Chapter 8, Embedding HTML5 Video in Dreamweaver, provides information about embedding a variety of HTML5-compliant video formats to web pages using HTML5 and Dreamweaver CS5.5.

Chapter 9, Creating Mobile Pages with jQuery, guides you through the process of creating jQuery Mobile-based pages—accessible, inviting, animated pages that work particularly well in mobile devices.

Chapter 10, Adding jQuery Mobile Elements, provides information about building jQuery Mobile-based pages from scratch with layout grids, and collapsible blocks.

Chapter 11, Generating Apps, guides you through the process of publishing mobile apps for iOS (iPhone, iPod Touch, and iPad) and Android devices using new tools in Dreamweaver 5.5.

For More Information:
In this chapter, we will continue our journey exploring HTML5 layouts in Dreamweaver CS5.5 (and CS5). In the previous chapter, we customized the content of the HTML5 layout elements that are generated with the two HTML5 layouts that come with the HTML5 Pack. In this chapter, we will examine and customize HTML5 layout elements—<header>, <nav>, <article>, <section>, <aside>, and <footer>.

This chapter, and the following one, will together form a double-edged attack on digesting how HTML5 layout elements are used to design pages. In this chapter, we will work through something the HTML5 layout elements generated in the Dreamweaver HTML5 Pack elements in more detail and depth, examining both specific uses of elements in combination with different kinds of CSS styles and the logic behind those techniques. In the next chapter, we will build something closer to an "ideal" HTML5 layout page, relying more consistently on HTML5 layout elements. By the time you have worked through both of these chapters, you will have had a chance to assimilate both the art and science of HTML5 layouts to the point that you can build best practise pages and deconstruct and work with more typical sites that tend to be a mishmash of HTML5 elements and other components.

What will we accomplish by critically dissecting the two HTML5 layouts provided by Dreamweaver?

For More Information:
Customizing HTML5 Layout Elements

First, the two HTML5 layouts that come with the HTML5 Pack are thoughtfully put together and a very flexible foundation on which to build all kinds of page designs. If I can editorialize for just a moment: I have been writing books, articles, and so on about Dreamweaver since before you were born and in the past, I have not had such positive things to say about the sample layouts included in previous versions of Dreamweaver, including pre-HTML5 Pack Dreamweaver CS5. That, to be fair though, has not mainly been the fault of Dreamweaver or the developers at Adobe. The problem has been with HTML itself. Lacking any standardized page layout tags, HTML lent itself to anarchy, chaos, and confusion—and I don't mean that in a good way.

Secondly, in part because Dreamweaver’s HTML5 layouts are quite complex, deconstructing them in rather tenacious detail is a good way to master HTML5 page design with Dreamweaver.

As fair warning, I will alert you to the fact that, more than in previous chapters, the "recipe" at the end of the chapter will constitute a good chunk of the experience here. The layouts come with useful documentation, but that documentation assumes quite a bit of perspective, which we will arm ourselves with before dissecting the layouts.

However, don't skip to the end of the chapter. We will prepare for the excursion through the layout with some conceptual and specific exploration of formatting HTML5 elements, including a survey of compatibility challenges presented by browsers that do not support HTML5 and the ways in which HTML5 layout elements fit into other layout tools such as the <div> tag and older HTML tags.

In this chapter, we will learn to:

- Understand and implement the ongoing need for <div> tags as layout elements within a framework of HTML5
- Identify and address browser compatibility issues for HTML5 layouts
- Explore and adapt CSS styles applied to HTML5 layout elements
- Understand and manage the class CSS styles to apply formatting across multiple HTML5 layout elements
- Control the element background, margins, padding, float, and other attributes applicable to many HTML5 layout elements
- Manage CSS styles and additional HTML5 elements specific to a particular layout elements (for example, using the <address> element within footer)

For More Information:
HTML5 layout and browser compatibility challenges

Ready for a not-news flash? Ready or not, here it is, HTML5 is a work in progress.

