



# Back to Basics: Structuring a Web Page with CSS and ASP.NET

28 January 2010

by *Nick Harrison*

Av rating: ★★★★★

Total votes: 135

Total comments: 16

✉ [send to a friend](#)

🖨 [printer friendly version](#)

Nick explains why such habits as using nested HTML Tables to position content in the right place on the browser page is bad practice and, nowadays, avoidable. This is just one 'Markup smell' that he discusses on the way to demonstrating the benefits of CSS Style-sheets and ASP.NET Master Pages.

## Introduction

Here we will go back to the basics and explore the finer points of structuring a web page. We will review some of the evolution of HTML and explore the reasoning behind some of these advances. We will explore those limitations at each step in the evolution that lead to the innovations for the next. Finally we will describe the steps to structure a web page that it will

- Be easy to maintain
- Comply with standards and best-practices,
- meet accessibility requirements
- support search engine optimization (SEO)

## From Hyperlinks to Tables to Style sheets to Master Pages

In the beginning, HTML was devised purely as a means of displaying content, but there was no control over the layout. You could make text bold, italicized, or underlined. You could make links, lists, and headers. In the early 1990s that was all you could do.

Then there were tables. Tables allowed you to control the way that content was positioned on a page. With tables, you could line up elements and you could structure content in to columns. You could specify all sorts of details that had nothing to do with the content itself. This started the beginning of HTML bloat. Soon, there was more page markup that had nothing to do with the content than there was actual content. This created a high “signal to noise ratio” for most web pages.

I was as guilty as any were of nesting tables to get the desired look. When your design relies on as many as eight levels of nested tables, more markup code is being used for layout and format than is being used for content.

Style sheets were introduced to help reduce the noise. With style sheets, all markup related to formatting can easily be moved from the page. This makes the content easier to find and edit. Style sheets also help make our pages more standards compliant, more easily maintained, and help bring the focus back to the content. Style sheets will also help us resolve some of the problems that came about with tables. The positioning and layout tricks that many people use tables for can be done much more cleanly and simply with style sheets.

Style sheets solve many problems, but they are not yet universally adopted. They also have a reputation for being more voodoo art than science. As a result, programmers often shy away from them. But fear not, they can be mastered or at least tamed and used for good.

Master pages were introduced to help improve consistency across a web application. “Master page” is perhaps a bit of a misnomer. A better name may have been page templates. The master page provides a template for how a page should look and provides content sections where individual page’s content should go. Master pages make it easier to have consistency across the pages in a site. Individual pages need only contain that page’s specific content. All structural markup code is in the master page. All common markup code is in the master page.

Content that is common across several pages can be defined once in a common master page. Items such as the header, footer, sidebar, etc can be defined in the master page and then are visible in every page using that master page. Each individual page will be easier to create because there are fewer items to worry about. By pushing common functionality and appearance to the master page, individual pages are easier to implement because they only have to focus on how they are different from the rest. By pushing all formatting and layout directives from the individual pages to the style sheets, we will have dramatically less markup in our individual pages. Because the same style sheet can be reused throughout the site, less webpage markup will need to be delivered down to the client. The reused style sheet needs to be downloaded only once and not once for each page in your site. This will improve load times, improve band width concerns, and lower hosting costs.

## So What is Wrong with Tables?

We mentioned earlier in this article that many of the advancements in HTML have come about as a response to try

## .NET Categories

.NET Home

ASP.Net

Windows Forms

.Net Framework

Performance

Visual Studio

.Net Tools

Editor's Corner

## Simply Web Dev

For web developers, by web developers

The best path to understanding ASP.NET MVC, with recommended reading and expert insight.

Find out more



## Custom RSS feeds

- ✔ SQL
- ✔ .NET
- ✔ SysAdmin
- ✔ Opinion
- ✔ Books
- ✔ Blogs

If you update your feed, please remember to tell your RSS reader the new URL

[Click here for advanced RSS options](#)

Get my feed

## Top Rated

**Acceptance Testing with FitNesse: Naming and Layout**

📖 Having dealt with Documentation and Infrastructure in the popular wiki-based acceptance-testing tool... [Read more...](#)

**Acceptance Testing with FitNesse: Documentation and Infrastructure**

📖 FitNesse is a popular general-purpose wiki-based framework for writing acceptance tests for software... [Read more...](#)

**Refactoring CSS with Sass and Compass**

📖 CSS is still a valuable way of specifying the rendered style of HTML objects. By using a preprocessor... [Read more...](#)

**TortoiseSVN and Subversion Cookbook Part 11: Subversion and Oracle**

and eliminate tables from our markup. So what is wrong with the use of tables? Making the case against tables is not always easy. They seem so natural and are so easy to understand that their problems are easily missed and not well understand.

