CSS - LAYOUTS

Hope you are very comfortable with HTML tables and you are efficient in designing page layouts using HTML Tables. But you know CSS also provides plenty of controls for positioning elements in a document. Since CSS is *the wave of the future,* why not learn and use CSS instead of tables for page layout purposes?

The following list collects a few pros and cons of both the technologies −

- Most browsers support tables, while CSS support is being slowly adopted.
- Tables are more forgiving when the browser window size changes - morphing their content and wrapping to accommodate the changes accordingly. CSS positioning tends to be exact and fairly inflexible.
- Tables are much easier to learn and manipulate than CSS rules.

But each of these arguments can be reversed −

- CSS is pivotal to the future of Web documents and will be supported by most browsers.
- CSS is more exact than tables, allowing your document to be viewed as you intended, regardless of the browser window.
- Keeping track of nested tables can be a real pain. CSS rules tend to be well organized, easily read, and easily changed.

Finally, we would suggest you to use whichever technology makes sense to you and use what you know or what presents your documents in the best way.

CSS also provides *table-layout* property to make your tables load much faster. Following is an example −

```
<table style="table-layout:fixed;width:600px;">
   <tr height="30">
      <td width="150">CSS table layout cell 1</td>
      <td width="200">CSS table layout cell 2</td>
      <td width="250">CSS table layout cell 3</td>
   </tr>
</table>
```

You will notice the benefits more on large tables. With traditional HTML, the browser had to calculate every cell before finally rendering the table. When you set the table-layout algorithm to *fixed*, however, it only needs to look at the first row before rendering the whole table. It means your table will need to have fixed column widths and row heights.

Sample Column Layout

Here are the steps to create a simple Column Layout using CSS −

Set the margin and padding of the complete document as follows −

```
<style tyle="text/css">
   <!--
   body {
      margin:9px 9px 0 9px;
      padding:0;
      background:#FFF;
   }
   -->
</style>
```

Now, we will define a column with yellow color and later, we will attach this rule to a :

```
<style tyle="text/css">
   <!--
   #level0 {
      background:#FC0;
   }
   -->
</style>
```

Upto this point, we will have a document with yellow body, so let us now define another division inside level0 −

```
<style tyle="text/css">
   <!--
   #level1 {
      margin-left:143px;
      padding-left:9px;
      background:#FFF;
   }
   -->
</style>
```

Now, we will nest one more division inside level1, and we will change just background color −

```
<style tyle="text/css">
   <!--
   #level2 {
      background:#FFF3AC;
   }
   -->
</style>
```

Finally, we will use the same technique, nest a level3 division inside level2 to get the visual layout for the right column −

```
<style tyle="text/css">
   <!--
   #level3 {
      margin-right:143px;
      padding-right:9px;
      background:#FFF;
   }
   #main {
      background:#CCC;
   }
   -->
</style>
```

Complete the source code as follows −

```
<style tyle="text/css">
   body {
      margin:9px 9px 0 9px;
      padding:0;
      background:#FFF;
   }

   #level0 {background:#FC0;}

   #level1 {
      margin-left:143px;
      padding-left:9px;
      background:#FFF;
   }

   #level2 {background:#FFF3AC;}
```

```
   #level3 {
      margin-right:143px;
      padding-right:9px;
      background:#FFF;
   }

   #main {background:#CCC;}
</style>
<body>
   <div >
      <div >
         <div >
            <div >
               <div >
                  Final Content goes here...
               </div>
            </div>
         </div>
      </div>
   </div>
</body>
```

Similarly, you can add a top navigation bar or an ad bar at the top of the page.

It will produce the following result −