What does that mean? In historic terms, it means that the standards for HTML5 are still being systematized, although a critical mass has been reached where the meaning and effect of most HTML5 terms is widely agreed on and browsers either support HTML5, or they are evolving in that direction in finite terms.

For developers, it means that as we exploit time and stress saving features of HTML5 to build web pages, we have to be aware of and compensate for the fact that some browsers will not support some aspects of HTML5.

We have addressed this in general terms in previous chapters, but in this chapter, we will examine exactly how this plays out in relation to HTML5 layout elements. Moreover, we will inspect and learn from how the HTML5 layout elements in Dreamweaver address compatibility problems.

Which browsers support HTML5 layout elements?

As noted, the list of browsers that support HTML5 layout elements that we are exploring in this chapter is growing. Currently (2011) browser environments using the Webkit engine are the most fully compliant. Those browsing environments include Google Chrome, Safari (both the desktop and the mobile versions), Dreamweaver CS4 and CS5's own Live view, the Android mobile browser, and the Palm webOS browser.

Firefox, Opera, and Internet Explorer do not completely support HTML5 layout elements. Overwhelmingly, with exceptions, we will note in the course of this chapter, those compatibility issues are resolved fairly simply by relying on CSS styles that provide these browsers with all the information they need to present the content in HTML5 elements as it should appear.

Using CSS to solve browser issues

We will examine an example of a compatibility issue with the HTML5 layout elements in Dreamweaver's layouts, and how to solve it. Moreover, in the process, pick up some valuable trouble-shooting and problem solving methods.

For More Information:

Customizing HTML5 Layout Elements

The case in point can be illustrated by looking at the `<header>` element that is used in both of the HTML5 layouts. With either of the HTML5 layouts open in the Design view, the header, with its default *kaki-ish green* background appears at the top of the page. As an HTML element, it occupies a full horizontal swath across the top of the page where it has been placed.

However, browsers such as Firefox and Internet Explorer do not recognize `<header>` as an element. They can appreciate that there is something called a "header" on the page, and they can associate attributes (like that green kaki-ish background) with that object. However, as they don't appreciate that `<header>` is an element, they don't reserve an entire horizontal line for the content.

If we look at the CSS Styles panel for the CSS file that comes with either of the Dreamweaver HTML5 layouts, then we can see that the background color (defined by default with the #ADB96E hexadecimal code) is assigned to the `<header>` element as shown in the following screenshot:

The magic of display:block

In addition to a background color, the `<header>` element has another CSS attribute, which is assigned to it and all the rest of the HTML5 layout elements in the HTML5 layouts. That is found in the CSS Styles panel by examining the CSS rules for `header`, `section`, `footer`, `aside`, `nav`, `article`, `figure`, as shown in the following screenshot:

For More Information:

The `display:block` property assigned to all these HTML5 layout elements in the CSS Style sheet linked to the page instructs non-HTML5 browsers that this box of content should be handled as a block—it should fill the entire horizontal row in which it is placed.

### Disabling a CSS rule

We can see how this works by disabling the `display:block` property in the CSS Styles panel for the set of HTML5 layout elements. In Dreamweaver CS5.5 (and CS5), we can temporarily disable any defined CSS attribute by clicking to the left of it, as shown in the following screenshot:

For More Information:

Clicking to the left of an attribute in the CSS Styles panel disables the attribute by turning it into a comment in the CSS code (indicated by /*)). Such disabled attributes display with the international "no" symbol in the CSS styles panel. Both the CSS code and the CSS Styles panel with display:block disabled are shown in the following screenshot:

```
/* HTML 5 support - Sets new HTML 5 tags to display:block so browsers know how to render the tags properly. */
header, section, footer, aside, nav, article, figure {
    [disabled]display: block; /*
}
```

### Disabling CSS Attributes
The technique of temporarily disabling CSS settings in the CSS Styles panel has a wide applicability in testing web page layouts and in experimenting with different looks for web pages.