HTML tables allow you to think of your page as a grid. It seems so easy to think about your page as simply adding content to this grid. The problem is that this grid has nothing to do with the content on your page. The tables require a substantial amount of markup without providing any extra value. This means that more data has to be transferred for your page to render properly. Your pages will load more slowly. Your bandwidth requirements will go up, and you get no added value for these extra hurdles.

This extra markup also gets in the way of automated (‘bot’) processes that need to understand your page. Search engines have to filter through meaningless table syntax to retrieve the content. Search engines will have more difficulty ranking your pages. The most common things that your page discusses may appear to be table data and table rows instead of the true content. Browsers that read the content to the viewer will get bogged down in meaningless details that are entailed in trying to understand the maze of table data. Mobile browsers will struggle with the process of rendering your page for alternate formats. Mobile browsers cannot easily render your page for optimal viewing because of the table definitions. Web browsers on widescreen monitors cannot adapt to take advantage of the extra space and hardware resources available.

The browser needs to render the page in the best way to take advantage of the wide screens and to adapt to small screens. To perform well in this, the goal is a page that has fluid design. Fluid design is difficult under the best circumstances. It is even harder with tables.

## Best Practices

With the exception of Master Pages, everything we will discuss here applies equally well regardless of your platform. These best practices and refactors all apply whether you are using ASP.Net, JSP, PHP, etc. Each platform provides its own mechanism for page templates. If you are not familiar with your platform’s solution, investigate it. You will be glad you did.

So what is the best way to structure our pages? How do we know that our structure is good? How do we improve upon what we have done in the past?

‘Refactoring’ code is a disciplined approach for changing existing code by modifying its internal structure without changing the external behavior. Refacoring includes techniques for brining code it in line with the best practices. The concept of refactoring html is a rather new discipline, but there is much that we can we adopt from the code-refactoring literature. Let’s borrow the term ‘smell’ to describe markup that may render satisfactorily but requires improvement. Some “Mark up Smells” with associated ‘refactors’ to consider include

Table Smell	<ul style="list-style-type: none"><li>Replace Table with Style sheet positioning</li><li>Replace Table with Explicit Width</li></ul>
Embedded Styling	<ul style="list-style-type: none"><li>Move Style Sheet to External Style Sheet</li></ul>
Embedded Formatting	<ul style="list-style-type: none"><li>Extract attributes to Style Sheet</li></ul>
Embedded Positioning	<ul style="list-style-type: none"><li>Replace Positioning with Style Sheet</li></ul>
Duplicate Markup	<ul style="list-style-type: none"><li>Move Redundant Markup to Master Page</li><li>Move Redundant Markup to User Control</li></ul>

Like code smells, markup smells imply that there may be a problem in your markup. Every time, you see one of these smells does not necessarily mean that you have a problem, but you do have something that needs to be investigated.

### Table Smell

If you see one table tag in you markup, there is probably not too much to worry about. If you see nested tables, you start smelly markup that needs to be cleaned up. You also need to be aware of why tables are being used. If the table is being used to display columns of data, you have a good use for your table. If your table is being used solely to put a border around a particular section, your table can easily be replaced with style sheet directives. If your table is being used to line up or structure content, it is time to consider streamlining your markup. Adding an explicit width to your style sheet classes will often suffice to make everything line up without having to resort to wrapping content in tables.

### Embedded Styling

Sometimes you will find a style sheet embedded directly in a page. This is easy to spot, if your mark up includes a style tag; you have an Embedded Styling smell. We may often embed the style sheet to make quick changes or to test a new style. We may also create an embedded style sheet as an intermediate step in “Extract attributes to Style Sheet” or “Replace Positioning with Style Sheet”. Embedding the style sheet streamlines the testing, but the last step needs to be “Move Style Sheet to External Style Sheet”. Even if this is a style sheet that only the one page will reference, move it to a separate file. This will keep all of the style directives together and result in cleaner markup code.

It is also important to view your page in various web browsers and at various resolutions and on various devices. If your page does not render well in a particular browser then you may be relying on browser specific markup. If your page does not render well on a smart phone or on a wide screen monitor, then you are probably using rigid constructions and need to move towards a more fluid design.

