Understanding Responsive Email Design

A How To Guide for Agencies

If an email is not responsive...

80%
of consumers will delete the email

34%
of consumers won't buy

27%
of consumers will simply unsubscribe

vision6

INTRODUCTION

Because email uses HTML and CSS, the same technologies as the web, many designers assume that the design approach will be very similar. Unfortunately, this just isn't the case.

While the web standards movement has established relative parity amongst web browsers, the same movement has not occurred to modernize email clients. Email designers are burdened with email clients that use old rendering engines. These email clients lack the standards compliance of modern browsers, making it difficult to achieve consistent results with many common responsive web design techniques. An email opened in Gmail can look very different from one opened in Outlook or Apple Mail.

In this guide, we will outline 7 simple steps to create a modular, reusable framework to achieve responsive email success. To help you get started, we've created an example responsive email template which uses many of the techniques described in this article.

It's time to remove the label of "dark art" from building responsive emails, because "responsive email design" IS "email design".

STEP ONE: SET YOUR BASELINE

Similar to web development, you need to "cut the mustard" and determine the baseline email client which you will support. It's impossible to design for the countless combinations of email clients and platforms, so you need to determine which email clients make the most sense for you to support.

A good first step in setting your baseline is to know the 10 top email clients by usage and market share. Vision6 releases a monthly metrics report that can provide you with the stats you need.

If you have an ESP, review the data on open rates by email client, and determine which are most important to support for your recipients.

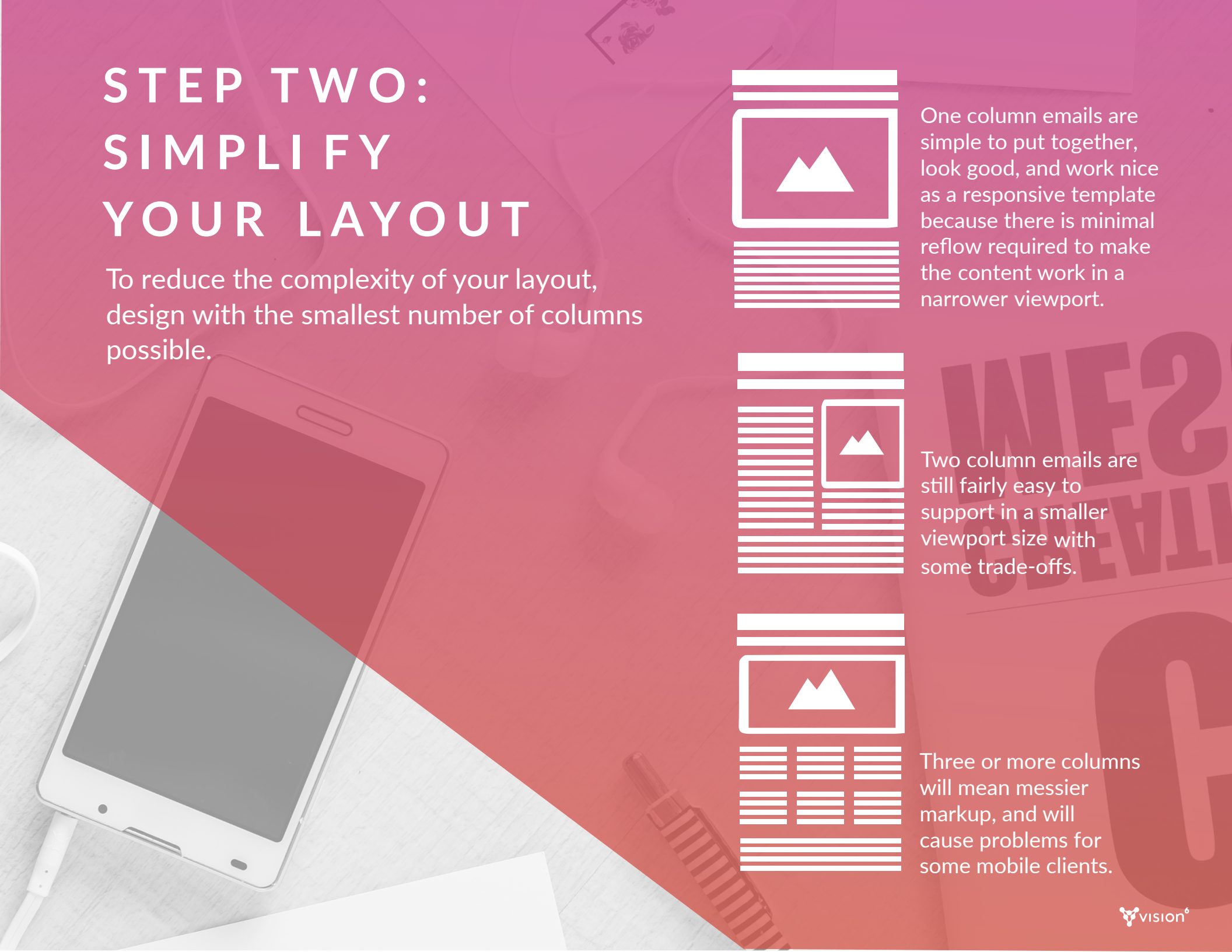
Carefully consider how much additional effort it will take to support older, less common email clients. If you're in a particular situation where you are required to support a given email client ("the CEO uses Outlook"), research the specific limitations of that platform and adjust your approach accordingly.