We can see how with the CSS style display:block disabled (for the moment), the header formatting collapses when the page is viewed in the Live view in the Document window. Moreover, it collapses when viewed in Firefox. If you have Firefox installed on your computer, then you can test this by selecting File | Preview in Browser | Firefox. Alternatively, preview in Internet Explorer. In either case, the header does not appear at all, as it is supposed to, as shown in the following screenshot:

**For More Information:**
By contrast, even with `display:block` disabled in the CSS Styles panel, the header displays as it is supposed to in Safari, as shown in the following screenshot:

![Instructions](image1.png)

**Instructions**

**How to use this document**

Be aware that the CSS for these layouts is heavily commented. If you do most of your work in Design view, have a peek at the code to get tips on working with the CSS for the fixed layouts. You can remove these comments before you launch your site. To learn more about the techniques used in these CSS Layouts, read this article at Adobe’s Developer Center - [http://www.adobe.com/go/adc_css_layouts](http://www.adobe.com/go/adc_css_layouts).

By the way, if you tried this at home, you may have noticed that the footer content also collapsed in Firefox, Internet Explorer, or other non-HTML5 browsers.

Again, the reason that the `<header>` element is handled correctly in Safari and not in Internet Explorer or Firefox is that we have disabled the CSS style that helps those non-HTML5 compliant browsers understand how to handle the header content.

If we re-enable the `display:block` attribute for all the HTML5 layout elements (by clicking again to toggle the international "no" sign off), then the header and footer content and formatting appears as intended in non-HTML5 browsers, as shown in the following screenshot:

![Instructions](image2.png)

**Instructions**

**How to use this document**

Be aware that the CSS for these layouts is heavily commented. If you do most of your work in Design view, have a peek at the code to get tips on working with the CSS for the fixed layouts. You can remove these comments before you launch your site. To learn more about the techniques used in these CSS Layouts, read this article at Adobe’s Developer Center - [http://www.adobe.com/go/adc_css_layouts](http://www.adobe.com/go/adc_css_layouts).

For More Information:

Global HTML5 layout element attributes

We just examined how the Dreamweaver HTML5 layouts adjust for non-HTML5 compliant browsers with supplemental CSS formatting. However, in the process, we identified another key piece of the puzzle of understanding how HTML5 elements acquire formatting attributes—this is done by defining CSS styles that are associated with those elements.

In order to move on and apply that technique to formatting all the HTML5 elements on our page, it will be useful to explore the following three essential CSS attributes that can be applied to nearly all the HTML5 layout elements:

1. Backgrounds
2. Size, margins, and padding
3. Positioning

While there are many attributes that can be assigned to layout elements, a conceptual handle on these three will prepare you with a foundation from which to dissect, learn from, and be able to adapt Dreamweaver's HTML5 layouts.

Working with backgrounds

The rules for `<header>`, `<section>`, `<footer>`, `<aside>`, `<nav>`, and `<article>` design elements all, at times, have backgrounds applied to them. While an examination of aesthetic and accessibility aspects of applying backgrounds to design elements is beyond the scope of this book, it would be irresponsible not to note them at all.

Very briefly, colors (and even more so images) can pose serious accessibility issues. One of the most widely noted accessibility problems for websites is the high percentage of people, whose color-blindness makes it impossible to distinguish text when placed on top of various colors, including when those colors appear in photos or other images or artwork. If accessibility is an issue for your web pages, then do sufficient research into dos and don'ts for mixing background colors and foreground text. Inexplicably, The Dreamweaver accessibility validation report feature has been deprecated as of Dreamweaver CS5. However, you can find multiple sites online that test your site for color accessibility issues.

While less potentially troublesome, aesthetic concerns are obviously something to consider when defining colors (or patterns) for layout elements. You will find valuable resources for creating color schemes at Adobe's Kuler site (http://kuler.adobe.com/) shown in the following screenshot:
With that ultra-compressed survey of aesthetic and accessibility issues down the hatch, let’s look at how to change the background color of any HTML5 layout element in the CSS Styles panel.

