Responsive Email Design

Designing Your Email to Look Good on Any Device

2017 Revised Edition

GUIDE

goolara

This guide is published by:
Goolara, LLC
1030 Country Club Drive, Suite D
Moraga, CA 94556
Telephone: (510) 522-8000
(888) 362-4575
Fax: (510) 522-2457

www.goolara.com

TABLE OF CONTENTS

INTRODUCTION

The hottest topic of discussion in the email marketing field right now is that of responsive design. Responsive design means that email changes its appearance if the viewing device falls within certain size ranges. For instance, suppose you have an email that features three columns of information in 12 pt type. It may look perfect on a desktop, but open the same email on a smart phone and suddenly that 12 pt type is reduced to three point type and is virtually unreadable. Of course, the reader can always double-tap or pinch-out (zoom) to expand the text to a legible size and then scroll around to read it, but asking your readers to work is always a dangerous proposition. It's better if the reader doesn't have to do anything to receive your message and you make it as easy as possible to respond to your Calls to Action.

There's no question that responsive design can help with this. It will automatically format your email according to the parameters you've set ahead of time and display the email in the optimal format. When used carefully, responsive design can help improve recipient response. So wouldn't you always want to use it? Well, maybe not. As with most aspects of email marketing, there's more to the story than meets the eye.

The email community it neatly divided into two groups on the subject of responsive design. Proponents point to studies that show that responsively designed email performs better. They get more clickthroughs and opens, and surveys show that people respond to them better. Opponents point to surveys that show that responsive design performs only marginally better, and the amount of work it takes to get a responsive design up and running is greater than its benefit.

goolara

Proponents will tell you that creating responsive emails is not really that difficult. This is a tricky argument. While it is not that difficult to create responsive emails once you're up to speed with the skills required to do so, it still adds considerable complexity to the process of email design and testing. For this reason, some ESPs now offer responsive templates. That's fine as long as you don't plan to exert much control over the appearance of your mailings, but most corporate designers have their own ideas of how their mailings should look, and they certainly don't want them to look like everybody else's. A few ESPs now offer tools for dealing with responsive design, but even here, much of your control over the design is lost to pre-coded sections of their choosing. If you want your email to have completely unique characteristics, you'll still have to handle the coding yourself.

In this guide, we'll look at all the pros and cons of responsive design and show you the basics of how it's done. We'll also discuss some of the little tricks to get responsive design to work that aren't always covered in other tutorials. We start the guide from square one and assume no previous knowledge of responsive design. The first chapter discusses the different types of email design and when and how to use them. In chapter two we begin to look at the nuts and bolts of responsive design. Chapter three gets into the actual coding structures and gives you information you need to create a basic responsive email. Chapter four discusses the latest version of Dreamweaver CC and the changes to this program that make it more useful than ever to the Email designer. Chapter five discusses all those little things that can mess up your design. And, finally, chapter six discusses the importance of testing and the various methods you can use to make sure that your email works across all clients, platforms and browsers.

If you are familiar with responsive web design, but have not used it to create email, you may still find out about some idiosyncrasies of responsive email design that were never an issue for you with web design.

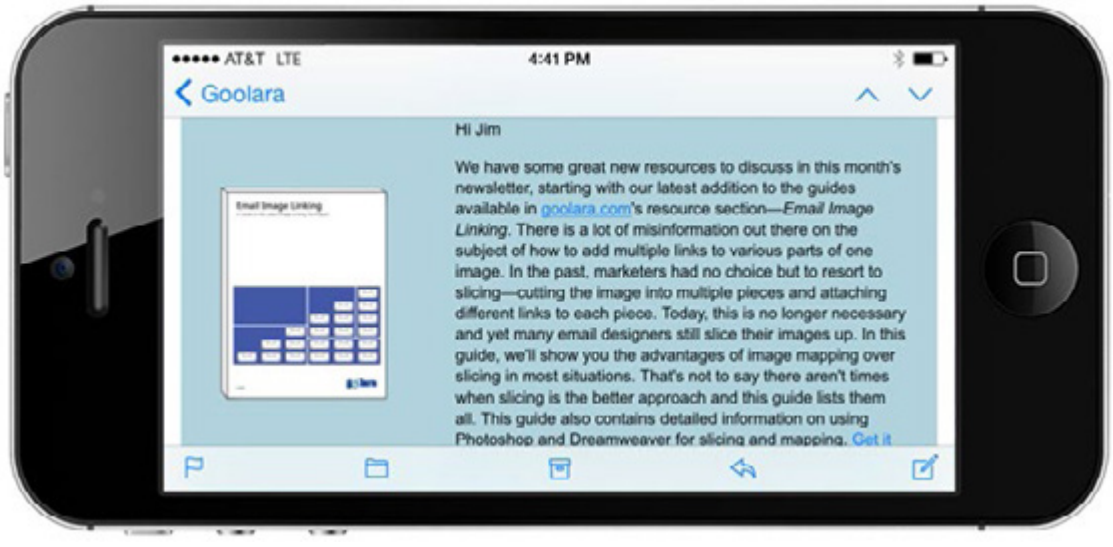
TYPES OF EMAIL DESIGN

When you are designing email that is easy to read on mobile devices, there are two approaches you can take: scalable and responsive. Each has its advantages, and both come into play if you decide to choose the responsive email route. Scalable design is easy. You may be doing it already without even realizing it.

The Scalable Alternative

The alternative to responsive design is scalable design, which has been around for a while. This simply means that you design your email so that it is readable on any device, regardless of its size. If the positioning of images and text is critical, those old email standards, tables, are still the safest way to go. They are still the most compatible way to control image positioning. All email clients support them, which can't be said for either divs or media queries (more on media queries later). Tables don't scale down very well, so you'll want to keep the maximum width low (650 pixels or less) and the minimum type size at least 14 pt to ensure that the type is still readable on a phone when the email is viewed on a small screen.

You still have a certain amount of control over the various design elements. A table, for instance, can be designated with a specific pixel width (e.g., 650 px). As an alternative, you can make it fluid by using a percentage instead (e.g. 90%). If the width is specified in pixels, then the table will maintain its proportions when viewed on a mobile device. In some cases, this means the table will be too small to read easily. In other cases, it means that the table will appear too big for the display and will require scrolling around to read everything.



You can set the elements in your email to be fluid, so that they will change size depending on the width of the screen. This has a slight advantage in some cases since

“The biggest problem with scalable design is that it doesn’t have any built-in mechanisms to recognize what type of device it’s being displayed on.”

the type will then reflow as the device width narrows, but it can also lead to a “scrunched” look if the device width gets below a certain size. Also, tables also don’t allow elements to interact with other elements that aren’t in the same cell, so you won’t get the type reflowing around an image when the window is resized.

An alternative is to use div tags. Divs have the advantage of being inherently more fluid, so you can have your text smoothly flow around an image when the window gets too small for the text and the image to appear side-by-side. And because they don’t scale down like tables, the type stays readable on small devices. On the downside, there is still no consensus on div support. Some email clients support most aspects of divs while others do not. Several email clients ignore positioning and overflow commands, which can cause elements to overlap.

The Responsive Approach

The biggest problem with scalable design is that it’s dumb. That is to say, it doesn’t have any built-in mechanisms to recognize what type of device it’s being displayed on. Whether it’s on a 420 pixel mobile phone, or a 1680 pixel monitor doesn’t matter. It will display its contents according to the parameters you set up as inline styles. Responsive design adds one more factor to this. Using something called a “media query,” the email can change its appearance if it recognizes that it is being displayed on a bigger or smaller screen. For instance, you might want a sidebar panel to disappear completely when the screen size falls below a certain pixel width. With a media query this is easy to indicate by including a “display:none” attribute for that table or cell when the screen size falls below that specified pixel width.

Given the advantages of responsive design, you might wonder why everyone isn’t designing their emails to be responsive. There are some good reasons for not using it. First and foremost, it doesn’t always work. Media queries can’t be used as inline styles, they must be inserted into the email before the actual contents—normally in between the head tags. Some Android phones, and some individual mail apps for both the iPhone and Android do not recognize media queries at this time. Emails displayed on these devices use only the inline styles and either ignore or throw everything else away.

Most browser-based email clients and email software also ignore media queries, although if you are using media queries exclusively to change the email's appearance on mobile devices, this is less of an issue.

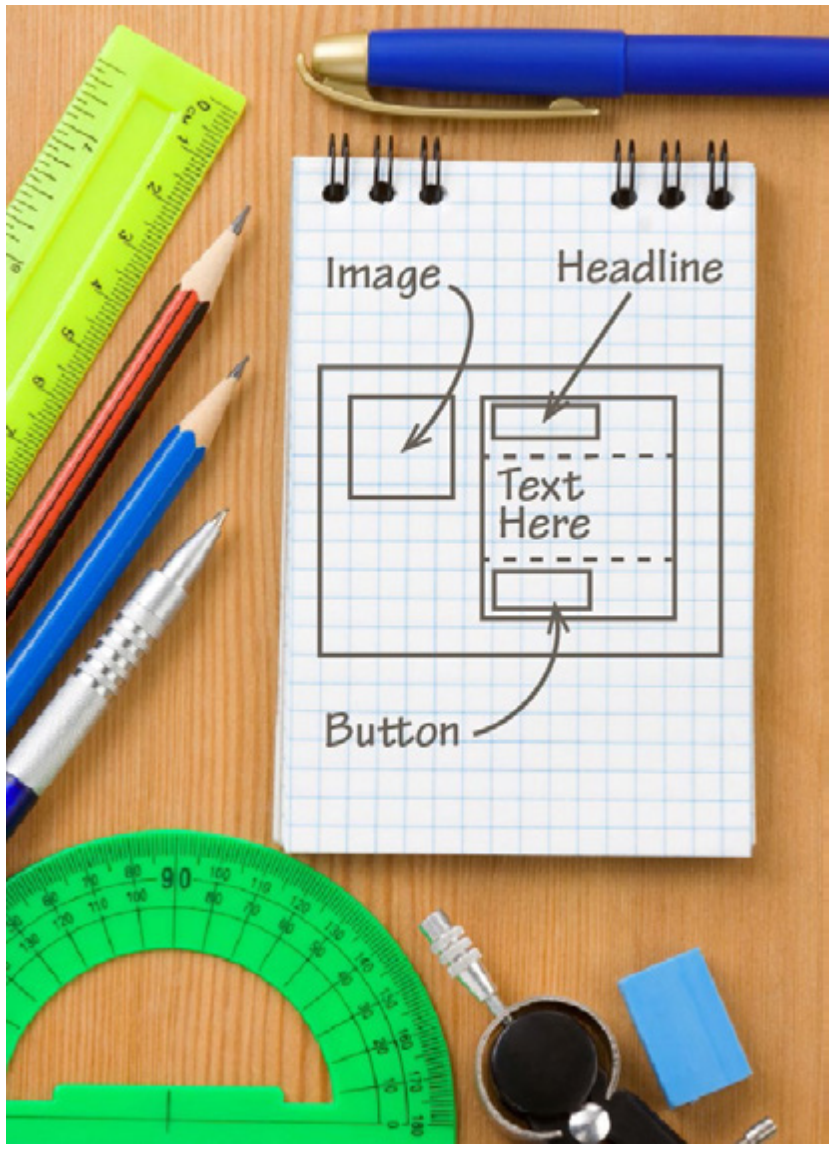
Responsive design also adds a substantial level of complexity to your HTML coding. This isn't much of an issue if you tend to use the same format from month to month, but businesses such as department stores and other retailers that send out several emails every month will often want to make sure that all the mailings look different to keep their recipients interested. If these emails do not share a common template, responsive design might require too much preparation to get the emails ready in a timely manner. This may get quicker as you accrue a library of usable media queries, but at first it is likely to create some traffic jams that you have to be ready to accept if you want to use responsive design.

