Making the View: Pt 2

Posted At : July 10, 2006 11:25 PM | Posted By : Cutter
Related Categories: Development, ColdFusion, Making the View

Ok, before we continue lets's pick apart our template a little.

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

<html xmlns="http://www.w3.org/1999/xhtml">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
  <title>Untitled Document</title>
 </head>

 <body>

 </body>
</html>
```

We defined the doctype of our page as a transitional XHTML 1.0 document with english as the language. The doctype declaration is a required element of any well-formed XHTML page, and references the appropriate DTD (document type definition) for the content of the document.

From **W3Schools**:

> DTD is used by SGML applications, such as HTML, to specify rules that apply to the markup of documents of a particular type, including a set of element and entity declarations.

There are three types of XHTML document: strict, transitional, and frameset. Here we use the transitional type, because it gives us a little more leeway for minor error.

Well-formed XHTML documents (like HTML) will also include, at minimum, the HTML, HEAD, TITLE, and BODY tags. Within the HTML tag we also define an XML NameSpace (xmlns) for the document. XHTML is meant as a transitional layer between HTML and XML for web developers. A way to get out of our bad habits, such as the often malformed HTML of the past, to properly structured documents, like those needed for XML.

Again from **W3Schools**:

> XML Namespaces provide a method to avoid element name conflicts... Since element names in XML are not predefined, a name conflict will occur when two different documents use the same element names.

We've also included a META tag to define the content type of the document, and the character set used. Oh yeah, one last (important) thing. Tags and attributes in an XHTML document are always lower case.

OK class, that's a little background on what little we have so far. Let's get on to the meat and potatoes of this tutorial. Style!

Cascading Style Sheets have been around for some time now. The concept is simple, separate style from content. This is great, because, by defining a global style sheet

for a site, every page can maintain the same basic look and feel. From a marketing and branding perspective this is extremely important, but even more so from a usability stand point. A consistency of design makes it easier on your users when navigating through the sites you create. Styles can be implemented in several different ways, but most of your files will be within an external stylesheet. You can define more than one stylesheet within a page, but we're going to start off with a site wide default stylesheet for basic page display. Place the following tag within the <head> section of your document, after your title element:

```
<link href="css/default.css" rel="stylesheet" type="text/css" media="screen" lang="en" title="Default Site Stylesheet" />
```

OK, there are few things going on here. I have an href reference to the external stylesheet, default.css, which I have saved in my css folder (this location is relative to the document itself, so you may wish to use a dynamic variable here). I have several statements that this is a 'stylesheet', to be processed as a 'text/css' document when loaded, that the language used is english, and that the stylesheet is intended to style for the 'screen'. I also included a title to denote it from other stylesheets I might include later.

Next we'll create a basic stylesheet to start off with. For this, first create a folder relative to your document titled 'css', then create a new file within the folder, 'default.css'. First we'll place a comment to explain the document, then define it's character set and a few starter styles for our site.

```
/*   default.css
//   Created: 07/10/2006

//   Author:  Cutter

//   Purpose: A default site-wide stylesheet

*/

@CHARSET "ISO-8859-1";

body {
 text-align: center;
 font-family: Arial, Helvetica, sans-serif;
 font-size: 10pt;
 margin-top: 0px; }

td {
 font-family: Arial, Helvetica, sans-serif;
 font-size: 10pt; }
```

Alright, we set our character set to the same one we used in the meta tag of our document template, we said that we wanted the contents of the body to be aligned to the center (more on this in a minute), set the font family to be used, the font size (more on this in another session), and that we wanted no top margin.

"Wait a minute? Why'd Cutter do the font stuff for the td tag as well?" Well, it's a glitch in IE (and maybe Firefox, I've never checked) that your styles won't cascade down into your table cells. I don't know why, but that's the way it is.

Now, you may also be asking why I didn't place a margin on the left. Well, notice how I aligned everything to the center? That's because, before I place anything else into the document I'm going to add a container element, in between the body tags, to hold everything.

```
<div id="totalBody">
     <p>Here&rsquo;s the body</p>
</div>
```

Then we're going to add another declaration at the bottom of our stylesheet to define our container a little.

```
#totalBody {
     margin-left: auto;
     margin-right: auto;
     margin-top: 5px;
     width: 999px;
     text-align: left; }
```

Let me explain. I've set my left and right margins to auto, so as to center my container, with a top margin of only 5 pixels, a width of 999 pixels, and aligning all inside it to the left. I've started making most of my layouts for 1024 x 768 now. Web usage stats are finally showing this above the 70% mark (thank you flat screen monitors). You could do liquid layouts, but I'm not discussing those within these sessions. I use 999 pixels to account for a (roughly) 25 pixel wide scroll bar, if it's needed. If you don't like the width, change it. It's up to you.

OK, this session I'll post some example files from what we've discussed. Nothing overly fancy, but if you load up the index.cfm and go into your **Web Developer's Plugin** for **Firefox**, then select ***Outline > Outline Block Level Elements***, then select ***Information > Display ID & Class Details***, you'll start to get an idea of where you're going here. One step at a time...