The most reliable and flexible way to change the background color is to open the CSS Rule definition dialog for any selected style using the Edit Rule icon at the bottom of the CSS Styles panel. For example, in order to open the CSS Rule definition dialog for the style associated with the `<footer>` element, select that element in the CSS Styles panel, and click on the Edit Rule... icon, as shown in the following screenshot:

For More Information:
Customizing HTML5 Layout Elements

In the **CSS Rule definition** dialog, the background category displays options for defining color and pattern background. Colors are selected from the background color picker. You can enter a hexadecimal value in the **Background-color box**, or use the eyedropper tool to select a color from the palette that appears when you click on the color picker box.

Alternatively, you can select a background image using the **Browse** button to navigate to and select that image from your computer or a connected drive. By default, background images tile, that is, they repeat horizontally and vertically to fill the assigned space. In the following screenshot, a tiling image has been applied to the `<header>` element and clicking on the **Apply** button displays how that tiled image will appear when applied:

![CSS Rule Definition for header in HTML5_twoColFixRhs.css](image)

**Defining size, margins, and padding**

If you think about translating a design to a webpage, the heart of the process is to create some kind of positioning containers to hold the content. Those containers are defined essentially by their size and location.

However, here things get complicated. First, positioning on the Web is positioning for a moving target. A layout element that is inviting and accessible on a laptop may be hidden and uninviting on an iPhone. We will address that challenge in *Chapter 5, Defining and Implementing Multiscreen Previews and Media Queries*.

Typically, both page size and position are defined by one, master container, often with an element name like "container." The page content is typically centered by creating a single such container whose main function is to constrain the size and positioning of all the page content. We will explore how this is done briefly here, and shortly in more detail when we dissect one of the HTML5 layouts.

For More Information:
Assigning a page size

The size of a web page is defined, as alluded to earlier, by the master container that holds all the page content. In recent years, a standard page size of 960 pixels has emerged. This size fits in and works well with both laptops and popular handheld devices.

The 960 pixel wide page also has mathematical advantages. It can be divided into as many as 16 evenly sized columns. This allows web designers to collaborate easily with graphic designers who can layout pages in programs such as Adobe Illustrator, Adobe Flash Catalyst, or even programs designed for other purposes such as Adobe Photoshop with which some designers might be more comfortable. As long as everyone in the workflow is using a 960 pixel wide grid, designs can be translated from wireframes and design files to HTML5 page layouts.

The following screenshot shows a container <div> generated by a Dreamweaver HTML5 layout with a defined width of 960 pixels:

Margins and padding are also part of positioning elements on the page and we will explore that next.

Margins versus padding

Margins define the space between any layout element and other content that element abuts. So, for example, if a vertical margin of 10 pixels is defined for a <header> element, then a space of 10 pixels high will be generated under the header, before any content that follows it.

Padding defines a space within an element, between the edges of an element, and the content within it.

For More Information:
Customizing HTML5 Layout Elements

Sometimes, either a margin or padding produces the same effect. At other times, the effect is more subtle. For example, using a margin combined with a border to separate content from other content will place space outside the border, while using padding to create space within the same element and border would create space between element content and the border.

Margins play a particular role in centering page content. Assigning a margin of auto to an HTML element used as a page container, as shown in the previous screenshot, is the standard technique for centering page content within a browser window.

Element padding versus content margins

One specific area where two approaches to spacing can be used is in creating space between the content of a layout element and the border of the element (or the outer edges of the element, whether or not a visible border is displayed).

As noted earlier, this spacing can be created with padding between the content (such as text) and the edge of the element. The same spacing can be created, alternatively, by assigning margins to the content itself. Therefore, for example, a 10-pixel margin applied to a text within an element creates more or less the same result as 10 pixels of padding applied to the element itself.

When to use which? This is often a decision involving more art, convention, and custom than anything else.

Positioning with float

As noted earlier, the page content is macro-positioned by defining the margin of the container holding all page content. Within that container, the content is often best positioned by using the float attribute.