The Fall-Back Plan

Even if you do plan to use responsive design, you should still make sure that your emails are readable across all platforms without the media query information. This usually means a maximum table width of 650 pixels and a minimum font size of 13 pt. This allows for an email that looks good on a large monitor, but is still readable on an iPhone in landscape mode. In this way, if for some reason the media queries are ignored, you'll still have an email that everyone can read.

In other words, any email you create—even if it's designed to be responsive—should still have the style information inline that it needs for its basic format. Don't fall into the trap of putting all your style information in with your media queries. If an email client can't read your media queries, it probably can't read anything else in the styles tags and is expecting to receive that information in the body of the email.

THE BASICS OF RESPONSIVE DESIGN



On the surface, responsive email design is not that complicated. In its simplest terms, responsive email contains a special set of instructions that lets the email change how it appears according to characteristics of the device upon which it's being viewed. An email with four columns across, for instance, might look great on a desktop computer, but it will be very difficult to read on an iPhone if those four columns are allowed to scrunch down to a mobile phone's 480 pixel width. Conversely, a one column design with type large enough to be easily read on the mobile phone might look strange on a 22" monitor. With responsive design, you can tell your email to change its format if the viewing area falls below a certain size.

About Media Queries

As we've already mentioned, the key to responsive design is a feature of CSS (Cascading Style Sheets) called media queries. When media queries are set up for web sites, they are normally included as part of a CSS file, but in email they are put between the style tags and either included between the head tags, or included before the visual content of the email, depending on your email marketing software's choice of structure.

A media query looks something like this:

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

@media only screen and (max-width:
425px) { table.container {width: 100%
!important;} }

</style>
```

With this media query, once the screen width falls below 425 pixels, any table with the class of "container" will change from its original size to fill 100% of the screen. We could add the

following class to the media query:

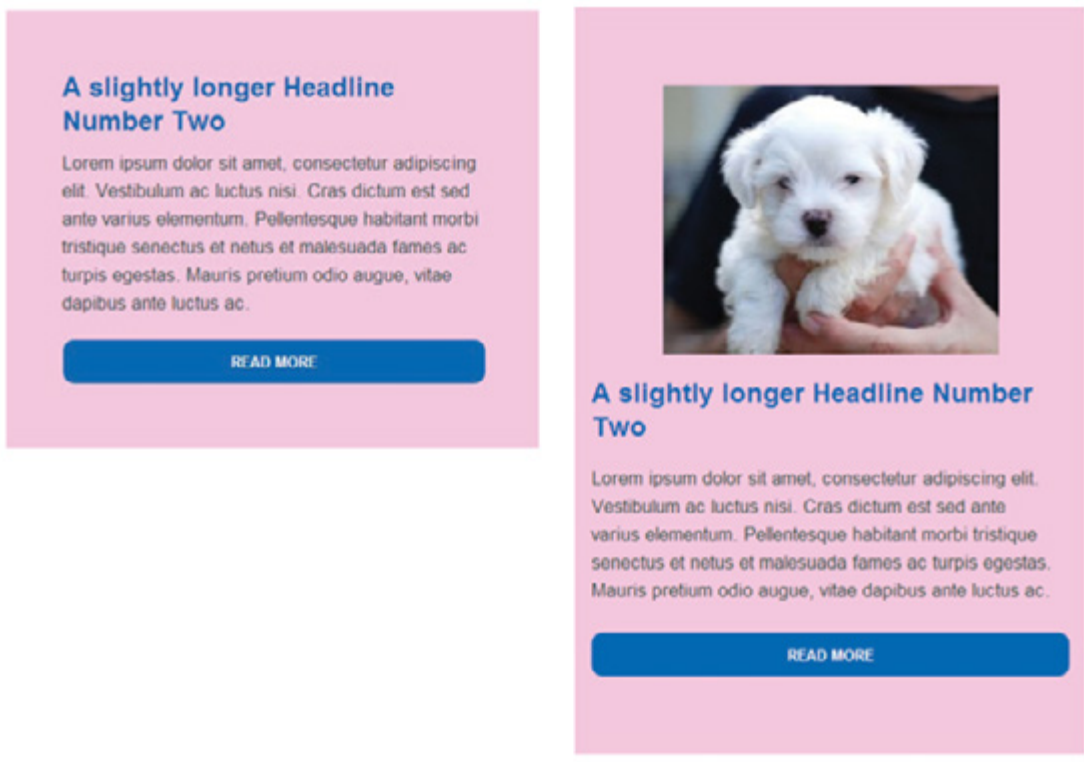
```
td.HideOnPhone {display:none !important;}
```

This would make any table cell that is of the class "HideOnPhone" hide its contents when the screen width fell below 425 pixels. This can be useful for sidebars or images that would clutter the email on a small screen. You may also see this information expressed like this:

```
td[class=HideOnPhone] {display:none
!important;}
```

Both of these statements accomplish the same thing, and which format you choose to use depends on your personal preference. For the examples in this book, we will be using the first format.

Here is a section of an email shown two different ways. In the version on the left, a class of "HideOnPhone" has been added to the cell containing the image. All other attributes are the same.



Media queries let you control virtually every element on a page. You can change the fonts sizes, the positions of the images, whether or not an image will appear at all below a certain size, and much more. You'll notice at the end of each of these style definitions is the word "important" preceded by an exclamation point. Don't let that exclamation point fool you. It is not the same as the <!-- > tag in HTML, which denotes a comment, nor does it indicate a "NOT" statement as it does in some programming languages. In CSS, the "!important" means that this style should override any previous style. How important or unimportant this attribute is will vary, but as a rule, it is a good idea to include the "!important" attribute in all the operators in the media queries.

Rethinking Tables and Divs

Web design has moved away from the use of tables to control element positioning, but email has not. If you are used to designing web pages and have forgone tables in favor of divs, there is a tendency to want to do likewise in your email design. But email design is not web design. It comes with its own set of rules and its own idiosyncrasies. One of these idiosyncrasies is that tables work much better in email than divs. In the first place, not all email clients play nice with divs. Outlook.com and Hotmail, for instance, choose to ignore floats and margin, unless the words are capitalized, which complicates the design process.

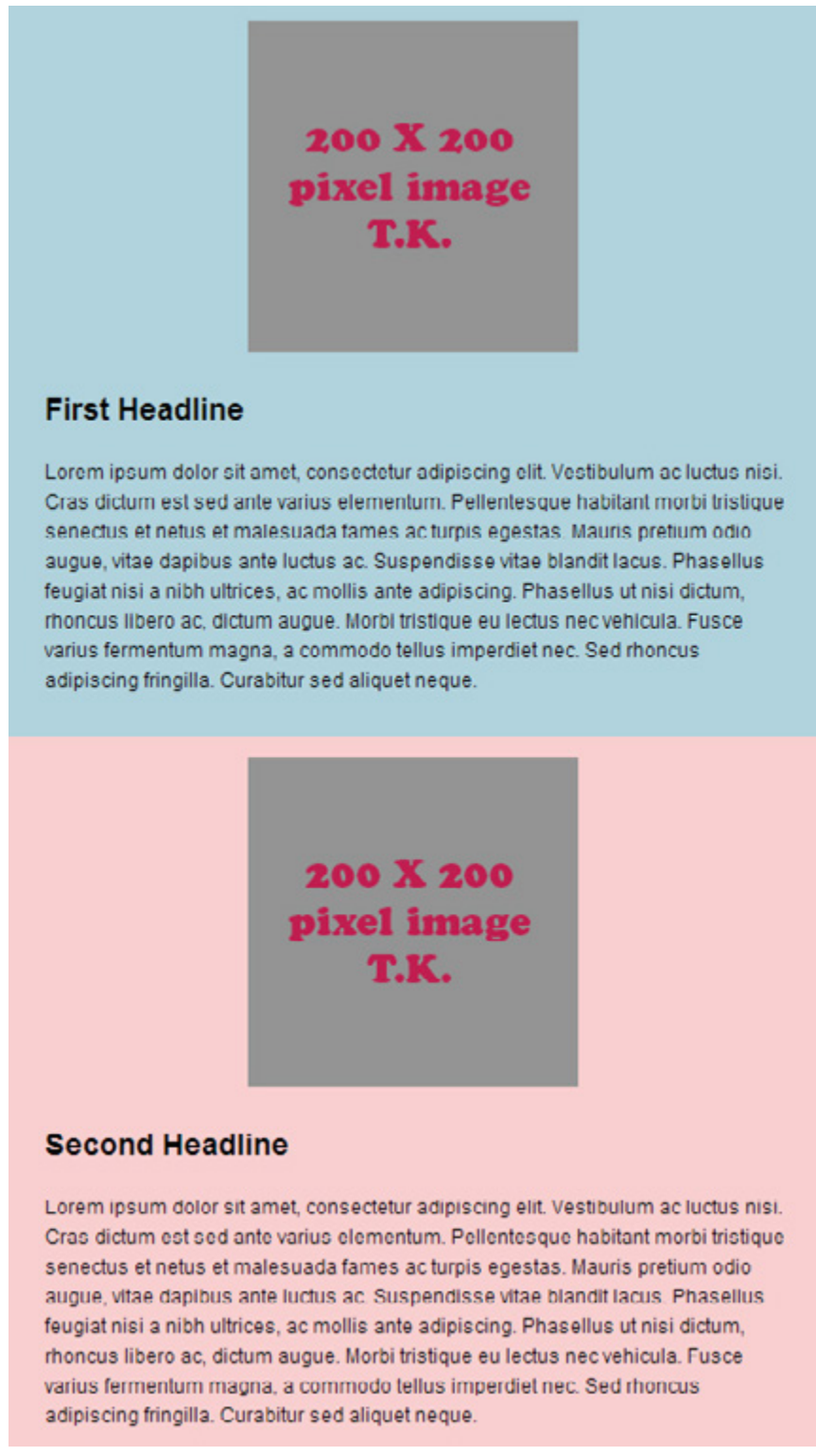
Tables can create problems for responsive design, because as powerful as the media query is, it can never place the contents of one column directly above the contents of another in the same table. Likewise, a table with two rows and one column are always going to keep the contents of the first row above the contents of the second.

Here, for example, is an email layout using a single one row, two column table. The images are placed alternately in the right or left columns next to the text.

In this example we've included the "container" media query in the file, which should take effect when it detects a screen size less than 450 pixels, but here's what happens when the width is reduced below 450 pixels:

On the left, the media query has no way to change the positions of the text and the images, so it does the only thing it can—it scrunches everything up.

Now here's the same layout where the images and the text block have been set up as separate tables, and reduced to the smaller width:



This time everything is as it should be. The image moves above the text and centers.

Normally, tables are intended as restrictive structures that place text and image in specific relationships to each other. That is one of the reasons tables became such a popular technique for controlling image positioning in early web design. But when working with responsive design, it's better to think about a table as an individual container rather than a rigid matrix for different elements. That's not to say there aren't times when you'll want multiple rows and/or columns in a table in responsive design—and we'll get to some of those examples a little later—but it does mean that when elements need to change their relationships to each other responsively, you're usually better off inserting these items into their own, individual, single-celled tables.

If we want to create a layout in which the text on the right moves under the image on the left on a small screen, we first must place each of these elements in separate tables, each consisting of a single cell. By expanding or contracting these containers, we can control their relationship to each with a high degree of accuracy.

In the first example on the previous page—where the image and text block are in separate columns in the same table—there are no styles that you can apply to change the relationship of the elements to each other. All the style and class information is in the file, and is the same as it is in the responsive version, but the fact that the image and text blocks are in the same table pins them in position next to each other. The text will always be either to the right or the left of the image according to which column it is in.

In the responsive version, the image is in its own table as is the text block, and both of those tables are within another table that contains the background color (blue in the top section and pink in the bottom one). In this way the elements are free to move based on the attributes you assign each of their tables. If the screen is

“When working with responsive designs, it’s better to think about a table as an individual container rather than a rigid matrix.”

larger than the max-width size listed in the media query, then the image appears to the left of the text in the blue section and to the right of it in the pink section. The only difference between the two sections, besides the background color, is that in the blue section, the table containing the image includes align=”left” as one of its attributes, while in the pink section the image attribute is set to align=”right”. In both sections, the table with the image in it appears in the HTML before the table with the text in it. This is important because we always want the image to appear above the text when the email switches to a small screen. If we placed the image table in the order that it appears (reading left to right) the image in the second table group would appear under the text. The order of the tables in either section as they appear in the HTML is as follows:

1. Container table: This always takes up 100% width; contents centered. It is there to center the email on a device with a larger screen. (Optional)

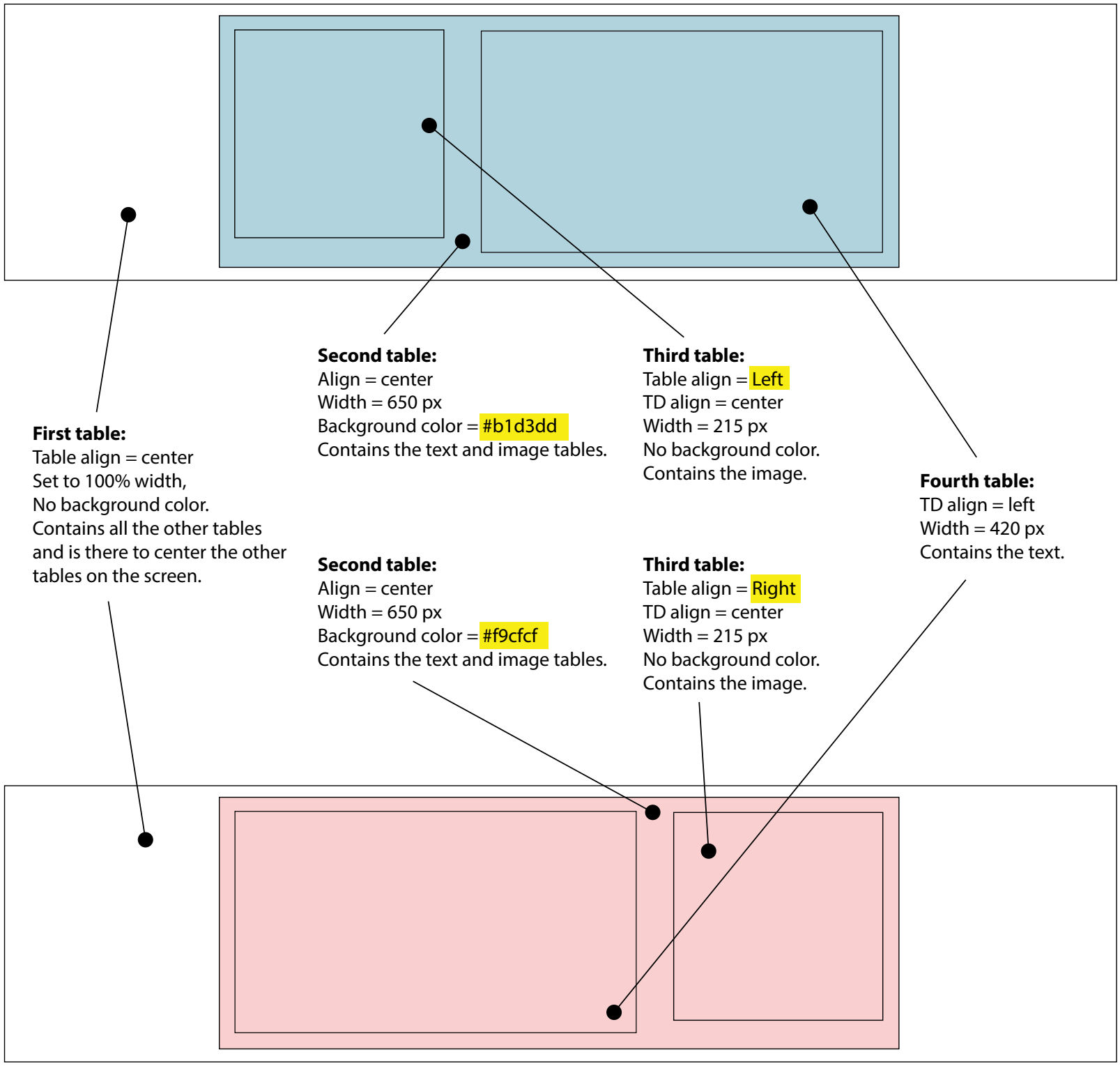
2. Background color table: Set to a width of 650 pixels and aligned to the center. It also includes cell padding to keep the text and image from touching the edges of the table. This provides the background color for the email. It also provides the boundaries within which the text and images must stay. The table is set to center but its cell has no alignment properties.

3. Image table: This table is aligned to the right and to the left in alternate sections. The image is centered within the table’s cell.

4. Text table: This table contains the text. The text is aligned to the left inside this table’s cell. The table has no other alignment properties because it doesn’t need them. The image table is going to determine where the text box falls.

In both cases, the image table is the first element inside of the background color table. This will become important when it is time to switch to the small screen view, as you’ll see. The background color table is set to 650 pixels and aligns to the center. The container table is optional and is there to keep the email from appearing on one side of the screen. You could also achieve the same effect by enclosing the contents in a div with the align=”center” property.

Here is a breakdown of the tables in the responsive example with the differences highlighted in yellow:



A Touch of Class

Now let's look at the HTML for the first section. This consists of the background color table, the image table, and the text table. I've removed most of the Loren Ipsum text and text attributes, and simplified the links to make it easier to read.

```
<table class="container" width="650"
align="center" border="0" cellspacing="0"
cellpadding="12" bgcolor="#b1d3dd">

<tr>

<td>
```

```
<table class=”container” width=”215”
align=”left” border=”0” cellspacing=”0”
cellpadding=”0” style=”mso-table-
lspace:0;mso-table-rspace:0;”>

<tr>

<td align=”center” valign=”top”
width=”200” style=”padding-left: 5px;
padding-right: 5px;”>

<img class=”visual” src=”image-
placeholder.png” alt=”Image.”>

</td>

</tr>

</table>

<table width=”410” class=

“container” style=”mso-table-lspace:0;

mso-table-rspace:0;”>

<tr>

<td style=”padding-left: 10px;”>

<p align=”left” class=”padding”>First

Headline</p>

<p style=”text-align: left;”>Text goes
here.

</p>

</td>

</tr>

</table>

</td>

</tr>

</table>
```

First thing you'll notice is that all three of the tables shown have the class "container" as a style attribute. That class, appears under the media query:

```
@media screen and (max-width: 525px) {
```


```
table.container { width: 100% !important;
} }
```

In other words, when the width of the screen on which you are viewing the email drops to 525 pixels or less, any table of the class "container" changes its width to fill 100% of the screen. This forces the image and text tables to stack on top of each other. Since the image table occurs before the text table in the HTML, the image will appear above the text, regardless of whether it appeared to the right or the left of the text on a larger screen.

More Complex Tables

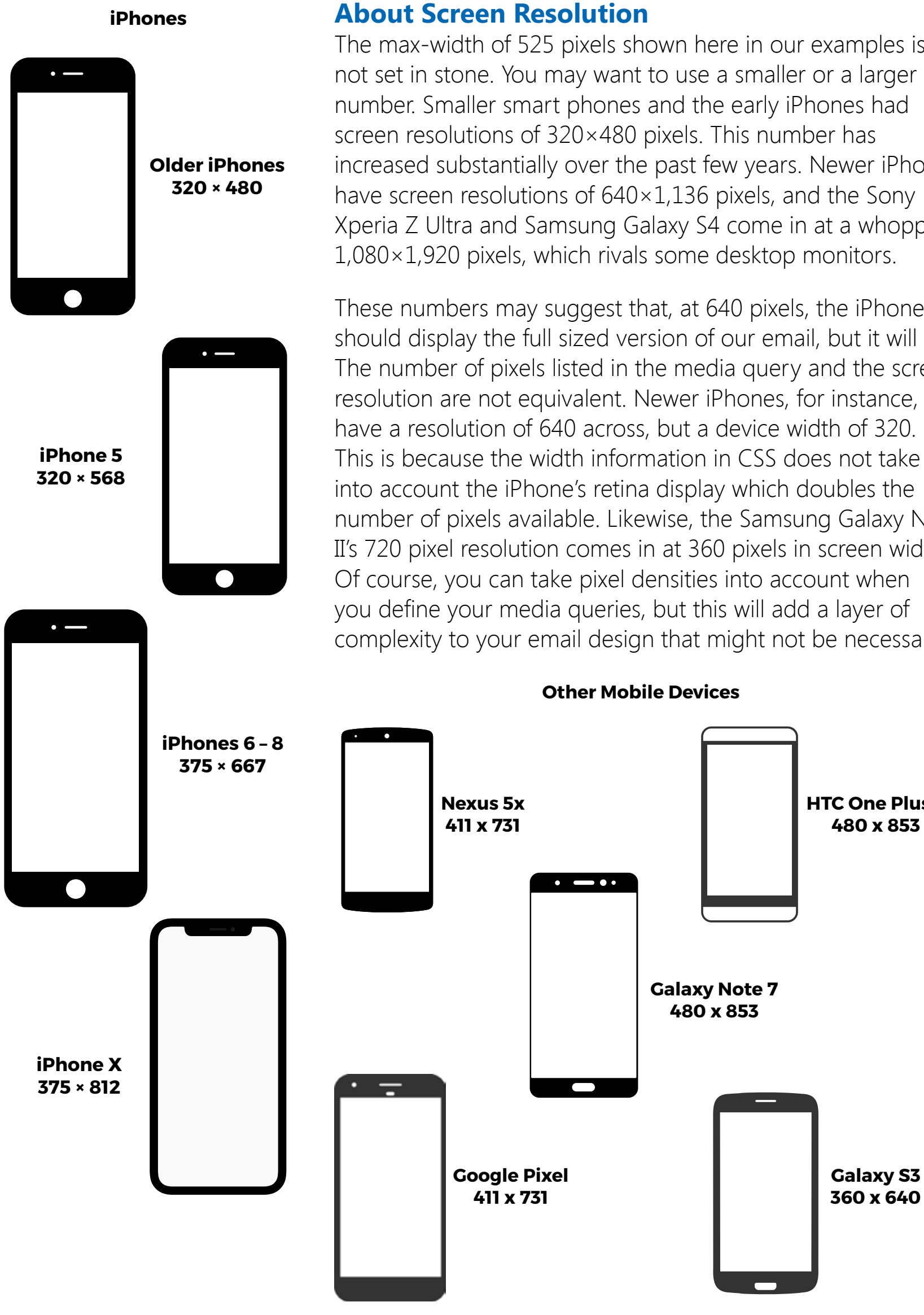
As mentioned earlier, each table does not have to consist of only one cell. Tables can still have multiple rows and columns if desired. For example, if we want to add a headline above the text and a link button below it, we may want to put these in their own rows:

In this way, the headline, text and link button keep a vertical relationship to each other that won't change when the screen changes sizes. The headline and the button are contained in their own tables, which allows us to center them and change the width of the button on a small screen without affecting the alignment of the text as seen in the example on the left.

About Screen Resolution

The max-width of 525 pixels shown here in our examples is not set in stone. You may want to use a smaller or a larger number. Smaller smart phones and the early iPhones had screen resolutions of 320×480 pixels. This number has increased substantially over the past few years. Newer iPhones have screen resolutions of 640×1,136 pixels, and the Sony Xperia Z Ultra and Samsung Galaxy S4 come in at a whopping 1,080×1,920 pixels, which rivals some desktop monitors.

These numbers may suggest that, at 640 pixels, the iPhone should display the full sized version of our email, but it will not. The number of pixels listed in the media query and the screen resolution are not equivalent. Newer iPhones, for instance, have a resolution of 640 across, but a device width of 320. This is because the width information in CSS does not take into account the iPhone's retina display which doubles the number of pixels available. Likewise, the Samsung Galaxy Note II's 720 pixel resolution comes in at 360 pixels in screen width. Of course, you can take pixel densities into account when you define your media queries, but this will add a layer of complexity to your email design that might not be necessary.



How Many Queries?

You can create as many different media queries as you like. You may be tempted to create an intermediate set of rules for tablets and the very large phones ("phablets"), but keep in mind that each of these will add another media query and another set of instructions to your head tags. As a rule, we don't recommend additional intermediate media queries for these devices. If properly designed, your email should be readable on these devices without requiring any responsive adjustments. An email with a max-width of 650 pixels wide with 13 point type should be readable on any tablet (some people may prefer to use the more cautious 600px width and 14 point type), even in portrait mode.

You may prefer to add a media query for tablets if your email is unusually wide and contains several columns, but an email with this structure might also have trouble displaying properly in browser-based email clients such as Yahoo and AOL that use up a lot of the available screen real estate for advertising and navigation bars. Tablet settings make sense for web pages where you have complete control over every aspect of the page. You may want to make your web pages adjust to several different screen sizes. But this just isn't the case with email.

There are other techniques you can use to take your email to the next level but be advised that the more complex you make your mailing, the more likely it is to have something wrong with it. It is also much harder to find an error when your email body copy changes from 50 lines to 500.

Presto Change-o

In the last chapter, we touched briefly on the ability of responsive design to make images disappear when the email reaches a certain size, so you may be wondering: Can I use this feature to switch an image out with a smaller image when the screen size reaches my media query screen max-width value? After all, the basic premise is simple enough. In our original example we created the class "HideOnPhone" with display:none as its sole declaration. This did a good job of removing the image, but how about if we create a second table or table cell that has the value of display:none already set in the body of the email, and switch that out for different values in the media query. Sounds simple enough, doesn't it? Too bad it won't work.

In all likelihood one of two things will happen. In the first instance you will display both images. This will happen with email clients that choose to ignore the display:none declaration (at one time, Gmail is the worst offender in this regard, but this is no longer true). A more likely result is that the second image simply won't display at all, even if the email client is otherwise compatible with media queries.

There is a way to achieve the switch without worrying about double images. It requires the use a background image, which some email clients will ignore. It would be nice if we could simply turn one image off while turning another one on,

but this won't work—especially when the email client is not responsive design compatible. By using the background image technique, you can change the image when it's possible, yet still have the original image display correctly where responsive design isn't an option.

To accomplish this, first let's take a look at the image we want to appear in the large display version of the email:

This image looks fine on a monitor, but once the picture gets down to the size of an iPhone's screen it becomes difficult to tell what we are looking at. For this, we are going to use the following image:

This image is radically different from the original, but it serves well to demonstrate the dramatic effect possible with image replacement.

First, let's place the larger image in the email. As with other aspects of our design, we will place it inside of its own table:

```
<table class="container" width="650"
align=

"center" border="0" cellspacing="0"

cellpadding="0" bgColor="#524b41">

<tr>

<td class="banner">

<img style="font-family: Impact, 'Franklin
Gothic Bold', Haettenschweiler, 'Arial
Black', sans-serif; color: #FFFF00; font-
size: 36px; text-align: center;" alt="Need
a New Car?" src="bannerimage_large.jpg"/>

</td>

</tr>

</table>
```

You'll notice that we've also given the table a background color that matches the predominant color in the image, and added styling to the alt tag, so its appearance approximates that of the text in the image. [Note: For more information on how to create styled alt tags, see our white paper: ***Using Text and Image—Designing Email for Maximum Impact.***]

The table class is still "container," although in this particular instance it won't have any impact on the email's appearance and can be omitted if you prefer. We've also added a new class to the td cell: "banner." This is where the magic will happen, but first we need to add the following selectors to the media query:

```
td.banner img { display: none; }

td.banner { width: 525px; height: 317px;

background: url(bannerimage_small.jpg)