Even after setting a baseline, you will discover that there are some unavoidable problem children, namely Outlook (especially 2007, 2010 and 2013) and Gmail. Outlook is the Internet Explorer of the email world: Microsoft Word is still used as the rendering engine for displaying HTML emails in Outlook 2013.

Webmail clients, including the popular Gmail, are also problematic. Gmail uses a preprocessor to strip out any code which could interfere with the way it chooses to render an email. This strips out JavaScript and Flash as well as <head> and <body> tags.

STEP TWO: SIMPLIFY YOUR LAYOUT

To reduce the complexity of your layout, design with the smallest number of columns possible.



One column emails are simple to put together, look good, and work nice as a responsive template because there is minimal reflow required to make the content work in a narrower viewport.

Two column emails are still fairly easy to support in a smaller viewport size with some trade-offs.

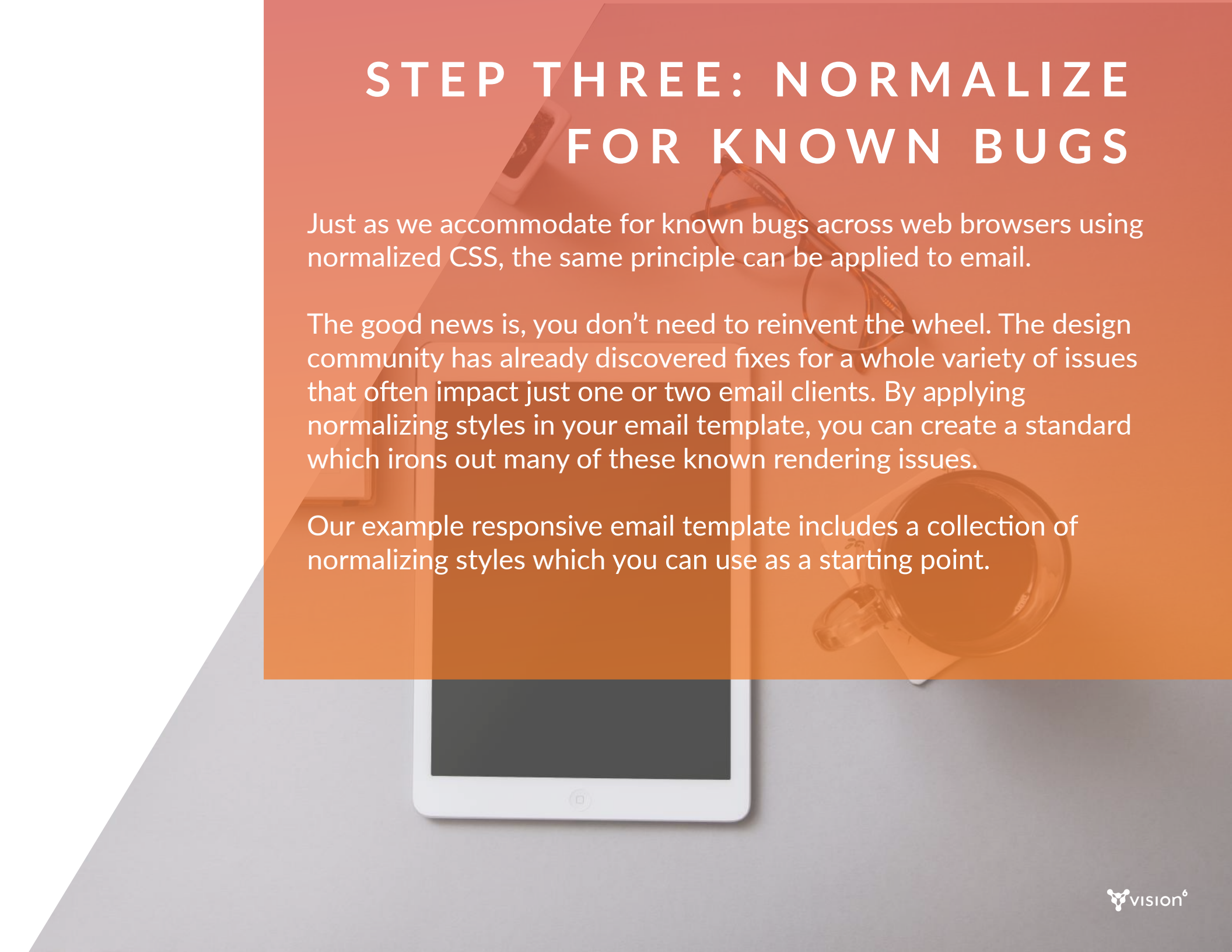
Three or more columns will mean messier markup, and will cause problems for some mobile clients.