In the following screenshot, for example, the .sidebar1 element used in the Dreamweaver HTML5 layouts includes float settings that align it on the left-hand side of the page (in the 3-column layout) or the right-hand side (in the 2-column layout):

For More Information:
Customizing HTML5 elements

As a final referential segue before diving into the HTML5 elements used in the Dreamweaver layouts, it will be helpful to identify particular design and formatting issues specific to the `<header>`, `<section>`, `<footer>`, `<aside>`, `<nav>`, and `<article>` elements.

For the `<header>` and `<footer>`, it will often be the case that you want to define a customized background, along with tweaked padding.

The formatting for `<section>` and `<article>` elements is often inherited from parent elements, that is, the global HTML tags such as the `<body>` tag, or enclosing tags such as a container tag, discussed earlier. However, within that, `<section>` and `<article>` elements might well have unique margin or padding, and specific formatting applied to text only within those elements.

The `<nav>` element, containing links, might well have a specific link formatting (colors, underlining, and so on). The `<aside>` element will have either a float attribute assigned to it or will inherit a float (right or left) from an enclosing `<div>` tag.

For More Information:
Customizing HTML5 Layout Elements

Back to the future: ID and class styles

Up to this point, we have alluded to two possible, different approaches to page design with HTML5 elements. The first is to assign actual layout attributes (size, position, float, and so on) to HTML5 elements. This is a basic, simple, intuitive, and potentially very powerful way to use HTML5 elements to design pages.

When we build HTML5 pages from scratch in the next chapter, we will use that approach. Moreover, as we will see then, such an approach has specific applicability for HTML5-friendly environments such as iPhones, iPod Touchs, and iPads.

However, that is not the approach taken in the Dreamweaver layouts, and for arguably valid reasons. In those layouts, much (not all) of formatting and positioning is defined by old-fashioned <div> tags that envelope (surround, and supply inherited formatting and positioning to) the HTML5 elements.

There are three basic advantages to this approach:

1. They solve compatibility issues by rendering HTML5 elements supplemental to page design.
2. They allow designers who have not yet acclimated to HTML5 elements to work with familiar techniques.
3. They "plan ahead" for dimensions of HTML5 elements that are works in progress, including the use of HTML5 elements in indexing the page content for search engines.

So, prepare yourself to find this approach applied as we examine the HTML5 layouts. We will look more specifically at the ubiquitous <div> tags, both with class and ID styles applied.

Layout with class styles

Class styles are the wildcard of web design. They can be used multiple times within a single document. Moreover, multiple class styles can be applied to a single element.

Let's walk through both these dimensions of class styles. As they can be used multiple times on a page, class styles have traditionally been used for layout elements such as boxes for pictures and captions, sidebar text boxes, boxes for media, and so on (elements that are often used more than once on a page).

For More Information:
Moreover, because multiple class styles can be applied to a single element and used repeatedly on a page, they are often used to "batch" formatting attributes that are then applied to more than one element.

**The role of ID styles**

ID styles are applied only once on a page, to a single element. They can and often are used to define `<div>` tags used for page layout, like a container `<div>`.

The big advantage of using ID styles instead of class styles for formatting is that the DIV tags with ID styles are programmable. Other books I have written on web and interactive design with Adobe Creative Suite explore programming interactive elements (elements that respond to visitor actions) in detail by generating JavaScript. That won't be a focus of this book, but you should be aware that this is a consideration in choosing which type of `<div>` tags to use in formatting.

**Recipe: Customizing the HTML5 page layout**

In the course of customizing the HTML5 layout elements that are generated with Dreamweaver, we learned to control foundational components of modern web page design.

In this recipe, we will focus on formatting HTML5 layout elements generated with the 2-column HTML 5 layout. Therefore, we will work with the generic content supplied with the layouts (for a systematic recipe for customizing the content, see the previous chapter).

Our recipe here, as promised, is somewhat extensive and is divided into two parts. The first part focuses on defining the position and size of the `<header>`, `<footer>`, `<nav>`, and `<aside>` elements. The second deals with changing the way links are displayed in the `<nav>` element.