### Embedded Formatting

Embedded formatting occurs whenever you have formatting directives embedded directly in the element tags. This could be a style attribute on any element. This could also be a width attribute, any attribute specifying color, an align attribute, or any of the background attributes. The first step in cleaning up this

It is only recently that the tools have existed to make source-control easy for database developers.... [Read more...](#)

**TortoiseSVN and Subversion Cookbook Part 10: Extending the reach of Subversion**  
Subversion provides a good way of source-controlling a database, but many operations are best done from... [Read more...](#)

## Most Viewed

**A Complete URL Rewriting Solution for ASP.NET 2.0**  
Ever wondered whether it's possible to create neater URLs, free of bulky Query String parameters?... [Read more...](#)

**Visual Studio Setup - projects and custom actions**  
This article describes the kinds of custom actions that can be used in your Visual Studio setup project. [Read more...](#)

**.NET Application Architecture: the Data Access Layer**  
Find out how to design a robust data access layer for your .NET applications. [Read more...](#)

**Calling Cross Domain Web Services in AJAX**  
The latest craze for mashups involves making cross-domain calls to Web Services from APIs made publicly... [Read more...](#)

**Web Parts in ASP.NET 2.0**  
Most Web Parts implementations allow users to create a single portal page where they can personalize... [Read more...](#)

## Why Join

Over 400,000 Microsoft professionals subscribe to the Simple-Talk technical journal. Join today, it's fast, simple, free and secure.

[Join Simple-Talk!](#)

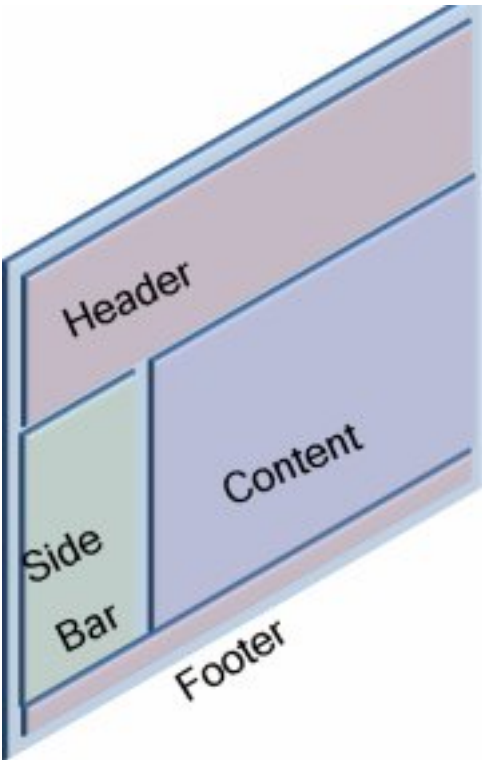
smell is to extract these attributes to an internal style sheet and then move that style sheet to an external style sheet.

## Embedded Positioning

The most common source of embedded positioning is using tables, but other practices are common. For example, you can easily create a very brittle layout with absolute positioning. When you see elements with style attributes that include left or top directives or a position directive with a value of absolute, you are relying on embedded positioning and it will lead to problems.

Tables are also used to explicitly provide formatting details. The problem here lies in the extra markup needed to make the positioning work.

## Where Do We Go From Here?



Consider a basic page layout like the one here. We have a header, footer, sidebar navigation, and a main content area. This can be defined in a Master Page and every page will easily share this common structure.

```
<body>
  <form id="form1" runat="server">
    <div id="container">
      <div id="header">
        <h4>Web Application Name</h4>
      </div>
      <div id="content">
        <asp:ScriptManager ID="script" runat="server">
        </asp:ScriptManager>
        <asp:ContentPlaceHolder ID="cphContent" runat="server">
        </asp:ContentPlaceHolder>
        <asp:ValidationSummary ID="valSummary" runat="server"
          DisplayMode="BulletList" />
        </div>
      <div id="sidePanel">
        <asp:ContentPlaceHolder ID="cphLeftNavigation" runat="server">
        </asp:ContentPlaceHolder>
      </div>
      <div id="footer">
        <p>Common Footer Content</p>
        <asp:ContentPlaceHolder ID="cphFooter" runat="server">
        </asp:ContentPlaceHolder>
      </div>
    </div>
  </form>
</body>
```

There are a couple of things worth emphasizing here. First, there are no tables. There are no formatting directives, and there are no positioning directives. Everything here is pure content. Note also that the main structural elements have an explicit ID attribute specified. Structural elements that can occur only once should be styled with an ID selector in the style sheet to further reinforce that the style directives are applicable for only a single element.