STEP THREE: NORMALIZE FOR KNOWN BUGS

Just as we accommodate for known bugs across web browsers using normalized CSS, the same principle can be applied to email.

The good news is, you don't need to reinvent the wheel. The design community has already discovered fixes for a whole variety of issues that often impact just one or two email clients. By applying normalizing styles in your email template, you can create a standard which irons out many of these known rendering issues.

Our example responsive email template includes a collection of normalizing styles which you can use as a starting point.



STEP FOUR: USE TABLES TO BUILD YOUR EMAILS

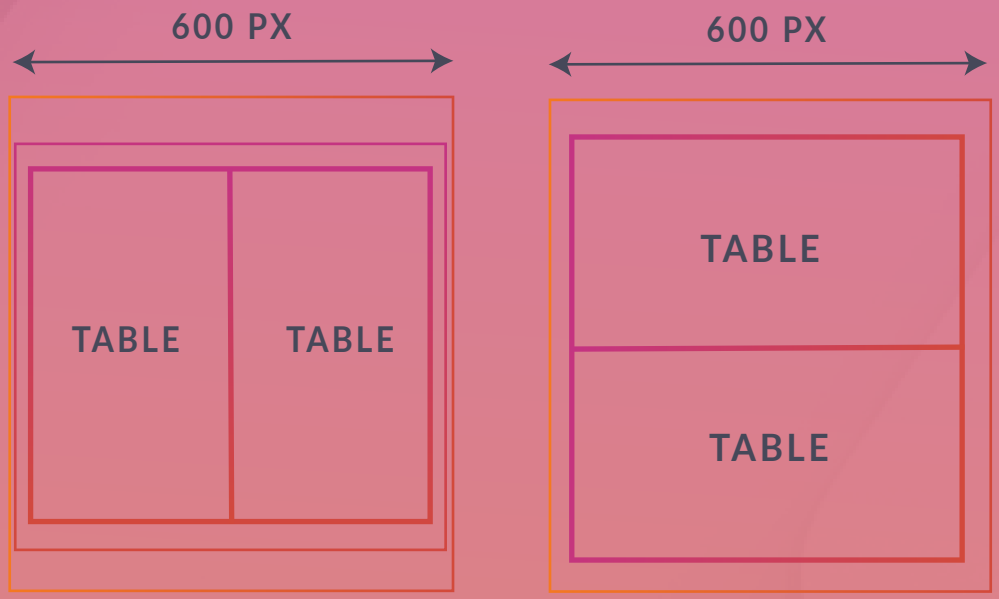
```
<table width="100%">
      <tr>
            <td align="center">
                  <table width="600">
                        <tr>
                              <td align="left">
                              <p>Hello, world!</p>
                              </td>
                        </tr>
                  </table>
            </td>
      </tr>
</table>
```

HTML emails are built using tables because table-based layouts have near-universal support in email clients. Semantic elements and float-based positioning have mixed support, so using table cell stacking allows for more consistency.

