E-Mail Coding Tips

How to Properly Compose E-Mail HTML Code

Learn best practices for formatting correct HTML code for ANY e-mail!

correct HTML code for ANY e-mail!
Learn best practices for formatting

For any successful business, creating and sending great looking e-mail is essential to project a professional image. With the proliferation of numerous e-mail readers it is a challenge to format each e-mail such that it looks good in all them. Here are some tips to create HTML e-mail code that will not break in the various readers.

1. Learn how to code HTML without the use of a visual editor

Visual editors used by WYSWYG editors (abbreviation for "What You See Is What You Get" editors) usually throw in bad HTML code that can break an e-mail. Extraneous and redundant code is applied that often makes no sense and worse can cause problems in readers such as G-Mail. It is best to learn the simple HTML that needs to be used. This HTML is very close to what was used back in the late '90s during the dawn of the internet. Each different e-mail reader may use different ways of processing HTML code compared to others. Only the most simple code will work in almost all email readers.

Tables and in-line styles are the way to format successful e-mails. For those who are unfamiliar with advanced HTML, you do not have to worry about this when composing e-mails. Only basic tables and CSS that is applied by inline styles are really needed. The use of tables is considered the ONLY way of creating e-mails in virtually all the tutorials found on the internet.

Make it a point to learn the basics of HTML and CSS if you want to compose professional, cleanly formatted emails!

2. Use tables everywhere and use divs sparingly

Tables are your friends in e-mail code. They surround all of the content in such a way that will not break in most e-mail readers. With a table you can make sure that a column of text will appear where you expect it to appear. Images will stay where you want them to stay and not overlap text or other content unexpectedly.

On the other hand, divs should be used sparingly. This is because some e-mail readers do not respect the boundaries that divs create around content. This can cause content to spread out in ways that can break an email. For example, if you need for text to appear in the body of an email contained within a 500px width, using a div set with a width of 500px to contain the text will have the unwanted result of some email readers ignoring that width and spreading the text out.

Tables can be used to get extremely accurate positioning of elements in a design if you use td tags that contain width or height attributes coupled with spacer graphics. The spacer graphic should be a transparent gif that is created with a width and height of exactly 1px. So, for instance, if a header graphic needs to be exactly 30px away from the border, instead of trying to use padding or margin of 30px use multiple tds instead. Margin and padding may be interpreted differently by various email readers, so use them sparingly (and use tds instead).

EXAMPLE:

```
<table border="0" cellpadding="0" cellspacing="0" width="300">
        <tr>
                <td width="30" height="150"><img src="images/spacer.gif"
                width="30" height="150" border="0" alt=""></td>
                <td width="270"><img src="images/header.jpg" width="270"
                height="150" border="0" alt="Bringing You Great
                Value!"></td>
        </tr>
</table>
```

In the above example you may notice that the width and height of the spacer graphic is given a width greater that 1, in this case width="30" and height="150". The original spacer graphic of course remains at 1px. What is happening here? Think of the spacer graphic as a maleable placeholder in the code that can be given any width or height as needed. This is to ensure that a width or height is properly rendered in most email readers. Outlook 2007 in particular may ignore a width or height set in a td containing a spacer graphic if those dimensions are only called in the td itself (and the spacer graphic is just given height and width of 1)

So place that value for desired width and height in both the td that contains the spacer graphic and inside the spacer graphic itself.

```
<td width=”30” height=”150”><img src=”images/spacer.gif” width=”30”
 height=”150” border=”0” alt=””></td>
```

Note that some tutorials found online specify not to use spacer graphics since some email readers don’t show graphics by default. However, this will not be a problem if you always specify a width and height either of 1 or to the exact dimension for the spacing desired (whichever the email design requres for accurate reproduction). That way if the spacer image is just shown as an empty placeholder then at least the desired width and height may be applied (thus not breaking a design).

3. Avoid the use of CSS except to create the basics such as background color, font formatting and some content padding.

Forget about using more advanced CSS such that is used, for example, in absolute positioning relative to a container. Most e-mail readers only accept the basics in CSS. On a side note, don’t try to use background images to create background textures or graphics in a td or div tag. Many e-mail readers ignore background images entirely. If you do try to use them, always also specify a background color that testing shows it will display well should the background image be stripped out.

There is one exception to this rule and that is for calling background images for the body tag. Even this exception does not display as needed in some e-mail readers. For only advanced e-mail coders - it is not covered here.

4. Don’t use external stylesheets for CSS

Don’t use external stylesheets to create CSS. External stylesheets are usually referenced inside the head tags. For some e-mail readers all content embedded inside the head tags is stripped out.

DON’T DO THIS (continued on next page):

```
<head>

    <link href="css/eN_offerstyles.css" type="text/css" rel="stylesheet">

</head>
```

5. Don't create CSS inside the head tags

Again, content inside the head tags may be stripped out.

