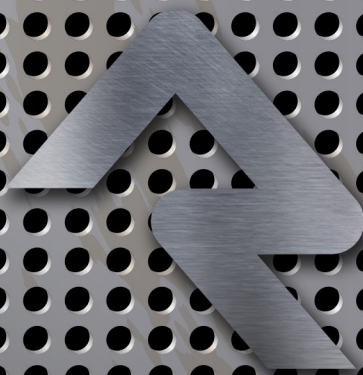


EMAIL TEMPLATE

SURVIVAL GUIDE



Welcome

You know HTML and have developed some websites so you're thinking, how hard could making an HTML email be? Well... you may want to remove all sharp objects from your desk. Seriously. We'll wait...

While browser standards have made developing for the web fairly consistent, this is not the case with HTML emails. As one example of how bad this has become consider the decision Microsoft made for Outlook 2007. Up until that version Outlook had used Internet Explorer to render HTML in emails. Starting with 2007 they decided to replace that... with... Word. Yes... Word. This decision has not been reversed (as of Outlook 2019). It's not just Microsoft, Google's Gmail does some unholy things with your HTML emails too (though nothing as bad as Outlook).

So, now that your expectations have been set, let's look at how to make an Email Template in Rock.

Writing HTML for Email

It's beyond the scope of this document to explain everything you need to know about writing HTML for emails. Instead, we'll focus on the characteristics unique to Rock. So as not to leave you out in the cold, below are some resources to point you in the right direction.

Getting Started

The best place to learn about the intricacies of HTML email is Campaign Monitor's "Coding Your Emails". This single document covers what you'll need to translate your HTML skills to email.

Inline CSS

One of the key pitfalls you should understand is the role CSS plays in HTML emails. If you're at all familiar with web development, you know that CSS is what you use to style your web pages. Unfortunately, many email clients (most notably Gmail) simply ignore traditional CSS syntax. They'll strip it out and pretend it never existed. Frameworks like Ink get around this by developing templates in HTML/CSS and then running the templates through an inlining process before using them. Inlining takes normal CSS rules and turns them into inline styles (yuck...). Here's an example:

```
Normal CSS  
<style>  
  h1 {  
    color: red;  
    font-size: 20px;  
  }  
</style>  
  
Inlined CSS  
<h1 style="color: red; font-size: 20px;">Title</h1>
```

As you'll see later Rock will do the inlining for you if you'd like. You can enable inlining for individual templates, or you can enable for all emails within the Communication Medium configuration.

You Don't Have to Start from Scratch

While you're welcome to start your template from scratch, most people prefer to start using a framework. One of the most popular is ZURB's Foundation for Emails.

Writing Templates for Rock

So, let's cut to the chase and see how to write templates for use in Rock. We'll assume you're starting with a valid working HTML email template. What we'll discuss below will be the secret sauce needed for Rock's power tools.

Enabling Wizard Support

There are two types of email templates in Rock: those that are used in the communications wizard and those for the simple email send (best used for things like leader toolboxes). For the most part you'll be wanting to create templates for use in the wizard.

In order for your email template to work with Rock's communication wizard you'll need to provide a bit of configuration. First, the wizard will need to know what portion of your template is allowed to be edited (this is where the blocks can be dragged to). You'll define this by using special "div" elements. Let's look at the structure of what is needed. Below is an example:

```
< html email layout / content...>
<div class="structure-dropzone">
  <div class="dropzone">
    <div class="component component-text" data-content="<h1>Email Title</h1>" data-stat
e="component">
      <h1>Email Title</h1>
      <p>
        Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean suscipit, a
rcu
        hendrerit mattis accumsan, quam sem tristique dolor, eget euismod enim torto
r
        sit amet augue. Interdum et malesuada fames ac ante ipsum primis in faucibus
.
        Donec finibus tortor lorem, at malesuada justo feugiat ut. Nam ac pharetra e
ros,
        nec feugiat nisl.
      </p>
    </div>
  </div>
</div>
</ html email layout / content...>
```

In this example, the `structure-dropzone` div tells Rock this is where you can drag new sections to (sections are the icons in the Communication Wizard with one column, two columns, etc.). The `dropzone` div is an actual section. Technically, you don't need to

provide this, but then the people using your template will be required to start by dragging a section. Not very friendly. Finally, the `component component-text` div is an HTML component to use as a starting point. Again, you don't need to provide this (technically, only the `structure-dropzone` div is required), but it provides a nice starting place.

Note that the text you place inside of the `component-text` div is the default content. Feel free to modify this to be whatever you'd like people to see when they start. Also note that you can 'pre-load' as many components in the dropzone as you'd like. In this example we've only loaded one, but feel free to go hog wild.

With one of these sections your template is ready to be used inside of the Communication Wizard... but... you can provide more than one if you'd like. If you want to have several dropzones on your template (say for side bars, footers, etc.) you're more than welcome to wire-up as many dropzones as you'd like.