no-repeat 0px 0px; }
```

These will affect any td cell assigned the "banner" class. The first expression contains the "display:none" command, which turns off the image that is currently inside of the cell. The second

expression adds the URL for the background image. The width and height are set to the actual width and height of the background image. Since it is a background image, we need to tell it that we don't want it to tile. This really isn't much of a concern here, but we've added the no-repeat declaration just to be safe.

The 0 pixel coordinates at the end of the declaration group are there to indicate that the image should start in the upper left corner of the image box. This could also have been written as "left: 0px;." As we reduce the width of the window, the right side of the image will start to disappear, but because it is set as a background image, scroll bars do not appear. This is the reason we've placed the headline in the upper left corner of the image. In this way, even on the smallest device, the headline should still be visible. If we had put the headline in the lower right corner of the image, we could have replaced the first occurrence of "0px" with "right," which would cause the image to move out of the cell on the left side first.

Importing Fonts

Although not part of a media query, @import acts in a similar way by letting you expand your design options past the usual limits. In this case, import lets you use fonts that are not installed on your computer. It is similar to the @font-face rule that already existed, but it is more versatile and works in more cases than font-face does.

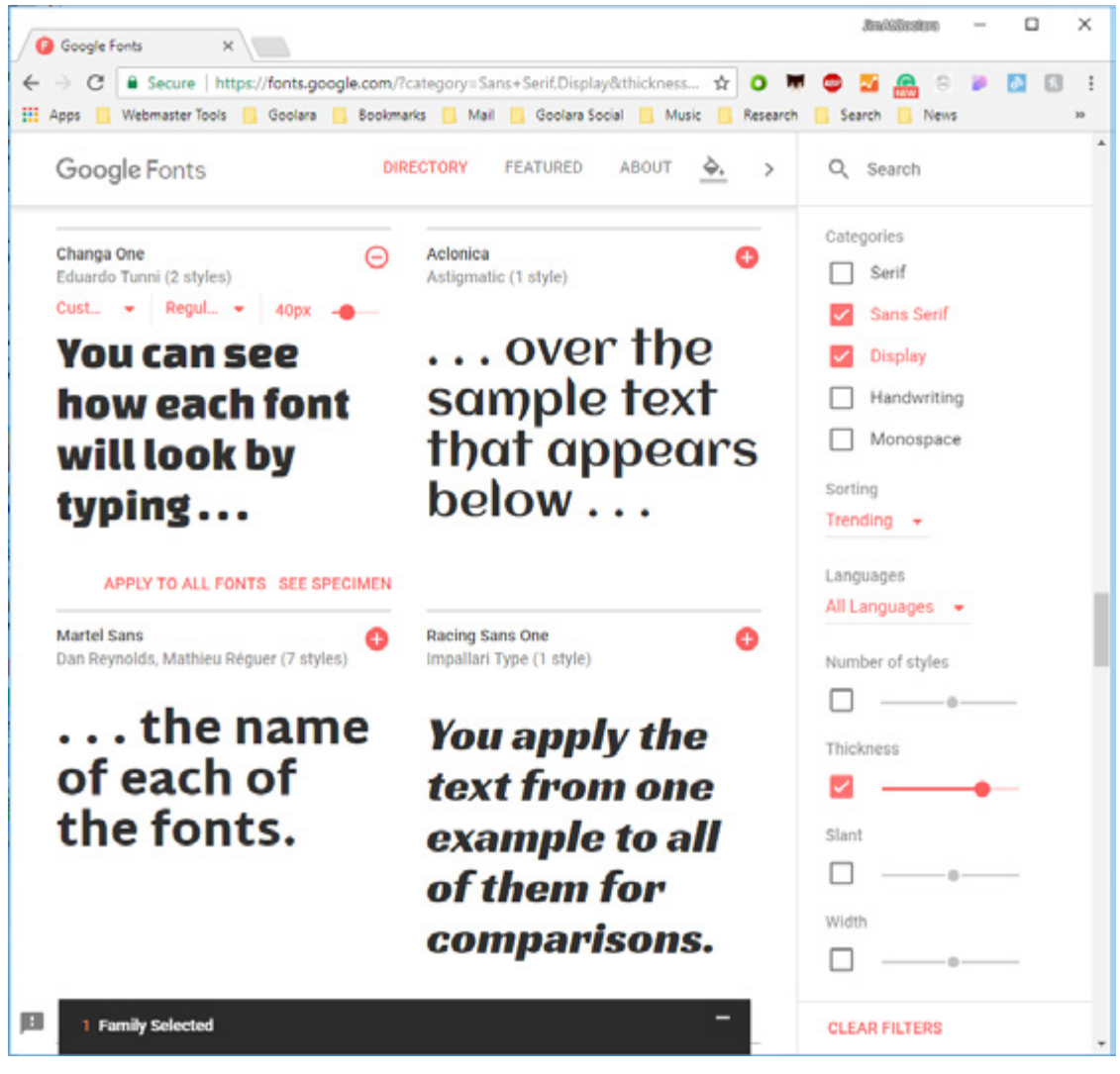
To do this, you can either subscribe to Adobe's TypeKit service, which will allow you to use many excellent fonts designed by professionals, or you can use Google Fonts, which are free, and are made by both professional type designers and amateur font enthusiasts. Before you get started with this process, however, you should be forewarned that the use of outside fonts in email is extremely limited. Most email clients will simply ignore requests to import fonts, and if you've built your design around one of these fonts, you might be in for a rude awakening.

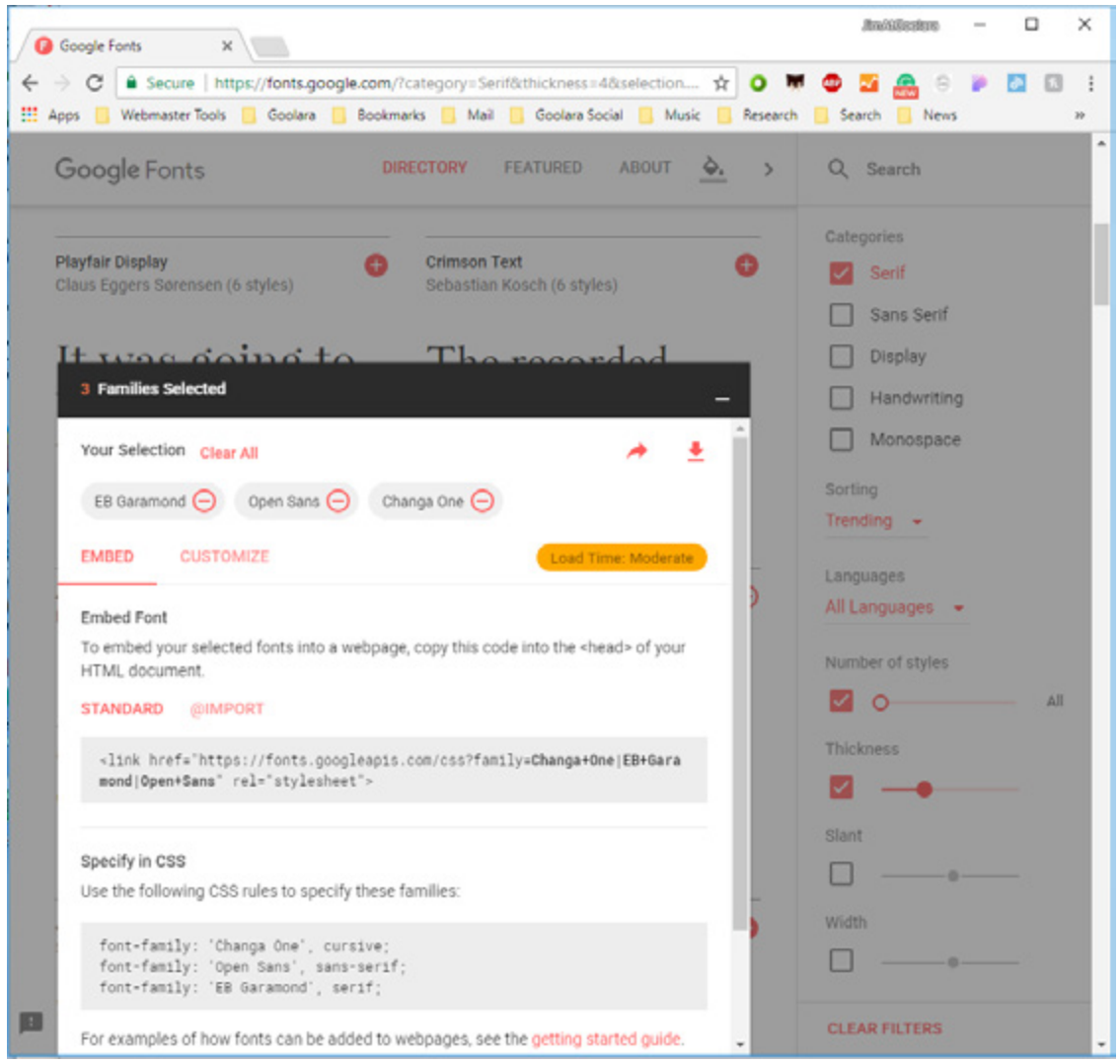
For the purposes of this guide, we'll be using Google's font library.

At the Google Fonts site (https://fonts.google.com/) you can choose from over six-hundred possible fonts, and the number is growing every day. If you know which font you want to use you can go to it directly, or limit your search using the filters to the left of the font display.

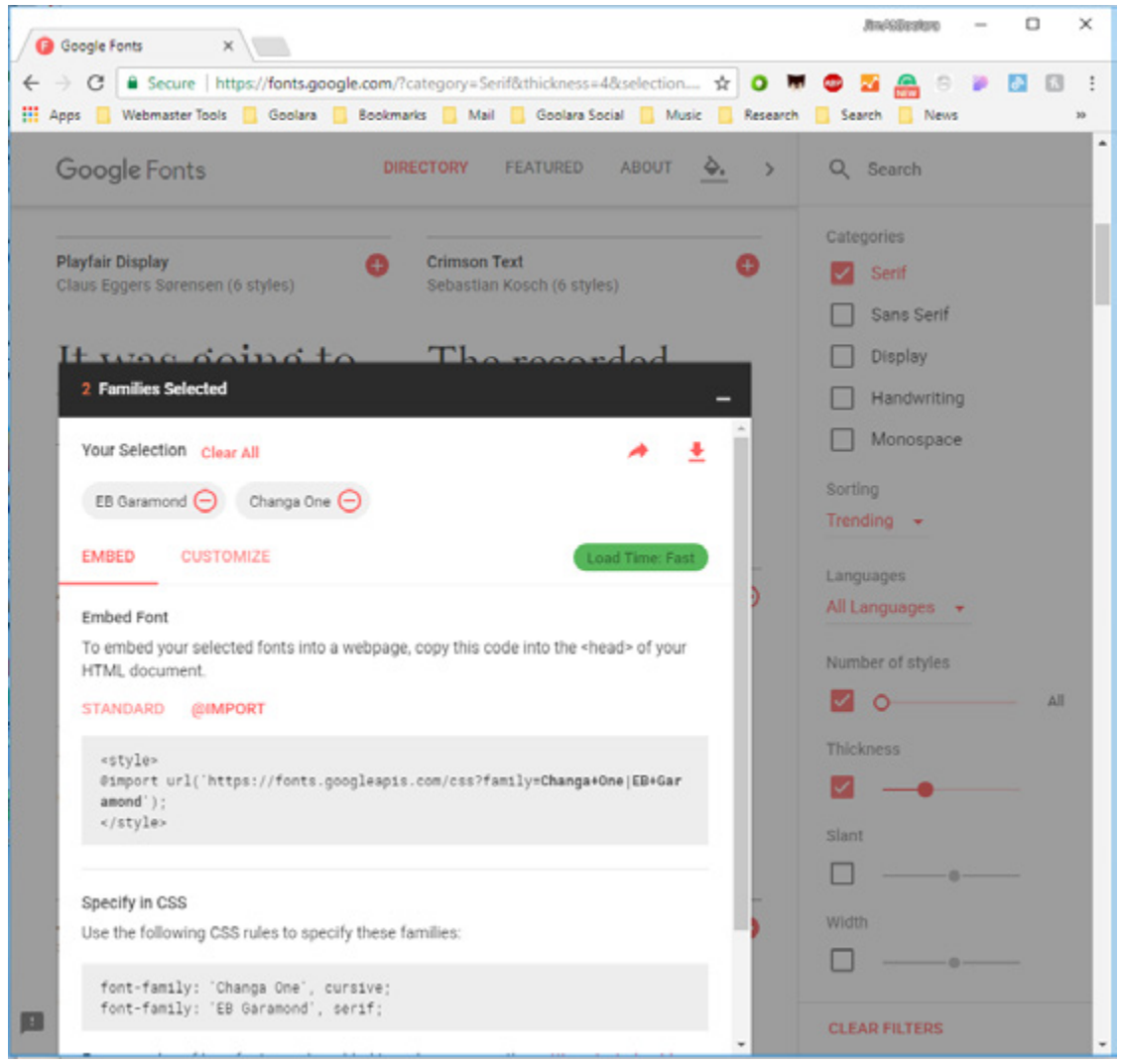
In our case, for instance, we know we want to use a sans-serif bold font. We'll uncheck everything in the "Categories" filter except for sans-serif and decorative (just in case there is a sans-serif decorative that might work). You can also reduce the number of choices by filtering the types based on their thickness, slant, width, and number of available of styles; and sort the fonts based on popularity, or what is trending at the time. In the example, we have increased the thickness number so that only the thicker typefaces will be displayed. In this case, we want to use "Changa One," which appears on the upper left in the image below.

By clicking on the plus sign to the right of the typeface name, we add that font to our selection. A black box appears on the bottom of the window to show how many fonts we have selected so far.





Continue choosing fonts by changing the filters based on each category of typeface you wish to use. Once you've selected all the fonts you'd like to use in your mailing, click on the "Families Selected" box at the bottom of the screen. This will open the pop-over listing your font choices. On the right, the approximate load time is shown. This indicates the impact that all the fonts you've chosen will have on your email load time. In the example above, the three chosen fonts will result in a moderate load time, which is longer than we want. Email needs to load quickly. By clicking on the minus signs next to typeface names, you can remove fonts from the list. In this case, we'll remove the Open Sans from the list, which gives us a faster load time rating.



Under "Embed Font," you have a choice between "standard" and "@import" for using your fonts. Our tests have shown that both of these will work as long as the email client will accept imported fonts and image display is enabled. For our example, we'll be explaining the "@import" version, although the technique for both is essentially the same approaches.

First copy the style information (or link information in the standard technique):

