Inhoud

What You Should Know About HTML Email

Email is an awesome medium. It goes straight into the inbox and its ROI is widely reported as being through the roof at 4000%. It’s also perpetually misunderstood and too often it’s done badly. With the recent explosion of smartphones, we’re more often reading our mail on our iPhone or Galaxy, but unfortunately a lot of email marketing has failed to keep up. I see this as a huge missed opportunity because a well executed email can be enjoyable to open and hugely successful.

Coding HTML Email Can be a Challenge

If you've already tried your hand at HTML email design and development, you've probably already established that it can be pretty difficult. And you're not imagining it — it is quite hard. Here's why:

Email Standards Don’t Exist

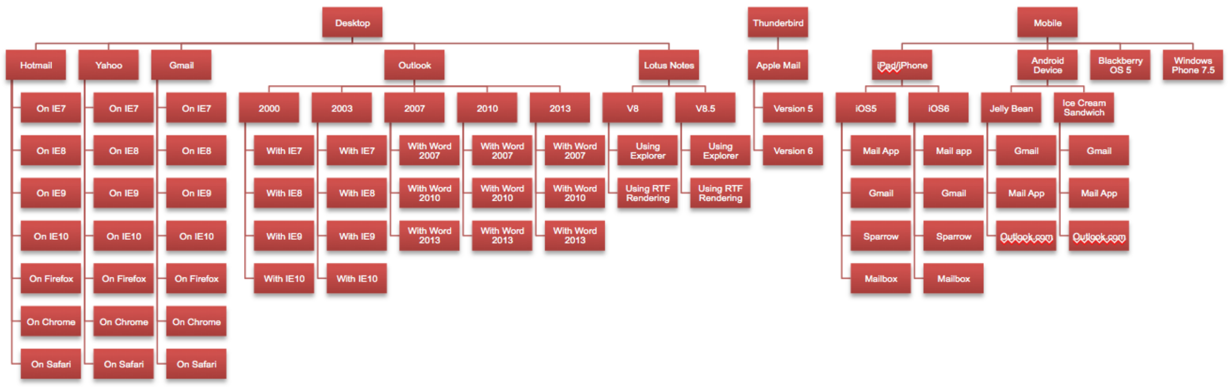
[We will] continue to use Word for creating e-mail messages because we believe it’s the best e-mail authoring experience around.

The Outlook Team

When you code for the web, you can at least count on the fact that all of the major browsers (Chrome, Firefox, Internet Explorer, Safari and Opera) are trying to adhere to web standards for rendering HTML and CSS.

When it comes to email clients, you're testing on a huge bunch of old and new programs. They range from new phones running on Android and iOS to IBM’s Lotus Notes or Microsoft Office 2007 (which infamously renders your lovingly-created HTML with the Word HTML rendering engine. The previous versions of Outlook used a browser to render their HTML — which is actually logical. Why switch to a word processor to render HTML you ask? Well, for "security reasons", they say). And none of these programs have to adhere to any standards. They basically all just make it up. You can see what standards support looks like for some of the most popular email clients at the Email Standards Project.

If that isn’t bad enough, couple that with the next fact: there are about a million different combinations of ways that email can render on desktop and mobile.



The rendering possibilities are (almost) endless.

Here’s a list of some of the most common email clients:

Mobile clients:

- Android 2.3 & 4.0
- iPhone 5 iOS 6
- iPhone 4S iOS 6
- iPhone 3GS iOS 5
- iPad 2 iOS 6
- BlackBerry OS 4 & 5
- Symbian S60
- Windows Phone 7.5

Desktop clients:

- Apple Mail 4, 5, 6
- Lotus Notes 8.5
- Lotus Notes 8
- Thunderbird
- Windows Live Mail
- Outlook 2013
- Outlook 2011 for Mac
- Outlook 2010
- Outlook 2007
- Outlook 2003
- Outlook 2002/XP
- Outlook 2000

Webmail clients:

- AOL Mail (on any browser)
- Gmail (on any browser)
- Outlook.com (on any browser)
- Yahoo! (on any browser)

That's a lot of devices!

If you are already familiar with web development, forget everything you know about it.

To compound all of this, conditional styling isn't much of an option either. There are some things that you can do with conditional comments, but it's limited to targeting certain versions of Outlook, or everything *except* certain versions of Outlook.

If you are already familiar with web development, forget everything you know about it. The single biggest obstacle to you is expecting things to work like 'normal' web development. This will frustrate you and hold you back. The worst thing you can do is get angry that you can't use DIVs or that margin isn't fully supported. So forget everything you know about semantic HTML and the latest CSS spec. Trust me, it will help.

How to Approach Your Work

Let's take a look at some email-building workflow suggestions.