Additional Components

The example in the section above shows how to add a text component to your email template. In this section we'll show you how to add the other components that you see when creating a new communication in the Communication Wizard.

Image

```
<div class="structure-dropzone">
  <div class="dropzone">
    <div class="component component-image" data-state="component">
      
    </div>
  </div>
</div>
```

Video

```
<div class="structure-dropzone">
  <div class="dropzone">
    <div class="component component-video" data-state="component">
      <a href = " " >
        
      </a>
    </div>
  </div>
</div>
```

Divider

```
<div class="structure-dropzone">
  <div class="dropzone">
    <hr style="height:4px;margin-top:20px;margin-bottom:20px;color:rgb(196,196,196);background-color:rgb(196,196,196);border-style:none">
  </div>
</div>
```

HTML

```
<div class="structure-dropzone">
  <div class="dropzone">
    <div class="component component-code" data-state="component">Add your code here...<
  /div>
</div>
```

Button

```
<div class="structure-dropzone">
  <div class="dropzone">
    <div class="component component-button v2" data-state="component">
      <table class="button-outerwrap" border="0" cellpadding="0" cellspacing="0" width="100%" style="min-width:100%;">
        <tbody>
          <tr>
            <td valign="top" align="center" class="button-innerwrap">
              <table border="0" cellpadding="0" cellspacing="0" class="button-shell">
                <tbody>
                  <tr>
                    <td align="center" valign="middle" class="button-content" style="border-radius: 3px;background-color:#2baadf"><a class="button-link" title="Push Me" href="http://" target="_blank" style="display: inline-block; font-weight: bold; letter-spacing: normal; line-height: 100%; text-align: center; text-decoration: none; color: #ffffff;background-color: #2baadf; padding: 15px; border: 1px solid #2baadf; border-radius: 3px;">Push Me</a>
                </tbody>
              </table>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
```

RSVP

This one gets a little complex. It follows a similar pattern to the other components, but there's a bit more to it.

```
<div class="structure-dropzone">
  <div class="dropzone">
    <div class="component component-rsvp" data-state="component">
      <table class="rsvp-outerwrap" border="0" cellpadding="0" width="100%" style="min-width:100%;">
        <tbody>
          <tr>
            <td style="padding-top:0; padding-right:0; padding-bottom:0; padding-left:0;" valign="top" align="center" class="rsvp-innerwrap">
              <table border="0" cellpadding="0" cellspacing="0">
                <tbody>
                  <tr>
                    <td>
                      <table border="0" cellpadding="0" cellspacing="0">

```

```

0" class="accept-button-shell" style="display: inline-table; border-collapse: separate !important; border-radius: 3px; background-color: #16C98D;">
    <tbody>
        <tr>
            <td align="center" valign="middle"
class="rsvp-accept-content" style="font-family: Arial; font-size: 16px; padding: 15px;">
                <a class="rsvp-accept-link" title="Accept" href="http://" target="_blank" style="font-weight: bold; letter-spacing: normal; line-height: 100%; text-align: center; text-decoration: none; color: #FFFFFF;">Accept</a>
            </td>
        </tr>
    </tbody>
</table>
</td>
<td style="padding-left: 10px;">
    <table border="0" cellpadding="0" cellspacing="0" style="border-collapse: separate !important; border-radius: 3px; background-color: #D4442E;">
        <tbody>
            <tr>
                <td align="center" valign="middle"
class="rsvp-decline-content" style="font-family: Arial; font-size: 16px; padding: 15px;">
                    <a class="rsvp-decline-link" title="Decline" href="http://" target="_blank" style="font-weight: bold; letter-spacing: normal; line-height: 100%; text-align: center; text-decoration: none; color: #FFFFFF;">Decline</a>
                </td>
            </tr>
        </tbody>
    </table>
</td>
</tr>
</tbody>
</table>
</td>
</tr>
</tbody>
</table>
<input type="hidden" class="rsvp-group-id">
<input type="hidden" class="rsvp-occurrence-value">
</div>
</div>
</div>

```

Inlining

You can determine if your template should be inlined. In most cases you'll want to ensure this is done, but we've left an option to disable this when adding your template to Rock. Sometimes though this is not an all or nothing decision. Sometimes you may want some of your CSS styles to be inlined but others not. If you mark your styles with an 'ignore' class, the inliner will pass over the styles and not consider them for inlining.