In order to complete this recipe, you will need to have a few things in place. One is a defined Dreamweaver site. This is necessary and if you are not positive that you have defined and are working within a Dreamweaver site, please review the discussion of that in Chapter 1.
Recipe: Customizing the size and position for header, footer, nav, and aside

With a site defined, and open, the following steps walk through the process of inspecting and customizing the key layout elements in the HTML5 2-column layout:

1. Generate a page by choosing File | New. In the New Document dialog, choose the Blank Page category. Select the HTML5: 2 column fixed, right sidebar, header and footer layout. Note that the DocType is HTML5 by default and the Layout CSS is set to Create New File. Click on the Create button.

2. The Save Style Sheet File As dialog opens. Change the name of the saved CSS file to 2col.css, as shown in the following screenshot, and click on Save:

3. Save the file as 2col.html, and add a title, "2 column layout."

4. Examine the <body> tag definition in the CSS Styles panel (as shown in the following screenshot). While the <body> tag is not an HTML5 layout element, HTML5 elements will inherit attributes assigned to this tag unless the HTML5 elements, or CSS styles attached to them, overwrite those attributes. So, note that the background color, font (including 1.4 line spacing, which creates 40% extra vertical spacing between lines of type), font color, padding, and margins are defined by the <body> tag:

5. The second global defining element is the .container style. Examine it in the Global to note the background color (#FFF=gray), the margin (the operative attribute being auto (for left and right), which centers the container), and the 960 pixel width, which adheres to current design standards.

For More Information:
6. Now we will change the alignment of the `<nav>` and `<aside>` elements from the right-hand side of the page to the left. As discussed in the chapter, the attributes for these elements are defined in an enclosing class style, not the elements themselves. Click anywhere within either the layout links or the sidebar content below them. In the tag bar at the bottom of the Document window, click on the `div.sidebar1` tag in the tag bar to see elements defined by that style highlighted in the Design view of the Document window, as shown in the following screenshot:

![Design view of Document window showing elements defined by style](image)

7. In the CSS Styles panel, change the Float for the `.sidebar1` style from right to left, as shown in the following screenshot:

![CSS Styles panel showing Float property](image)

For More Information:

8. Next, we will change the size and background of the page header by editing the `<header>` element. First, click on the **Insert_logo (180 x 90)** box and press your **Delete** key. Customized sites created in Dreamweaver should have customized header content not generically sized and placed logos that stamp a page as being constructed using out-of-the-box templates from a blogging app like WordPress. Without the placeholder image to expand the height of the header, we will define our own header height. In the CSS styles panel, click on the **Add Property** link and enter height in the first column and a value of 100 px, as shown in the following screenshot:

9. Examine the style attributes for the `<footer>` element. The height of the footer is undefined—the footer expands (or contracts) to accommodate the content placed within it. Defining a fixed height for the footer can be done in the same way we just defined a height for the header. No width definition is necessary, as the footer expands to fill the space available. The **position: relative** attribute backs up the **clear:both** code for Internet Explorer 6. While the position attribute is only necessary to present the page correctly in IE6, the clear attribute is essential in any environment—it keeps the page from collapsing—by "turning off" any float attributes that are inherited from other page elements. In order to see for yourself how this works, click to the left of the clear attribute for the footer style in the CSS Styles panel to disable it (as shown in the following screenshot) and examine the disastrous result this has on how the different page elements fit together. Then, of course, click again to enable the clear attributes and restore the page.

For More Information:

www.packtpub.com/dreamweaver -cs5-5-mobile-and-web-development-with-
html5-css3-jquery/book
Customizing links in the nav element

In examining and changing the appearance of content in the nav element, we will have a chance to dig rather deeply into how CSS styles are applied to specific tags within an element. In this case, we are going to solve the challenge of turning off the default underlining that appears in the <nav> element for unvisited links. The documentation that comes with the layout justifies this underlining as necessary to make it clear these are links. For the purposes of posing an instructive resolution to that, we will maintain underlining for links elsewhere on the page, but redefine links within the <nav> element.
Customizing HTML5 Layout Elements