```
<style>

@import url('https://fonts.googleapis.com/
css?Changa+One|EB+Garamond');

</style>
```

Place this below the @media query. The @import information should have its own style tags as shown above.

Next, copy the CSS font-family attributes and place them inline in the body copy based on how you want the type in that paragraph or header to appear. In our example, we're going to modify the information for the "Changa One" font slightly because we do not want our headlines to default to cursive.

We would rather have them default to Arial or san-serif. Here's what the information for the first headline looks like after we're finished:

```
<p align="left" class="padding"
style="padding: 0px; text-align: left;
font-family: 'Changa One', Arial, sans-
serif; font-size: 24px;">

First Headline

</p>
```

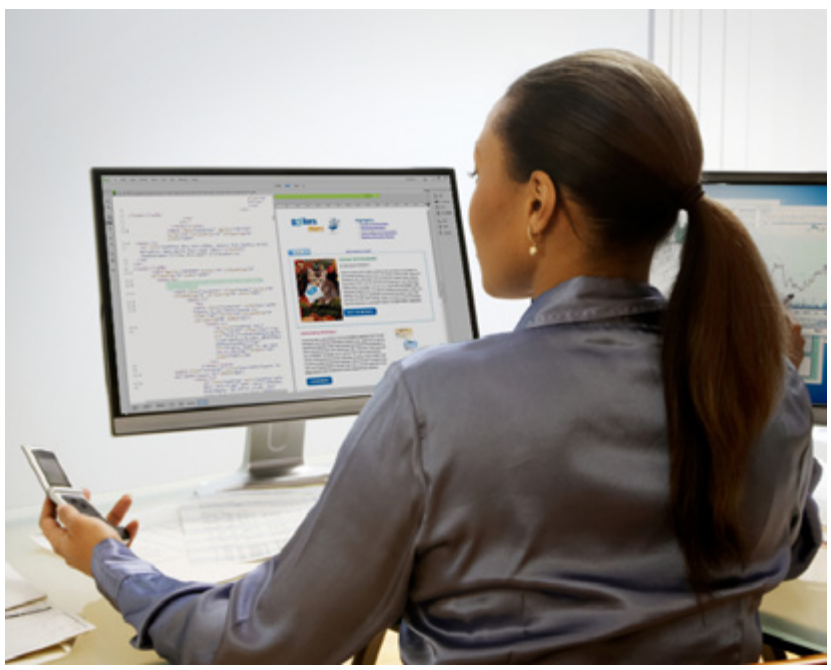
And here's the body copy:

```
<p align="left" class="padding"
style="padding: 0px; text-align: left;
font-family: 'EB Garamond', serif; font-
size: 24px;">

Body copy goes here.

</p>
```

The @import and standard link methods both require your email client to get information from outside of the email. As a consequence, the email clients that can use these features treat them in the same way they treat images. If you have image display turned off, the web fonts won't display either. Once you've clicked "Display Images," the fonts will also appear.

Right now, the import function only works in a few places. These include the Apple Mail, Thunderbird, and Outlook versions 2000 and 2011. Gmail, Yahoo, AOL, and most other email clients will default to their standard fonts. As with your basic email layout, if you use imported web fonts, you'll have to make sure that the inline style information still yields an acceptable alternative to these typefaces.

USING DREAMWEAVER CC



For people who prefer to use Dreamweaver to produce their email HTML, another obstacle to facilitating responsive email design has been the lack of an easy way to create the type of all-in-one files you need for email. Dreamweaver had plenty of tools for creating responsive design, but it always wanted to produce separate CSS and JavaScript files to go along with the HTML.

In May of 2013, Adobe announced a new product called Edge Reflow that was intended to help with the responsive design process. Unfortunately, however, the software was not created with email design in mind. To make matters worse, it wouldn't open HTML documents, and used its own Reflow (.rflw) format exclusively. After a few months, the product was abandoned, and Adobe Systems started adding features to Dreamweaver specifically intended for the email designer.

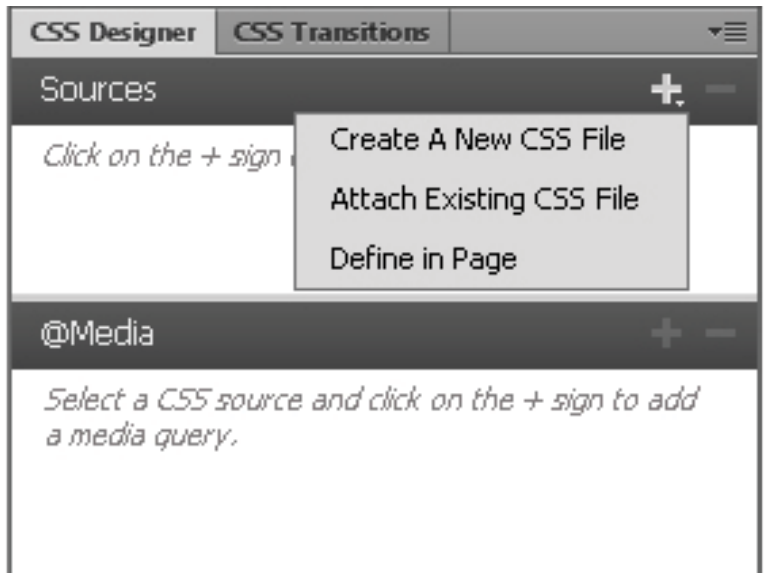
If you are new to HTML, CSS, and responsive design, the easiest way to get started is to use the responsive email template that is built into Dreamweaver.

How to create media queries for email in Dreamweaver CC

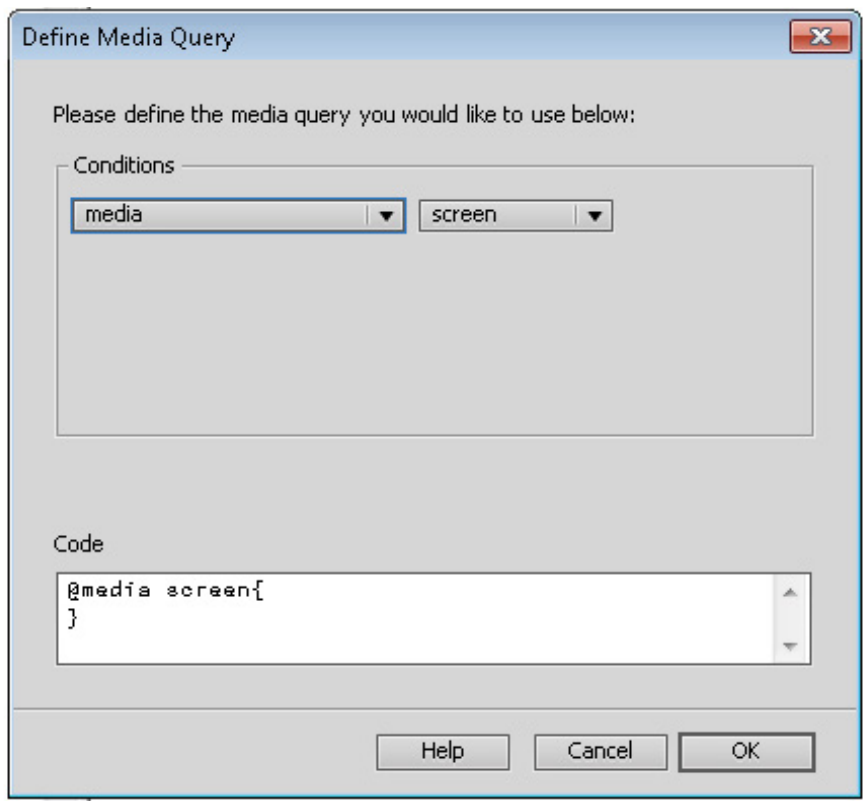
1. Create a new HTML document in Dreamweaver CC

2. If it is not already open, select the CSS Designer tabs from the Windows menu, or press Shift + F11.

3. In "Sources" bar, click on the plus sign on the right and choose "Define in Page." This will automatically insert your style tags between the document's head tags.

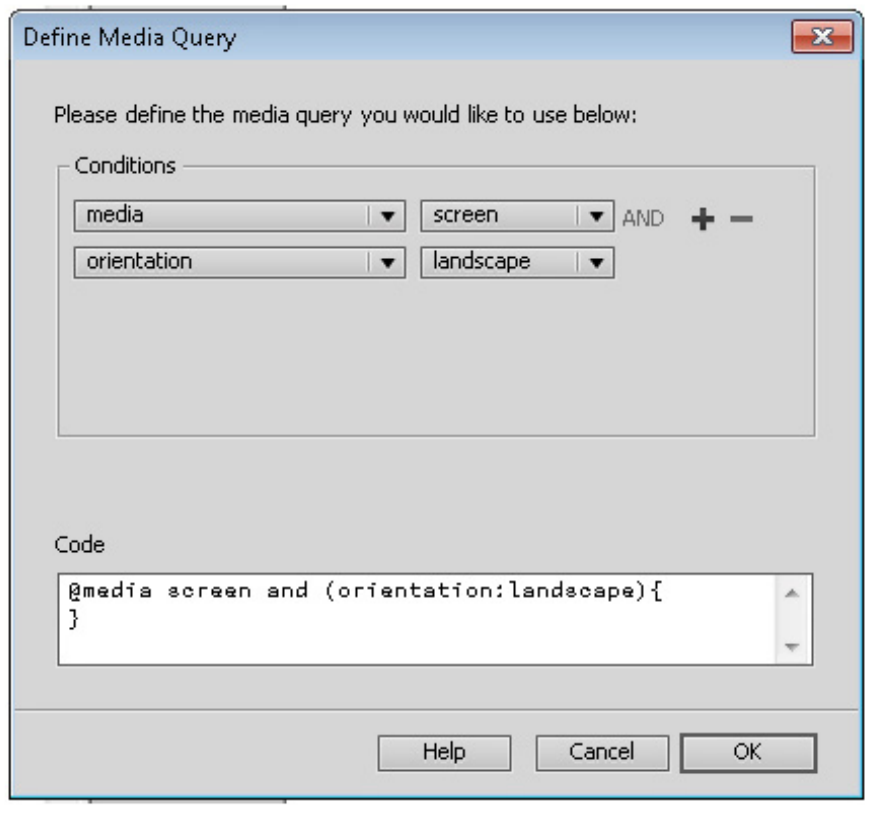


4. Under the @media bar, below the Sources bar, click on the plus sign. A "Define Media Query" pop-up window will open.

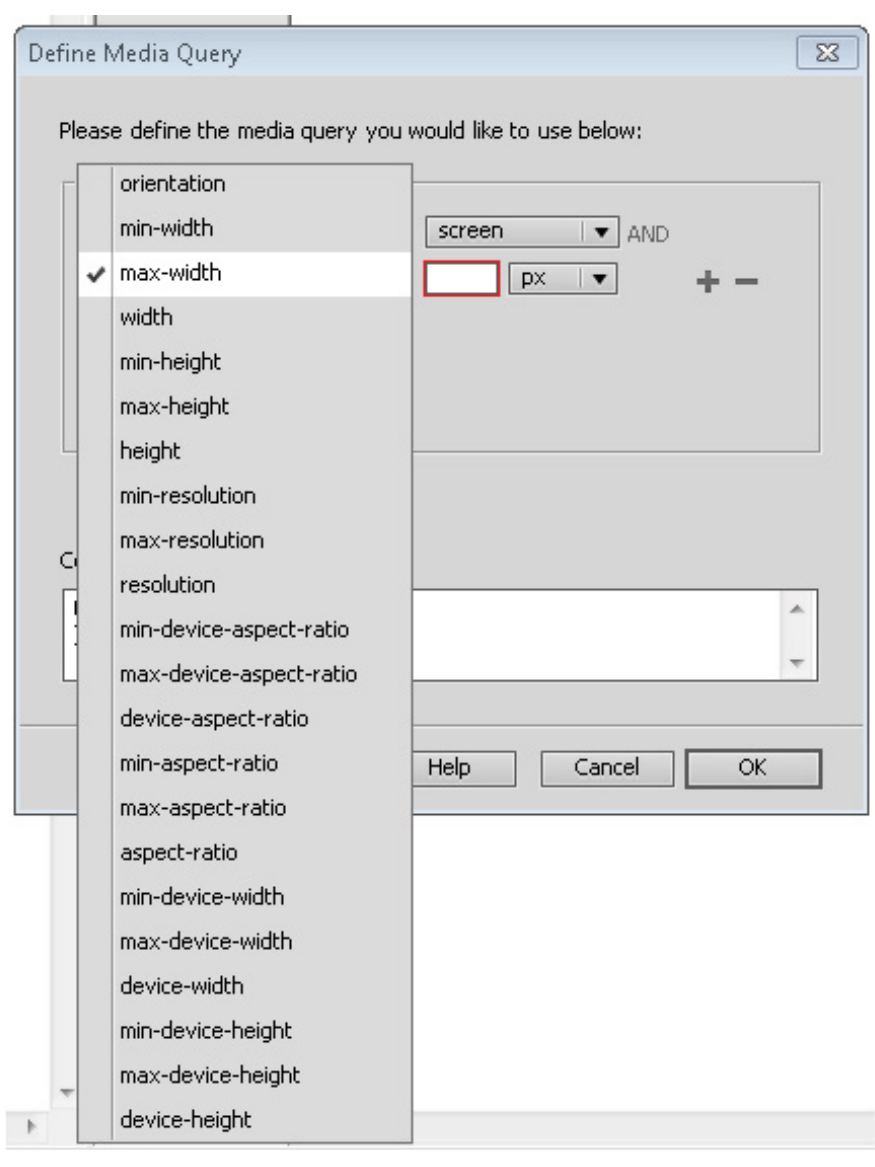


The media query property defaults to "screen," which is what we want, but if some other property appears, you can change it with the drop down menus, or in the small code window at the bottom.

5. Move the mouse to the right of the screen property. A plus sign should appear. Click on the plus sign. A second condition statement appears.



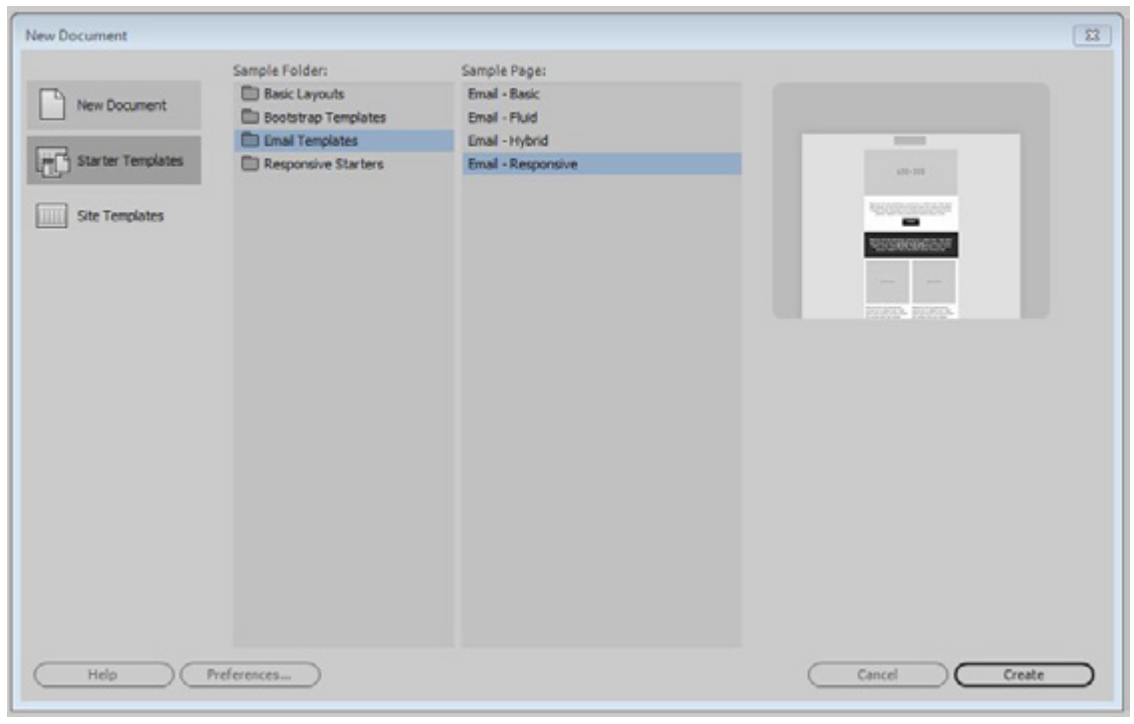
This may default to the first condition in the list. If so, click on the condition and a pop-up menu appears.



6. Choose “max-width” and the condition and set the value to 525px.

Now you have the basic construction for your media query. If you like, you can continue working in the CSS Designer by selecting the media query and adding classes under the selector tab. You can then adjust the properties of each of these classes in the properties tab, but honestly, it is just as easy to enter these properties and attributes directly in the code at this point.

If you want to make the process even easier, you can use the responsive email template built into Dreamweaver. You will, of course, be limited this way to the parameters set by Adobe, but it is a good way to get used to responsive design and test how it works. To do this, Go to the File Menu and choose New. Then select Starter Templates → Email Templates → Email - Responsive, and click “Create.”



IT'S THE LITTLE THINGS THAT MATTER

This chapter includes the little potential pitfalls that you are likely to encounter when you work on a responsive email design. Most of these have to do with Microsoft Outlook, which continues to play by its own rules. It is not hyperbole to say that Outlook can double the time it takes to develop a responsive email design that will work across all versions of this software.

If you look at the style information for nearly any email, you'll see a few things that may not make sense. This is because there are still some differences between how the various browsers handle HTML. In some cases, they don't even accept the same commands. Nonetheless, there are a few little things you can do to make your email as compatible as possible. This chapter is devoted to those little things, starting with the first few lines you'll see in most responsive email:

```
.ReadMsgBody {width: 100%;}
.ExternalClass {width: 100%;}
body { margin: 0; padding: 0; }
table {border-collapse: collapse;}
text-decoration: none;
```

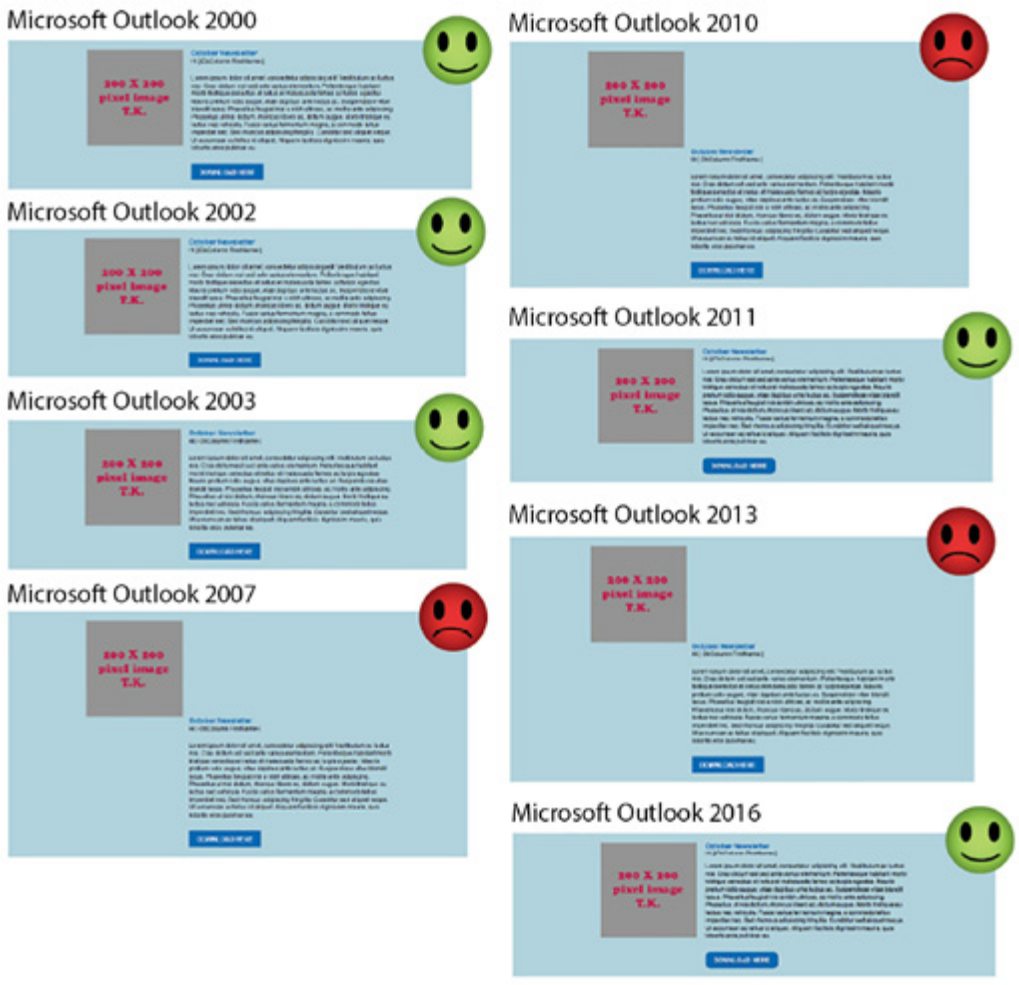
These are placed above the media query as the first rules in the email's styles. These are primarily there to help avoid unwelcome formatting variations when the email lands in different email clients.

We've also seen some email include the following additional bits of code to deal with the iPhone's tendency to change font colors and add underlines to things like dates, addresses and links:

```
.appleBody a {text-decoration: none;}
.appleFooter a {text-decoration: none;}
.appleEvents a {text-decoration: none;}
```

The Outlook Dilemma

By far the biggest stumbling block in developing responsive email is Microsoft Outlook. For reasons known only to Microsoft, they've changed how Outlook works with nearly every iteration of the program. As an example, our test newsletter template came out with the following results:



Microsoft's Office 2007 was part of Microsoft's attempt to offer better security features, and it featured some major changes in the way Outlook handled email information, so the sudden shift from working to not working was understandable. However, why the 2011 version of Outlook formatted properly, when neither 2010, nor 2013 did is anyone's guess. As you

“There’s really only one way to make sure what you send out looks good across all platforms and that’s to test it.”

finalize and test your designs, you’ll find that Outlook is often the last stumbling block to a final design.

But take heart, there are a few things you can do to alleviate some of the problems with Outlook. The foremost problem with Outlook is the alignment of tables. Two tables that sit nicely next to each other in Gmail and Thunderbird, are staggered in some versions of Outlook. You can solve most of these alignment issues by adding the following bit of code to any tables that requirement alignment:

```
style=”mso-table-lspace: 0; mso-table-
rspace: 0;”
```

So, for instance, the code for the image in the first section should read:

```
<table width=”215” align=”left” class=

“container” style=”mso-table-lspace: 0;

mso-table-rspace: 0;” border=”0”
cellspacing=”0” cellpadding=”0”>

<tr>

<td width=”215” align=”center”
valign=”top” style=”padding-right: 5px;
padding-left: 5px;”>

<img class=”visual” alt=”Turn on display

images to see what you’re missing!” src=

“ImageLocation/image1.jpg”>

</td>

</tr>

</table>
```

Also keep in mind that while Outlook does a better job than most at handling HTML5 commands, it still shares with its competition the preference for table alignments over divs.

At this point, you may think you’re through, but there are so many potential pitfalls when it comes to responsive email design that there’s really only one way to make sure what you send out looks good across all platforms, and that’s to test it.

Email clients

Browser-based

Gmail

Yahoo

Outlook.com/Hotmail

AOL

Desktop-based

Outlook (2003-present)

Thunderbird

Apple Mail

Mobile Devices

iPhones

Samsung phones

iPads

Android Tablets

To avoid unwelcome surprises, you should always test your email across as many clients and devices as possible. This is true whether you are creating responsive email or not, but it is much more critical if you've used media queries. First of all, not all email clients will recognize or use the media query information. If that is the case, does your email still look good? This becomes more and more critical as your design becomes more complex.

Most email marketing software—Goolara's Symphonie included—have the ability to show you what your email should look like in various email clients. These will give you a good idea of what to expect, but never confuse them for the real thing. There are so many idiosyncrasies with each email client that the only way to be absolutely sure your email is okay is to run it in various email clients, across all browsers and devices. Since this is usually not possible, there are a few alternatives that will help you get your design pegged.

There are some services that provide renderings based on the actual email clients, both stand-alone and browser-based. These can be very helpful, but keep in mind that although they reportedly use the actual rendering engines to do this, you will sometimes still find disparities between their results and what you see on the screen. In a recent test, we found that, in one instance, the rendering service displayed an acceptable email for Outlook 10, where the actual email still came out wrong on the desktop.

The safest approach is to get one of everything, but this is an expensive proposition. Barring that, the combination of a rendering services and a few actual test emails to different platforms is a good idea. Ideally, you'll want to send a test email to all the email clients shown in the list on the left.

It's possible that your recipient list does not include anybody using an iPhone, for instance, but keep in mind that the email might be forwarded to someone who does, so it is still a good idea to make sure your mailings look good across all clients and devices.

ASSESSING VALUE VS. COST

The question remains: Is Responsive Email Design worth the effort? Although this guide should go a long way toward helping you work with responsive design in email, the fact remains that responsive design is not for everybody. There are a few things to consider before embarking upon a shift to responsive design in your email marketing efforts.

Standard Template vs. Unique Every Time?

The first time you design a new email responsive format there is inevitably a certain amount of trial and error and testing that you have to do to get it to look right across all browsers. Once that's done, using that same template in the future will be easier. A company that does not use standardized templates, but prefers instead to change the format with each new mailing, is probably not a good candidate for responsive email design. With this approach, each email is like starting from square one. Everything has to be thoroughly tested all over again, and new problems are sure to crop up.

Another approach is to use a modular design, much like the newsletter template used in the examples in the previous chapters. If each section is independent of the next, you can maintain some section designs and replace others. In this way, you're not spending every month reinventing the wheel, but you can still insert enough variety into the mix to keep things interesting.

Skill Level

It takes very little skill to put together a simple email. Some knowledge of HTML and familiarity with the features of your email marketing system, and you can put together an email in no time. A responsively designed email, on the other hand, demands much more care. You have to sit down and think it through ahead of time. What will each part of the email do when the screen changes sizes? How many attributes do you

"You need to ask yourself whether the effort you put into creating a responsive design is taking time away from things you could be doing to make your mailings more personal."

want to change? Do you want the images to keep a fixed size, or shrink as the viewing window becomes smaller? Do you want images to disappear or be replaced with other images? Once you've answered all these questions, you still have the task of creating an email that will act on your decisions. Then you have to debug the final code because, inevitably, there will be things that don't work as they should.

The designer who is familiar with responsive web design will have a slight edge here, but only a slight one. With responsive web design, you have the option of using separate JavaScript and CSS files. The email designer has no such luxury. Everything has to be contained in the email. A web designer is also more likely to rely on divs instead of tables, which can be difficult to get to behave properly in responsive designs.

If your design team is completely new to responsive design, then this must be factored into your expenses. No matter how good they are, they won't be able to hit the ball out of the park on the first try. If you're working with people who have done responsive email design before, they'll be able to push out their work quicker and with fewer mistakes. At this time, that pool of designers is a small group, but growing every day.

Weighing the Options

In the end, the question you need to ask yourself is whether the effort you put into creating a responsive design is taking time away from things you could be doing to make your mailings more personal—things like segmentation and dynamic content. Larger companies, where the design is separate from the coding, won't find this to be much of an issue, but if your designer is also the person who codes the file for email purposes, you'll need to take into account the added time that they may require to get everything to work correctly. If it comes down to a choice between a nice responsive design and the good use of dynamic content, we recommend dynamic content preparation as a better use of your time.

An email that addresses the specific interests and needs of a recipient is going to have more engagement than one that is merely easier to read on a phone. Every day, more and more people use their phones and "phablets" to read their email, but they're still more likely to respond to an email that is personally relevant to them over one that merely looks good on their phones.

goolara

SUMMARY

The hottest topic of discussion in the email marketing field right now is that of responsive design. Proponents say that responsively designed email performs better, gets more clickthoughs, and that surveys indicate people respond to them better. Opponents say that responsive design performs only marginally better, and the amount of work it takes to get a responsive design up and running might be better spent in other areas of email marketing.

When you are designing email that can be read on mobile devices, there are two approaches you can take: scalable and responsive. With scalable design, the images and text shrink on smaller devices to fit the screen size. This can render them unreadable. With responsive design, the text and images adjust their sizes and widths according to a specified screen width. Responsive design does not work in all email clients or on all devices. Whether you use responsive design or not, you should still make sure that your email is readable under all circumstances.

Responsive design is created using the @media query in the email's style tags. The most common technique is to assign a max-width between 500–600 pixels. Inside the @media declaration brackets, you create classes for the tables in your layout. Although you can use divs, tables are a much safer option for email because they are compatible across more email clients and platforms than divs.

You can also use media queries to replace images by tricking it into displaying a background image instead. A few email clients will also support the use of web fonts using the @ import rule at the beginning of your styles. However, this is only supported in a few places.

The latest version of DreamweaverCC now includes the ability to create and define media queries for email, which makes this program more useful to email designers, but you'll probably still need to do some tweaking to the HTML to ensure everything is working correctly.

A few additional lines of code will help ensure that your email design won't break across browsers. These include the following:

```
.ReadMsgBody {width: 100%;}
.ExternalClass {width: 100%;}
body { margin: 0; padding: 0; }
table {border-collapse: collapse;}
text-decoration: none;
```

The biggest obstacle to successful responsive email design is Microsoft Outlook, which often has problems displaying responsive designs correctly. Many of the problems with Outlook can be eliminated or alleviated by adding the following bit of code to your tables

```
style="mso-table-lspace: 0; mso-table-
rspace: 0;"
```

Testing is crucial to successful responsive email design. There are online email rendering services that are very useful, but whenever possible it is always better to test them on the actual devices and in the actual email clients.

Whether it is worth the effort to create responsive email designs for your mailings will depend on several factors, including how often you use standardized templates, and the skill level of your design team. In some cases, the time and money spent developing responsive designs might be better used to enhance your dynamic content capabilities, which is simpler to implement, and can also improve your clickthrough rates.

REVISION HISTORY

September 2017 Revisions:

Section on Dreamweaver CC revised to reflect Adobe's changes to the Dreamweaver interface.

References to Edge Reflow removed (product discontinued).

Information on Gmail revised to reflect the changes to Gmail's CSS implementation.

Specs on mobile phones updated for the latest versions on the Apple iPhone (now current through the iPhone X).

New graphics for the mobile device section.

Information on floats and margins in Outlook.com revised.

Information on using standard and @import font embedding with Google Fonts updated to reflect changes to the Google Fonts interface.

Summary updated to reflect changes listed above.

Address and copyright information updated.

About Goolara

Goolara has been in the email marketing business since 2005. Symphonie, Goolara's premiere email marketing solution is available in on-premise and cloud-based, SaaS deployments. The powerful software features many advanced capabilities, such as full-featured dynamic content, transactional and triggered email, and customizable report generation features. It is easy to use and runs from a browser-based interface using Chrome, Firefox, Internet Explorer, or Safari. Goolara is headquartered in Moraga, California and can be found online at www.goolara.com.

Goolara, LLC
1030 Country Club Drive, Suite D
Moraga, CA 94556
Telephone: (510) 522-8000
(888) 362-4575
Fax: (510) 522-2457

www.goolara.com