DON'T DO THIS:

```
<head>
    <style type="text/css">
        p {font-weight: normal; color: gray; }
        h1 {color: black; }
    </style>
</head>
```

6. Use inline styles for CSS

Create CSS at the root element by using inline styles.

EXAMPLE:

```
< p style="font-family: Tahoma, Arial, sans-serif;color:#F6F6F6;" >Some text</p>
```

7. If classes or ids for CSS are used, place them under the opened body tag

If for some reason classes or ids are used, be sure to embed them directly under the opened body tag. This is unheard of for CSS used in regular HTML web pages but is the only way for many e-mail readers to apply classes. Enclose each CSS declaration inside <!--   --> so that Lotus Notes can read it.

EXAMPLE:

```
</head>
<body>
<style type="text/css">

    /*
    The comment tags around each style are there to help Lotus notes not
    ignore them.
    */

    <!--  .border1 { border:1px solid #DEDEDE;} -->
    <!--  .disclaimerText { font:normal 10px Tahoma, arial, sans-
           serif;color:#979696;line-height:14px; } -->
    <!--  .fontSize1 { font-size:12px; } -->
    <!--  .fontSize2 { font-size:11px; } -->
    <!--  .marginPadding1 { margin:0;padding:10px 0 6px 0; } -->
    <!--  .padding1 { margin:11px 0 6px 0;padding:0 } -->

</style>
```

8. Always match all “class=” statements with “style=” at root element

The only way to guarantee that most email readers will accept CSS is to apply “style=” for the desired CSS values at the root element (where the class or ID is to be used). Just referencing classes/ids declared under the closed head tag without also spelling out the class/id attributes with “style=” will cause some e-mail readers to ignore the attributes.

EXAMPLE OF CORRECT USE OF CLASSES PAIRED WITH INLINE STYLES:

```
< p class=”font1” style=”font-family: Tahoma, Arial, sans
  serif;color:#F6F6F6;”>Some text</p>
```

This is why in almost all cases it is better never to use classes or ids and simply apply CSS using inline styles. Unfortunately some visual editors such as the one used by Dreamweaver insist on placing “class=” to create content attributes. You must then go into the code and place “style=” paired with the desired attributes anywhere “class=” is used.

9. Prevent Gmail and Hotmail from placing unwanted padding around images

A quirk of Gmail and Hotmail is to place unwanted padding around images. This can be a problem if you are using spacer images inside <td> tags to control layout spacing. The fix involves two parts.

First, inside the <td> containing the graphic place this:

```
<td height="10" style="line-height:1.0">
        <img src="images/spacer.gif width="1" height="10" border="0" alt=""
        style="display:block">
</td>
```

NOTE: Only do this when a <td> contains JUST images and NO TEXT. If text is also used here, then it will be squashed tiny because the text line height will be forced into a height of 1px!

Next, in the code for each image place this:

```
<td height="10" style="line-height:1.0">
        <img src="images/spacer.gif width="1" height="10" border="0" alt=""
        style="display:block">
</td>
```

NOTE: If 2 or more images are to be displayed inline inside a single container, placing this code will force them onto separate rows. An example of where not to do this is shown below. 2 images, say for social media icons like facebook and twitter, are to be on one line and are also both contained inside just one <td> or <div>. The way to force them onto the same line is to use inline CSS of "display:inline":

```
<td height="10">
        <img src="images/facebook.gif" width="20" height="10" border="0"
        alt="Facebook" style="display:inline"><img src="images/twitter.gif"
        width="15" height="10" border="0" alt="Twitter" style="display:inline">
</td>
```

If "display:block" is used they will probably be forced onto 2 separate rows and will not be side by side.

To avoid this problem altogether it is better to contain each social media icon inside a separate <td>. Then you can use "display:block" and "line-height:1.0" to avoid any excessive padding issues.

```
<td height="10" style="line-height:1.0">
        <img src="images/facebook.gif" width="20" height="10" border="0"
        alt="Facebook" style="display:block">
</td>
<td height="10" style="line-height:1.0">
        <img src="images/twitter.gif" width="15" height="10" border="0"
        alt="Twitter" style="display:block">
</td>
```

10. Prevent iPhone text enlargement

The iPhone automatially changes font sizes when a web page is zoomed in or out.

For many email designs this will cause the text to flow in unwanted areas and break the overall design. To prevent this, use the following code placed just after the opening body tag (same as would be used to create CSS classes:

```
</head>
<body>
<style type="text/css">
        /*
        CSS to prevent iPhones resizing text and breaking the email design
        */

        <!-- * { -webkit-text-size-adjust: none; } -->
</style>
```

11. Validate HTML code using the free W3C validator

When you are done with the HTML code, use the free W3C validator to catch many possible errors (see next page):

http://validator.w3.org/#validate_by_input

Not all the errors shown will be a concern. The validator is very strict on code. Use it to catch tags that are not closed properly, redundant code, etc..

Applying these coding tips will make for much happier results when sending out e-mails. At least there will be less chance of making a bad impression from the email design awkwardly breaking!