We recommend building a fluid layout using one fixed width as the containing column size. The containing column size is the width at which your email looks the best, and will serve as our default layout for wider viewports.

In the example to the left, we have outlined the basic building block for your responsive email template. We have set the column width to 600 pixels (as an attribute). This is the only fixed width in our responsive email template.

USE A FLUID CONTAINER WIDTH



We are frequently asked by designers if it is possible to use a media query to create a responsive email template. The short answer is yes, but it won't be supported in all clients, so shouldn't be relied upon as the only means for your content to scale.

We recommend taking a hybrid approach, where your basic layout is fluid so that it will automatically stack when it gets smaller than the column size, then use a media query as a progressive enhancement for those clients which support it.

You'll see in our example template that wherever we fix a width, we also apply a .responsive-container class, which sets the width to 100% when our media query fires.

Unlike responsive web design, where we commonly design for a number of breakpoints at different device sizes, in responsive email design our primary concern is that our containing table sticks to the width of the viewport. The most important factor is how the email looks when it is first opened, not how it scales up and down.

Although there are many limitations when designing for email, we can employ our existing practice of progressive enhancement to introduce nice-to-have features on email clients with better support. As on the web, we just need to make sure that the fallback works sufficiently well on those clients below our baseline.

For example, take rounded buttons and webfonts.

We can achieve a button with rounded corners using some simple CSS, reminiscent of what we might use on the web. The trick here is to use an extra-wide border instead of padding to ensure the whole surface of the button can be activated.

Webfonts have a reasonable level of support in email clients, so feel free to include them. Just ensure that you've specified a font stack that contains at least one default font. We can wrap our font-face declaration in a media query to avoid Outlook falling back to an incorrect default.

Rounded Button

```
<!-- Abusing border to give us a rounded button -->

<a href="#" target="_blank"
style="display:block; border:16px solid #006666;
border-radius:3px; color:#FFFFFF; background-color:
#006666;">Rounded button</a>
```

Webfonts

```
/* Wrap webfonts in a media query so Outlook ignores
them, otherwise it falls back to Times New Roman. */

@media screen{
        @font-face{
                font-family: 'Source Sans Pro';
                font-style: normal;
                font-weight: normal;
                src:
  url:(https://fonts.gstatic.com/s/sourcesanspro/v9/
  0De1aHBYDBqgeIAH2z1NV_2ngZ8dMf8fLgjYEouxg.woff2)
  format('woff2');
        }
}
```

STEP FIVE:

USE PROGRESSIVE ENHANCEMENT

STEP SIX: STYLE INLINING

Webmail clients often strip the <head> element from an email, taking your painstakingly crafted styles with it. Other clients have unpredictable behaviour when applying CSS.

To maximise the chance that our styles are applied accurately by all email clients, it is advisable to use style inlining. “Inlining” refers to copying a style and applying it inline (as a style attribute) wherever the CSS selector matches an element.

Inlining CSS is a time-consuming manual process, but fortunately can be automated. Many ESPs have an option to do this automatically before sending, and there are inlining tools available online.

STEP SEVEN: TESTING

Before sending your email, it is always advisable to test your code to see how it displays in a variety of email clients. Even if you follow all of these steps, things can and do go wrong, and adequate testing helps pick up issues before your email goes out.

If resources weren't an issue, you could build a device lab to test directly on all of the platforms you're looking to support. Realistically, for most of us this is not an option.

Instead, you can use a testing tool like Litmus, which allows you to preview and test your code in real time across dozens of the most widely used email clients. Vision6 has an inbuilt Litmus integration that allows for convenient instant previews.

vision6

CONCLUSION

This guide has provided you with an introduction to the tools and techniques necessary to design responsive emails. By employing responsive email techniques you should find a significant increase in user engagement.

Remember: if in doubt, keep it simple.

To get started, go to vision6.com/responsiveemailtemplate to see what the template looks like live and download it for yourself!