**Reading 1: Understand CSS page layout basics**

Learn the basics of CSS page layout functions which use cascading style sheet format, rather than traditional HTML tables or frames, to organize the content on a web page.

The basic building block of the CSS layout is the div tag—an HTML tag that in most cases acts as a container for text, images, and other page elements.

When you create a CSS layout, you place div tags on the page, add content to them, and position them in various places. Unlike table cells, which are restricted to existing somewhere within the rows and columns of a table, div tags can appear anywhere on a web page. You can position div tags absolutely (by specifying x and y coordinates), or relatively (by specifying their distance from other page elements).

Creating CSS layouts from scratch can be difficult because there are so many ways to do it. You can create a simple two-column CSS layout by setting floats, margins, paddings, and other CSS properties in a nearly infinite number of combinations. Additionally, the problem of cross-browser rendering causes certain CSS layouts to display properly in some browsers and display improperly in others. Dreamweaver makes it easy for you to build pages with CSS layouts by providing some pre-designed layouts that work across different browsers.

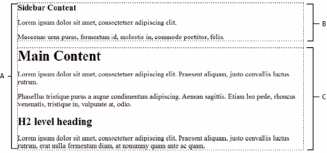
Using the pre-designed CSS layouts that come with Dreamweaver is the easiest way to create a page with a CSS layout, but you can also create CSS layouts using Dreamweaver absolutely positioned elements (AP elements). An AP element in Dreamweaver is an HTML page element—specifically, a div tag, or any other tag—that has an absolute position assigned to it. The limitation of Dreamweaver AP elements, however, is that since they are absolutely positioned, their positions never adjust on the page according to the size of the browser window.

If you are an advanced user, you can also insert div tags manually and apply CSS positioning styles to them to create page layouts.

**About CSS page layout structure**

Before proceeding with this section, you should be familiar with basic CSS concepts.

The basic building block of the CSS layout is the div tag—an HTML tag that in most cases acts as a container for text, images, and other page elements. Figure 1 shows an HTML page that contains three separate div tags: one large "container" tag, and two other tags—a sidebar tag, and a main content tag—within the container tag.



**Figure 1**. A. Container div B. Sidebar div C. Main Content div.

Here is the code for all three div tags in the HTML:

<!--container div tag-->

<div id="container">

  <!--sidebar div tag-->

  <div id="sidebar">

    <h3>Sidebar Content</h3>

    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>

    <p>Maecenas urna purus, fermentum id, molestie in, commodo porttitor, felis.</p>

  </div>

  <!--mainContent div tag-->

  <div id="mainContent">

    <h1>Main Content</h1>

    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Praesent aliquam, justo convallis

      luctus rutrum.</p>

    <p>Phasellus tristique purus a augue condimentum adipiscing. Aenean sagittis.

      Etiam leo pede, rhoncus venenatis, tristique in, vulputate at, odio.</p>

    <h2>H2 level heading</h2>

    <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Praesent aliquam, justo convallis

      luctus rutrum, erat nulla fermentum diam, at nonummy quam  ante ac quam.</p>

  </div>

</div>

In the above example, there is no “styling” attached to any of the div tags. Without CSS rules defined, each div tag and its contents fall into a default location on the page. However, if each div tag has a unique id (as in the above example), you can use the ids to create CSS rules that, when applied, change the style and positioning of the div tags.

The following CSS rule, which can reside in the head of the document or in an external CSS file, creates styling rules for the first, div tag on the page, known as or container div tag:

#container {

  width: 780px;

  background: #FFFFFF;

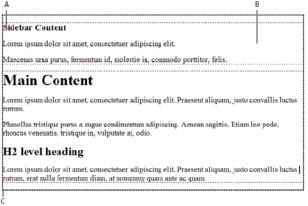
  margin: 0 auto;

  border: 1px solid #000000;

  text-align: left;

}

The #container rule styles the container div tag to have a width of 780 pixels, a white background, no margin (from the left side of the page), a solid, black, 1-pixel border, and text that is aligned left. Figure 2 shows the results of applying the rule to the container div tag.



**Figure 2**. Container div tag, 780 pixels, no margin A. Text aligned left B. White background C. 1-pixel solid black border.

The next CSS rule creates styling rules for the sidebar div tag:

#sidebar {

  float: left;

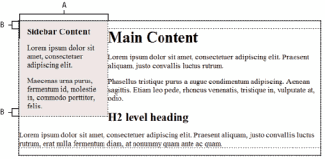
  width: 200px;

  background: #EBEBEB;

  padding: 15px 10px 15px 20px;

}

The #sidebar rule styles the sidebar div tag to have a width of 200 pixels, a grey background, a top and bottom padding of 15 pixels, a right padding of 10 pixels, and a left padding of 20 pixels. (The default order for padding is top-right-bottom-left.) Additionally, the rule positions the sidebar div tag with float left—a property that pushes the sidebar div tag to the left side of the container div tag. Figure 3 shows the results of applying the rule to the sidebar div tag.



**Figure 3**. Sidebar div, float left A. Width 200 pixels B. Top and bottom padding, 15 pixels.

Last, the CSS rule for the main container div tag finishes the layout:

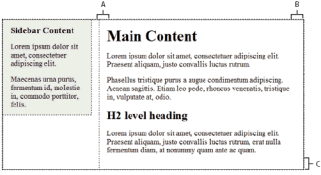
#mainContent {

  margin: 0 0 0 250px;

  padding: 0 20px 20px 20px;

}

The #mainContent rule styles the main content div with a left margin of 250 pixels, which means that it places 250 pixels of space between the left side of the container div, and the left side of the main content div. Additionally, the rule provides for 20 pixels of spacing on the right, bottom, and left sides of the main content div. Figure 4 shows the results of applying the rule to the mainContent div.