Work Structurally First

Building the structure of your email first can help you avoid many bugs and issues later down the track. Never build the whole thing and then test — you can often end up with too many bugs to deal with, and they may all be influencing each other.

Test Often

Work until you reach a minor development milestone (for example, when you finish the basic structure) and then run a test. The best way to test is using Litmus or Email on Acid. I recommend taking out an unlimited plan with either of these companies because being able to test frequently is really important.

I also really like leaving in all my table borders so that I can see what I am creating, then I turn them all off at the end. You can also perhaps color the background of certain cells to help see which sections are where. My ideal workflow is to create a skeleton, test, then add my content, test, style the colors and fonts, test again and finally remove my borders and test again before sending.

Validate Often

Validate it using the W3C Validator as often as you possibly can. This will help you iron out small details and it will pick up on mistakes like missing or open tags.

Sending Your Email

There are a huge number of options when it comes to sending your email. The two services that I use the most are MailChimp and Campaign Monitor. They offer competitive pricing and they are very easy to use. There are loads of commercial platforms, too — it all depends on your needs. Sign up for a free account with either of these and have a tinker in their systems to see which one you like. Make sure you utilize the useful data that both services collect about your emails, such as open times and email client usage. This can really help you focus your efforts in the right area the next time you send.

Content, Development and SPAM Scores

When it comes to SPAM; content, design and development all go hand in hand. It’s important to avoid typical spammy tactics like using all-caps and lots of exclamation points in your subject line. There are certain words that are likely to trigger SPAM filters too (like ‘free’ and ‘invest’). The cleaner your code, the less likely your email is to be marked as SPAM, and the ratio of images to text also has an effect. Image-reliant emails with no text are more likely to be marked as SPAM and so are emails with really long image filenames.

The world of SPAM scores is a tricky one and it’s important to run a SPAM test through your testing account with Litmus or Email on Acid before you send your email, to make sure all your hard work isn’t headed straight for the Junk folder.

Diving into Development

Now, for the nitty-gritty of email development..

Tools of the Trade

You’ll need a text editor that you like (I use Sublime Text) and a test account with Litmus or Email on Acid. I highly recommend having an unlimited test account with either of these companies as it will make your life so much easier. If you don’t pay a monthly fee, you will end up paying between $3 and $5 per test which can add up pretty quickly.

Starting With a Good Base

I think that it’s good to start with a blank slate. Frameworks like the HTML Email Boilerplate are full of wonderful tricks and snippets that you can implement piece by piece. However, if you are just starting out I don’t recommend using it as a starting point as it contains a lot of elements that you won’t need. Boilerplates can often make it more difficult to troubleshoot any issues if there is a lot of unused code in your file.

Note: Because it can be very precarious to use any kind of editor (especially when it's time to troubleshoot), you should never use a WYSIWYG editor, or any kind of editor that promises to take your formatted design and magically turn it into valid HTML for emailing. This stuff just never works.

!DOCTYPE

This might seem like a technical detail to start with, but you need a blank template to start working with, and that template needs a Doctype. A doctype is essentially a line of code which informs the program reading it which HTML tags to expect and which set of rules the HTML and CSS adhere to. Quite a few clients strip your Doctype out, and some even apply their own. Many clients do honor your doctype and it can make things much easier if you can validate constantly against a Doctype.

Using an XHTML doctype generally has the fewest quirks and inconsistencies between documents. I use XHTML 1.0 Transitional because it has proven itself to be the most reliable doctype in my experience. In the following tutorial, during which we'll build a complete HTML email template, we'll use the following code to start off our document:

```
<!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=UTF-8" />
<title>Demystifying Email Design</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0"/>
</head>
```