Also, you will have noticed the content place holders. These provide the configuration points where individual pages can provide custom content. In the markup provided, individual pages cannot alter the header. They can provide custom footer content but not override the common footer content. The individual pages have complete control over the content for the Content Area and the Side Bar area but do not need to worry about styling or positioning these structures.

The master page also includes the Script Manager and defines a Validation Summary control. These are elements that every page will most likely need. By defining these elements in the master page, the individual pages do not need to define them.

The layout and positioning for these structural elements may look similar to this:

```
#container
{
```



```
width: 673px;
border: solid 1px black;
margin: 0px auto;
}

#header
{
    color: #FFFFFF;
    width: 221px;    height: 100px;
    background-color: #778CB3;
}

#sidePanel
{
    width: 213px;    height: 372px;
    margin-top: 5px;
    background-color: #778CB3;
    color: #FFFFFF;    padding: 4px;
}

#content
{
    width: 446px;    float: right;
    height: 380px;    margin-top: 5px;
    background-color: #A0AFCB;
}

#footer
{
    background-color: #F7CB33;
    padding: 12px;    width: 649px;
    color: #000000; font-size: 90%;
    text-align: center;    clear: both;
    border-top: 5px solid #FFFFFF;
}
```

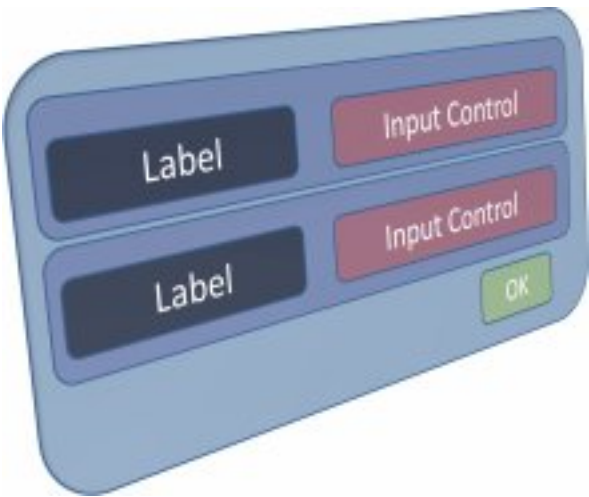
This is a suggestion rather than a best-practice. There are arguments to be made for using pixels, points or 'ems' in specifying absolute lengths or widths. The color choices are definitely a matter of preference.

The important thing to note here is that all formatting and positioning directives are isolated to the style sheet. This has two important implications.

- 1. As a developer, you don't need to worry about them.
- 2. The design process can be completed separately, without needing to access your individual pages.

You can now simplify development and maintenance by separating responsibilities. If you have several people working on different parts of the project at the same time, they won't be stepping on each other's toes. Even if you alone are responsible for a site in its entirety, you will benefit from this separation of responsibilities. While working on the content, you can ignore the presentation directives, and while working on the formatting, you don't have to get bogged down in the presentation details.

## What About an Input Form?



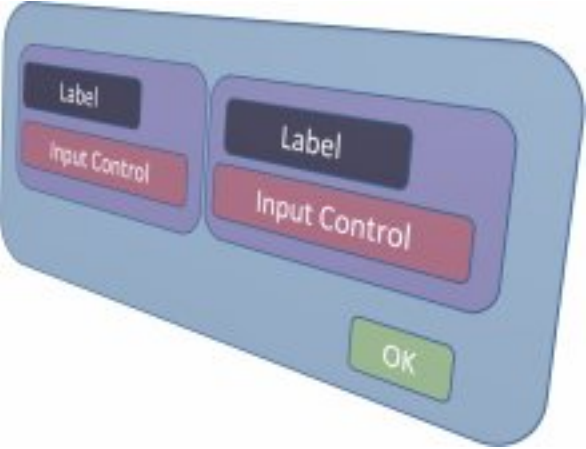
Typically a simple input form such as the one illustrated above may be build using markup similar to this:

```
<table>
  <tr>
    <td>
      <label>Label</label>
    </td>
    <td>
      <asp:TextBox ID="txtBox" runat="server"></asp:TextBox>
    </td>
  </tr>
  <tr>
    <td>
      <label>Another Label</label>
    </td>
    <td>
      <asp:TextBox ID="txtAnotherTextBox"
        runat="server"></asp:TextBox>
    </td>
  </tr>
</table>
```

```
<tr>
  <td>&nbsp;</td>
  <td align = "right" >
    <asp:Button ID="btnOK" runat="server" Text = "OK" />
  </td>
</tr>
</table>
```

With just a little bit of formatting, this will look exactly like the image above, so what is wrong with it?