**Figure 4**. Main Content div, left margin of 250 pixels A. 20 pixels left padding B. 20 pixels right padding C. 20 pixels bottom padding

The complete code looks as follows:

<head>

  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

  <title>Untitled Document</title>

  <style type="text/css">

    #container {

      width: 780px;

      background: #FFFFFF;

      margin: 0 auto;

      border: 1px solid #000000;

      text-align: left;

    }

    #sidebar {

      float: left;

      width: 200px;

      background: #EBEBEB;

      padding: 15px 10px 15px 20px;

    }

    #mainContent {

      margin: 0 0 0 250px;

      padding: 0 20px 20px 20px;

    }

  </style>

</head>

<body>

  <!--container div tag-->

  <div id="container">

    <!--sidebar div tag-->

    <div id="sidebar">

      <h3>Sidebar Content</h3>

      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit.</p>

      <p>Maecenas urna purus, fermentum id, molestie in, commodo porttitor, felis.</p>

    </div>

    <!--mainContent div tag-->

    <div id="mainContent">

      <h1>Main Content</h1>

      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Praesent aliquam, justo convallis

        luctus rutrum.</p>

      <p>Phasellus tristique purus a augue condimentum adipiscing. Aenean sagittis.

        Etiam leo pede, rhoncus venenatis, tristique in, vulputate at, odio.</p>

      <h2>H2 level heading</h2>

      <p>Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Praesent aliquam, justo convallis

      luctus rutrum, erat nulla fermentum diam, at nonummy quam  ante ac quam.</p>

    </div>

  </div>

</body>

Note: The above example code is a simplified version of the code that creates the two-column fixed left sidebar layout when you create a new document using the predesigned layouts that come with Dreamweaver.

**Create a page with a CSS layout**

When creating a new page in Dreamweaver, you can create one that already contains a CSS layout. Dreamweaver comes with different CSS layouts that you can choose from. Additionally, you can create your own CSS layouts and add them to the configuration folder so that they appear as layout choices in the New Document dialog box.

Dreamweaver CSS layouts render correctly in all modern browsers. To create a page with a CSS layout:

* Select File > New.
* In the New Document dialog box, select the Blank Page category. (It's the default selection.)
* For Page Type, select the kind of page you want to create.

Note: You must select an HTML page type for the layout. For example, you can select HTML or PHP. You cannot create a CSS, Library Item, JavaScript, or XML with a CSS layout. Page types in the Other category of the New Document dialog box are also restricted from including CSS page layouts.

For Layout, select the CSS layout you want to use. The Preview window shows the layout and gives a brief description of the selected layout. The predesigned CSS layouts provide the following types of columns:

* Fixed Column width is specified in pixels. The column does not resize based on the size of the browser or the site visitor's text settings.
* Liquid Column width is specified as a percentage of the site visitor's browser width. The design adapts if the site visitor makes the browser wider or narrower but does not change based on the site visitor's text settings.
* Select a document type from the DocType pop‑up menu.
* Select a location for the layout's CSS from the Layout CSS in pop‑up menu.
* Add To Head: Adds CSS for the layout to the head of the page you're creating.
* Create New File: Adds CSS for the layout to a new external CSS stylesheet and attaches the new stylesheet to the page you're creating.
* Link to Existing File: Lets you specify an existing CSS file that already contains the CSS rules needed for the layout. This option is particularly useful when you want to use the same CSS layout (the CSS rules for which are contained in a single file) across multiple documents.

Do one of the following:

* If you selected Add to Head from the Layout CSS in pop‑up menu (the default option), click Create.
* If you selected Create New File from the Layout CSS pop‑up menu, click Create, and then specify a name for the new external file in the Save Style Sheet File As dialog box.
* If you selected Link to Existing File from the Layout CSS in pop‑up menu, add the external file to the Attach CSS file text box by clicking the Add Style Sheet icon, completing the Attach External Style Sheet dialog box, and clicking OK. When you're finished, click Create in the New Document dialog box.
* Note: When you select the Link to Existing File option, the file you specify must already have the rules for the CSS file contained within it.
* When you put the layout CSS in a new file or link to an existing file, Dreamweaver automatically links the file to the HTML page you're creating.
* Note: Internet Explorer conditional comments (CCs), which help work around IE rendering issues, remain embedded in the head of the new CSS layout document, even if you select New External File or Existing External File as the location for your layout CSS.
* (Optional) You can also attach CSS style sheets to your new page (unrelated to the CSS layout) when you create the page. To do this, click the Attach Style Sheet icon above the Attach CSS file pane and select a CSS style sheet. (For a detailed walk-through of this process, see David Powers's article, [Automatically attaching a style sheet to new documents](http://kb2.adobe.com/community/publishing/505/cpsid_50564.html).

**Add custom CSS layouts to the list of choices**

* Create an HTML page that contains the CSS layout you'd like to add to the list of choices in the New Document dialog box. The CSS for the layout must reside in the head of the HTML page.
* Tip: To make your custom CSS layout consistent with the other layouts that come with Dreamweaver, you should save your HTML file with the .html extension.
* Add the HTML page to the dw\_root\Configuration\BuiltIn\Layouts folder.
* (Optional) Add a preview image of your layout (for example a .gif or .png file) to the dw\_root\Configuration\BuiltIn\Layouts folder. The default images that come with Dreamweaver are 227 pixels wide x 193 pixels high PNG files.

**Tip**: Give your preview image the same file name as your HTML file so that you can easily keep track of it. For example, if your HTML file is called myCustomLayout.html, call your preview image myCustomLayout.png.