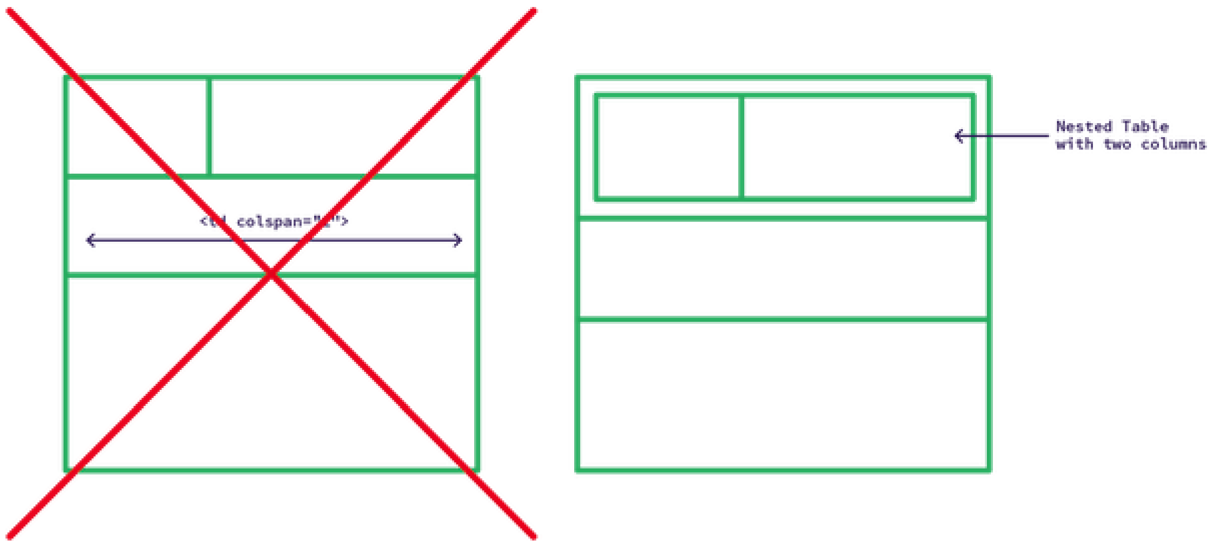
The `Content-Type` meta tag is for telling the destination rendering engine how to process text and special characters. In reality, you need to encode all of your special characters anyway (e.g., & becomes &amp; for ampersand) to be safe, but it's worth keeping this line in there anyway.

The viewport meta tag is telling the device to set the viewable area to the width of the device's screen. It also sets the initial scale to 'normal' which is neither zoomed in nor out. If you don't specify this, many smartphones may scale your content down so that the content fits within the viewable area, but not any of its padding or margins. This can result in text and images butting right up against the edge of the screen.

Finally, always enter a meaningful title because this is what people will see when they view the email in a browser, or share it with their friends.

Email is All About Nesting Tables

Due to the lack of standards support in email, it's not possible to use divs, sections or articles — instead you have to use tables. Moreover, you need to use lots and lots of nested tables because neither the `colspan` nor `rowspan` attributes are properly supported.



Paragraphs or Cells?

Again, because of the lack of standards support, it's not a great idea to use standard tags like `h1`, `h2`, `h3` or `p`. I find that these can render really inconsistently across email clients and can create some pretty big headaches.

Your best bet is to place every block of text into its own cell and apply inline styling to that cell, for example:

```
<tr>
<td style="font-size: 12px; font-family: Arial, sans-serif; color: #666666;">
  Text
</td>
<tr>
```

Inline Styles or Stylesheets?

This one is more of a personal choice. I prefer to keep all my styles inline (ie: within the HTML tags themselves) because I like to know exactly where everything is and what is affecting what. You can code using styles and then pull them all inline at the end (Campaign Monitor and MailChimp can do this for you automatically, you can also use Premailer or something similar) but the reason I don't like this is because you get to know your code, run it through an inliner, and then your code can become somewhat unrecognizable. I find that this makes it more difficult to troubleshoot. Saying that, if this is the way you want to work, that's fine; there is no technical reason why you shouldn't do it.

Don't forget: You can't apply multiple classes to things in HTML email because it is not supported. Every element can have a maximum of one class.

Also don't forget: You can't use shorthand for things like font size (i.e. style="font: 8px/14px Arial, sans-serif;") because it is not supported.

Image Names and SPAM Scores

When saving out images remember that it’s good to give your images names that are short and meaningful because it will improve your spam score. Names like "campaign_054_design_0x0_v6_email-link.gif" are likely to have a much higher SPAM score than "email.gif".

Image Size

It’s also a really great idea to try to keep your entire email as small as humanly possible: under 100kb is ideal but not always possible, under 250kb is pretty standard. Use a compression app like JPEGmini or tinyPNG to cut all your images down to size as much as possible before you send. Slower load times, especially on mobile, can make or break your email if the overall file size is too large.

Other Gotchas

Don’t leave anything up to the email client. Specify all your widths, because otherwise you could end up with unexpected results. For your main container elements always set the size in pixels. You can then use percentages inside your containing element if you wish.

Conclusion

There's a lot to take into account when designing and developing HTML email, most of which involves "un-learning" standards you've been encouraged to practice for web design over the years. However, this tutorial should have given you a solid base to work from, and you're now ready to jump into the actual build process. Onwards!

Useful Links

I referenced a few things during this tutorial - so here they are again, all in one place.

- Litmus testing tools
- Email on Acid testing tools
- The Outlook Team Blog
- The Litmus Team Blog
- The Email Standards Project
- W3C Validator
- MailChimp
- Campaign Monitor
- Premailer, preflight check for emails
- JPEGmini image compression tool
- tinyPNG image compression tool
- Sublime Text, my preferred editor
- Say Hello to the HTML Email Boilerplate
- Don't Forget the Viewport Meta Tag
- Thumbnail icon by Pierre Borodin