1. As defined by the HTML5 layout, unvisited links throughout the page are displayed with underlining, while visited links, hovered links, active links (that display while a link is being selected), and focused links (links clicked in devices where hovering is inoperative, such as mobile devices) display with no underlining. There are several ways to test this, but the most effective way is to view the Style Rendering toolbar (View | Toolbars | Style Rendering). Use the link, visited (shown in the following screenshot), hover, active, and focus buttons to see how links display in each of these modes. Note that in the visited mode, links display with underlining outside the <nav> element, but without underlining within the <nav> element. Our mission in this second recipe is to identify why that is, and change the formatting, so that unvisited links inside the <nav> element also display without underlining, while allowing links outside the <nav> element to continue to display with underlining:

2. There are almost countless approaches we could take to identifying why, within the <nav> element, unvisited links are underlined, while visited links are not. However, often a good place to start on detective work such as this is the Current tab of the CSS Styles panel. With one of the links selected in the <nav> element and the link selected in the Style Rendering toolbar, note that the underlining style for links is defined in the style attached to a:link, as shown in the following screenshot:

For More Information:
3. Now, let’s compare the style rules in effect for unvisited links within the `<nav>` element, to the rules in effect for visited links. Select the :v (visited) button in the Style Rendering toolbar and note that here, a different, specific rule is in effect defining `text-decoration:none` for visited links (see the following screenshot). This rule has been defined to apply to visited links (`a:visited`) within an unordered list (`ul`) within a `<nav>` element:

Unordered lists: We will explore the generalized implementation of unordered lists (aka bullet lists) in defining and styling menus in `<nav>` elements in the next chapter of this book, when we build pages from scratch relying on HTML5 layout elements.

For More Information:
4. To sum up what we have learned from this investigation so far: the style sheet provided with the HTML5 layout includes specific styles that turn off underlining within the unordered list in the `<nav>` element for visited links. If we did further, similar investigation, we would learn as well that underlining is turned off for other link states (hovered, active, and focused) throughout the page. Therefore, our mission is to duplicate the CSS rule that turns off underlining for visited links in the `<nav>` element, but tweak the duplicate, so that it applies to unvisited links as well. There are many ways to do that; for example, we could go into the code for the CSS file and copy, paste, and edit. Alternatively, we could start from scratch and create a new CSS rule, jumping back and forth to remember what attributes we want to steal from the rule for visited links in the `<nav>` element. The first option is too code-centric, the second too tedious. So instead, we can switch to the All tab in the panel and select the rule `nav ul a, nav ul a:visited`. With that rule selected, right-click (Windows) or Control-click (Mac), and choose Duplicate from the context menu, as shown in the following screenshot:

5. Selecting Duplicate opened the Duplicate CSS Rule dialog. Edit the selector name to `nav ul a, nav ul a:link` as shown in the following screenshot. By doing this, the new (duplicate) rule will apply to links within the same parameters as the original (links in the nav element, in unordered lists):

For More Information:
6. Verify the results of this recipe by examining the new style in the CSS Styles panel: `nav ul a, nav ul a:link`. With a regular link selected in the Style Rendering toolbar, note that unvisited links in the `<nav>` element no longer display with underlining, while unvisited links elsewhere (such as the link to adobe in the sample text) do display with underlining, as shown in the following screenshot:

For More Information:
Customizing HTML5 Layout Elements

Tracking down the source of the CSS definitions for a link displayed within the `<nav>` element was a good experience. We actually solved what I consider one of the most difficult problems to solve in changing Dreamweaver HTML5 layouts. Moreover, beyond that, we explored how formatting can be applied within an HTML5 element through multiple layers of CSS styles.

Summary

In the course of dissecting and customizing the size, position, background, and other attributes of HTML5 layout elements generated with the HTML5 Pack, we learned to format the key design components of these foundational elements of a modern web page design.

In addition, we prepared ourselves to construct web pages from scratch using HTML5 elements using truly next generation design techniques. We will do that in the next chapter.
Where to buy this book


Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our shipping policy.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.