Sample:

```

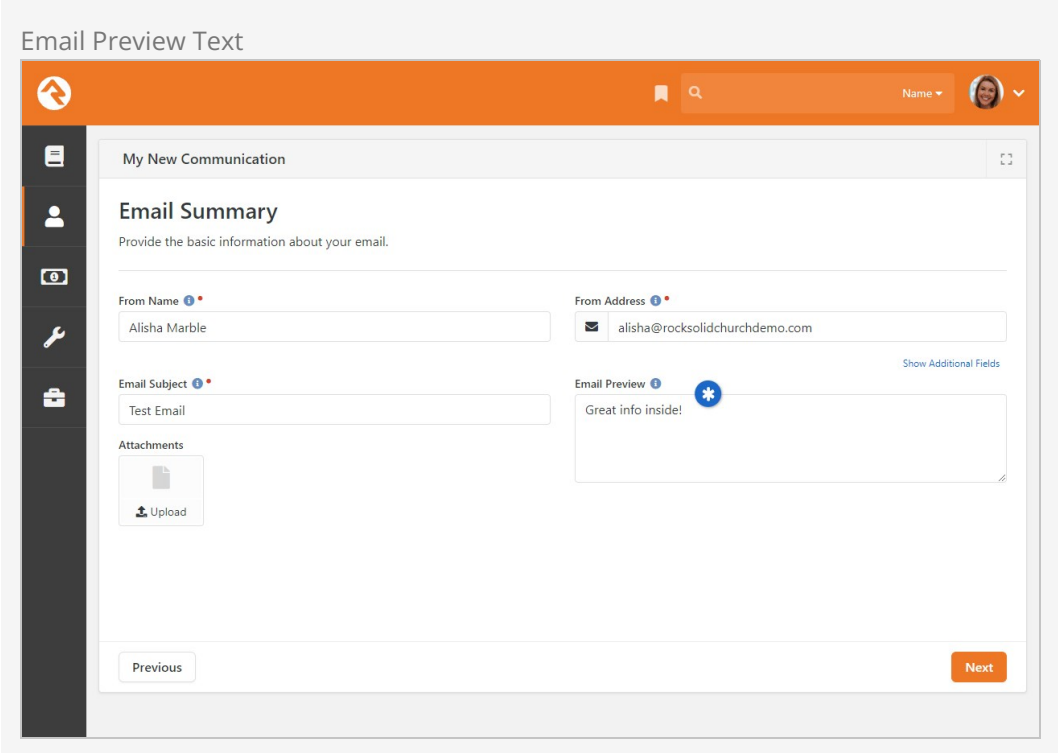
<style class="ignore">
    p {
        color: red;
    }
</style>

```

You can allow CSS Inlining for all emails in the Communication Medium configuration located at [Admin Tools > Communications > Communication Mediums](#).

Preview Text

When you get an email, you often see "preview text" that shows either the first line of text in the email or a custom message describing what the email is about. You can provide preview text from within the Communication Wizard, as pictured below.



The screenshot shows a web interface titled "Email Preview Text" for configuring a new communication. The main content area is titled "Email Summary" and contains the following fields:

- From Name:** Alisha Marble
- From Address:** alisha@rocksolidchurchdemo.com
- Email Subject:** Test Email
- Email Preview:** Great info inside!

There is also an "Attachments" section with an "Upload" button. At the bottom, there are "Previous" and "Next" navigation buttons.

You can enable preview text for your emails by adding the HTML below to your template. Just make sure it's placed after the opening `<body>` tag.

```
<span id="preheader-text" class="preheader" style="display: none !important; font-size: 1px ; line-height: 1px; max-height: 0px; max-width: 0px; mso-hide: all !important; opacity: 0; overflow: hidden; visibility: hidden;"></span>
```

Configuring Theme Customization

Rock allows themes to be easily customized by non-technical people. It's up to you, as a theme designer, to design your themes in a way that enables this. Don't worry though. It's super simple and borderline fun.

Header Logos

The one thing everyone wants to be able to change is the logo at the top. To enable this logo to be updated you'll need to add the ID of 'template-logo' to the image. You can also provide instructions (think the recommended size) by using the 'data-instructions' attribute.

```
<img id="template-logo" src="/Content/EmailTemplates/placeholder-logo.png" width="200" heig
```



```
ht="50"
data-instructions="Provide a PNG with a transparent background or JPG with the background color of #ee7725.">
```

Custom Variables

You can take your customization options to the next level by configuring variables in your theme. You configure these variables using the template editor. As you add the variables, with default values, you'll notice that some Lava is added to your template. With `Lava` in your hands you now can go crazy with styling.

```
<style>
  a {
    color: {{ linkColor }};
  }
</style>
```

Lava In The Template

You saw above how Lava is used to drive the theming of the template. So, what about times when you want Lava to be passed (unchanged) to the communication? Say for instance you would like Lava to display the person's name to be passed through to the wizard. In these cases, simply wrap that Lava in `raw` tags.

```
{% raw %} {{ Person.NickName }} {% endraw %}
```

Final Tips and Tricks

Below are some other tips to keep your templates working well with Rock:

- Make sure you're using an HTML5 Doctype for your template. This will ensure that the inliner treats your markup with the respect it deserves. If you don't, it may attempt to 'fix' your markup by providing closing tags and such. For example:
<!DOCTYPE html>
- You can apply security to each template individually under `Admin Tools > Communications > Communication Templates`. This will restrict who can see the communication on the History tab of the Person Profile page and on the Communication Detail block.