The first problem is the extra markup. The table and table data elements serve no value other than to add extra markup to the page. The other problem is the rigidity. What has to change to go from the image shown initially to this image?



Such layout changes are often more common than you would expect, and nearly every line of the html markup using a table for layout will have to change. There are additional problems, but these two alone should be enough to cast doubt on the virtues of using a table for positioning.

The following markup can be used to accommodate both looks. All that has to change is a couple of changes to the style sheet.

```
<div class="workArea">
  <p>
    <label>Label</label>
    <asp:TextBox ID="txtBox1" runat="server"></asp:TextBox>
  </p>
  <p>
    <label>Another Label</label>
    <asp:TextBox ID="txtAnotherTextBox1" runat="server"></asp:TextBox>
  </p>
  <p>
    <asp:Button ID="btnOK" runat="server" Text="OK"
      CssClass="commandButton" /></p>
</div>
```

With this markup, a style sheet similar to this will render the first image:

```
label
{
  display: inline-block;
  width: 10em;
}
.commandButton
{
  float: right;
}
.workArea
{
  width: 350px;
  border-top: solid 1px silver;
  border-left: solid 1px silver;
  border-right: solid 1px black;
  border-bottom: solid 1px black;
}
```

Simply change the style sheet to this and get the second image:

```
label
{
  display: block;
  width: 10em;
}
.commandButton
{
  float: right;
}
p
{
  margin-top: 0px;
  margin-bottom: 0px;
  display:inline-block;
  width : 170px;
}
.workArea
```

```
{
    width: 350px;
    border-top: solid 1px silver;
    border-left: solid 1px silver;
    border-right: solid 1px black;
    border-bottom: solid 1px black;
}
```

The transformation happens by displaying the label in a block and displaying the paragraphs as an inline block. This simple model can be extended to let the browser dynamically show as many columns of input control as the current screen resolution should reasonably support.

Amazingly, we get this extra flexibility with less markup.

## Conclusion

There are many ways to structure a web page; many ways that could even produce web pages that look identical. Each of these ways is not necessarily equivalent, some ways are better than others. Understanding these differences will make it easier to build web pages that are more standards compliant and easier to maintain. The better you understand html and style sheets the more easily you can master new technologies like silver light and beyond.

This article has been viewed 38087 times.


Thank this author by sharing:     2



**Author profile:** [Nick Harrison](#)

Nick Harrison is a Software Architect and .NET advocate in Columbia, SC. Nick has over 14 years experience in software developing, starting with Unix system programming and then progressing to the DotNet platform. You can read his blog as [www.geekswithblogs.net/nharrison](http://www.geekswithblogs.net/nharrison)

[Search for other articles by Nick Harrison](#)

**Rate this article:** Avg rating:  from a total of 135 votes.

☐ Poor

☐ OK

☐ Good

☐ Great

☐ Must read

Have Your Say

Do you have an opinion on this article? Then [add your comment below](#):

You must be logged in to post to this forum

[Click here to log in.](#)

<b>Subject:</b>	<b>Thanks...</b>
<b>Posted by:</b>	<i>Soner Gönül</i> (not signed in)
<b>Posted on:</b>	<i>Friday, January 29, 2010 at 3:49 PM</i>
<b>Message:</b>	Thanks !
	That's really useful..

<b>Subject:</b>	<b>My biggest problem is the height difference between sidebar and content</b>
<b>Posted by:</b>	<i>Anonymous</i> (not signed in)
<b>Posted on:</b>	<i>Sunday, January 31, 2010 at 10:52 PM</i>
<b>Message:</b>	<p>That is why I still use a table for the main layout. I've tried a lot of things to get the sidebar and content to expand to the height of the other both ways. Haven't found a method that works 100% of the time except for tables.</p> <p>If you have any suggestions...</p> <p>One thing I don't understand is why the standard body that define how HTML and CSS should work make it so hard to do this kind of layout. I don't like having to trick the browsers into doing something just to make it work. It would have been better if they thought more of how they could make it easier for the designer to put things together.</p> <p>A great start to this is to change the div tag so it fills 100% height or go 100% height when told to by the height style. It already goes 100% in width.</p> <p>That's probably the biggest reason people are not moving away from table layouts.</p> <p>Jeff</p>

**Subject:** More best practice examples would be great!  
**Posted by:** Oleg (not signed in)  
**Posted on:** Monday, February 01, 2010 at 1:52 AM  
**Message:** Thank you for te artichel. A bit more detail about CSS layout best practice would be great. It will help to become more familiar with DIV+CSS layout.

btw, Nick, are there any "CSS design templates" like patterns for programming?

---

**Subject:** ASP.net?  
**Posted by:** Jan C (not signed in)  
**Posted on:** Monday, February 01, 2010 at 6:04 AM  
**Message:** Being new in webdesign, the choice of CSS over tables has been a no-brainer from the start; this article gives me some insight into why (external) stylesheets are indeed the way to go.

I fail to see the ASP.net angle (as suggested in the title) in this story tough.

---

**Subject:** Tables vs CSS  
**Posted by:** David ([view profile](#))  
**Posted on:** Monday, February 01, 2010 at 8:29 AM  
**Message:** When I create a web page from scratch, I try to use CSS as much as possible but still have to resort to tables for some portions. Lately, I have been looking at using a CMS for website development but they too use tables in their dynamically created web pages. Even Microsoft Visual Web Developer 2008 Express edition uses tables.

I do not believe CSS was designed as a table replacement. Its power is in controlling the appearance of a website by modifying a single file.

Thanks for your article

---

**Subject:** Tables and markup  
**Posted by:** Anonymous (not signed in)  
**Posted on:** Monday, February 01, 2010 at 9:39 AM  
**Message:** Markup isn't designed to replace tables and does not work well.

"The table and table data elements serve no value other than to add extra markup to the page"

They serve to format the data. They do so in a logical manner that's easier to make work than css (especially in multiple browsers).

"The other problem is the rigidity"

You're touting the standard technophile line. It should be "elegant for programmers" not functional for users.

This was a poorly thought out post.

---

**Subject:** Re: Tables and Markup.  
**Posted by:** Phil Factor ([view profile](#))  
**Posted on:** Monday, February 01, 2010 at 10:34 AM  
**Message:** I'm not entirely sure I agree with this comment. Tables are there in their full splendor in HTML 5. Nobody is arguing that they should be dropped. I think that Nick argues the point well that they should be used for tables of information rather than being used as a means of laying out the page. If you're still not convinced that CSS can do this, then take a look at a CSS framework such as Blueprint CSS, or read any of the excellent books by Eric Meyer. Also, check out what Ext Js achieves with CSS.

Obviously nobody is going to say that you shouldn't ever use tables, particularly as some design tools make prodigious use of them: However, I've spent quite a lot of time maintaining sites that use deeply-nested tables for general layout, and I'll never be convinced they're a good idea.

---

**Subject:** Eat your own dog food  
**Posted by:** Anonymous (not signed in)  
**Posted on:** Monday, February 01, 2010 at 8:04 PM  
**Message:** There are a lot of self proclaimed web design experts out there praising that CSS should be used instead of tables. These authors are the same ones producing poor quality material that only works in one browser and in some cases, only one version of the browser.

If you tried the examples above, you will realise that they do not work the same across firefox, safari and IE. The examples don't make a point because they don't work unless you have difference CSS files



for different browsers which makes it more difficult to maintain, not easier.

If you try the table example, the method that this author does not recommend, not only do you get a consistent look and feel across all the browsers, but it also positioned correctly without having to tweak the hieghts and widths (for this example).

Regarding the point of added markup, what do you think placing div's and classes are? All you are doing is replacing one markup with another one which doesn't do the job.

CSS is good for formatting certain things, but when it comes to organising content on pages, it wins hands down.

---

**Subject:** **Eat your own dog food**  
**Posted by:** *Anonymous* (not signed in)  
**Posted on:** *Monday, February 01, 2010 at 8:05 PM*  
**Message:** CSS is good for formatting certain things, but when it comes to organising content on pages, tables win hands down.

---

**Subject:** **Fine in theory but ....**  
**Posted by:** *Anonymous* (not signed in)  
**Posted on:** *Tuesday, February 02, 2010 at 4:48 AM*  
**Message:** I really want to go with css style sheets for layout, for all the reasons you outline, but the plain fact is that as soon as you try to use them in the real world, they just don't work reliably. Even those who push them can't manage to get reliable results - time and time again I've downloaded examples which people claim will gi=ve a particular effect, only to find that they don't quite work. And if you come across a website where the layout seems to be going berserk, then you just know its written using stylesheets to try to do the layout.

If only the body that designed style sheets had involved someone who knew a bit about typographical layout!

By the way, anyone capable of using the pseudo-word "advancements" has no right to call anyone elses style smelly :-)

---

**Subject:** **Main points being missed**  
**Posted by:** *Nick Harrison* ([view profile](#))  
**Posted on:** *Tuesday, February 02, 2010 at 9:07 AM*  
**Message:** There are a couple of main points being missed in these discussions.

I do not advocate abolishing tables blindly. I know that tables do serve a very useful role and sometimes even in less obvious roles their use is inevitable.

Issues like sizing the side bar with the content area are real. Sometimes, I will solve this problem with tables as well. I also often solve them by using scrollable regions in the content area, but that does not work for everyone or in every situation.

Another solution that I will often reach for is to make the style sheet dynamic by using a web page that outputs the style sheet customized based on the user's browser settings. This is not as difficult as it sounds, but it can lead to performance / scalability concerns.

There is no harm in using a table to provide the main line structure for a page. The problem that I see is the over use of tables.

I was a difficult convert to the virtues of style sheets. It is no exaggeration that I have written pages with as many as 10 levels deep of nested tables. It is also not an exaggeration that this layout is very difficult to maintain.

The other important point to make is that even if you do not move everything to the style sheet, moving everything that you can lessens what you have to worry about while implementing a page. This can greatly improve the time to market for a project and improve the over all look, even if you are responsible for both the functionality and look and feel.

The example that I mentioned of switching from having labels beside the controls to labels above the controls nearly derailed a project that I was working. After this project, I bit the bullet and learned style sheets I cannot tell you how many hours were wasting reworking tables to make what should have been a trivial change.

Adopting style sheets should not be viewed as an all or nothing proposition. Use style sheets where they make sense, where they simplify your tasks. When they create more complications that they



solve, look for a new solution. Eventually, the browsers will standardize and the standards will catch up with what we need.

---

**Subject:** Just a good example of CSS usage.  
**Posted by:** Oleg (not signed in)  
**Posted on:** Wednesday, February 03, 2010 at 3:30 AM  
**Message:** Guys, take a look at this site: <http://www.csszengarden.com>. It is awesome. The page has one HTML file and many many CSS styles that dramatically change the look of the page. Find "Select design" area and switch from one design to another. Find more designs under Archive label.

---

**Subject:** Add a Back-To-Top button to your extra long web pages  
**Posted by:** elizas ([view profile](#))  
**Posted on:** Wednesday, February 10, 2010 at 3:59 AM  
**Message:** You may have seen 'back to top' link to take the user to top of the page.

There are two ways to do it.

1. Giving the position to (0,0) through javascript.

```
<a href="javascript:scroll(0,0)">
Back to top
</a>
```

You can specify any position by x- and y- coordinate to scroll to particular position in the page.

2. <%--Link to take the User to starting of the page--%>  
<a href="#MainContainer">back to top</a>

where MainContainer is the ID of the main container of the page.

Some jQuery plugins are also available to give sliding effect.  
<http://www.mindfiresolutions.com/>

---

**Subject:** Nice tut  
**Posted by:** MCSE ([view profile](#))  
**Posted on:** Tuesday, September 07, 2010 at 12:57 AM  
**Message:** nice post  
thanks  
<http://www.certxpert.com>

---

**Subject:** Very nice  
**Posted by:** Tahir ([view profile](#))  
**Posted on:** Friday, October 08, 2010 at 1:08 PM  
**Message:** very good post.

---

**Subject:** I really enjoyed reading this post, but I do have a question  
**Posted by:** mikener ([view profile](#))  
**Posted on:** Wednesday, November 23, 2011 at 11:34 AM  
**Message:** I tried to add the CSS you used for the input form into one of my pages, but I was unable to because there is no Head Tag on pages that use Master Pages. The only page that has a head tag is the Master Page. Does that mean that the only way to add CSS to a page is by adding it to the Master Page?

Thanks

---