



# Let's Learn **Themes**

---

What are they, how to build them, and how to test/proof them

# What you'll learn

---

- **Overview**
  - Power of Themes
  - Programs & Tools
- **The Process**
- **The Themekit**
  - What's included
  - Structure & what to update
  - Adding elements
- **Mapping to Quick Starts**
  - "Required" CSS classes
  - How classes link to colors
- **Dive into LESS**
  - Structure walkthrough
  - Structure of a micro-theme
  - Coding
- **Testing & Proofing**
  - Themekit testing
  - Browser testing
  - Quick Start testing



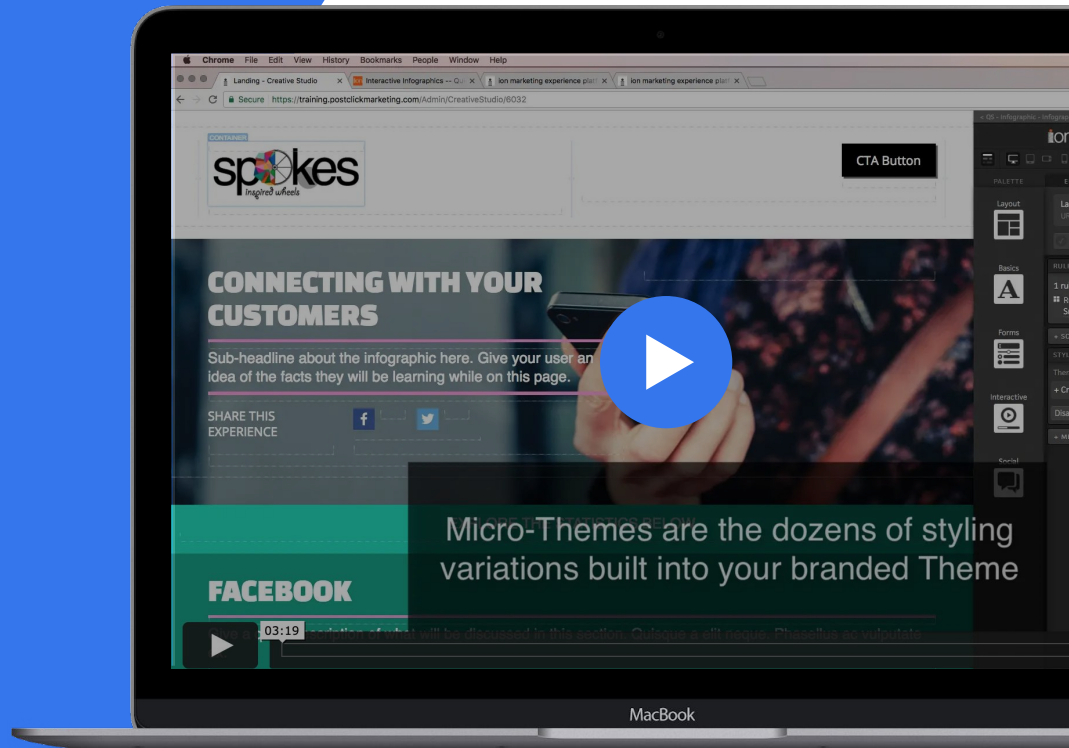
# Overview

---



# The Power of Themes

If you are unfamiliar with what a theme is, start by taking a look at our Power of Themes video. We would also recommend reviewing our [\*\*Developer's Guide PDF\*\*](#) to familiarize yourself with our terminology and micro-themes.



# Programs, Tools & Files

## Programs & Tools

- Programs & Tools
- An internet browser
- An Ion console
- A code editor like Visual Studio Code
- A LESS compiler like Codekit

*Note: You'll want to make sure that you have "Enable Javascript in LESS files" checked in whatever compiler you are using. Otherwise, you may run into an error and your files may not compile.*

## Files

- Your logo files (SVG or PNG)
- Your fonts (web font files, or an embed link)
- Your brand guidelines

# The Process

---

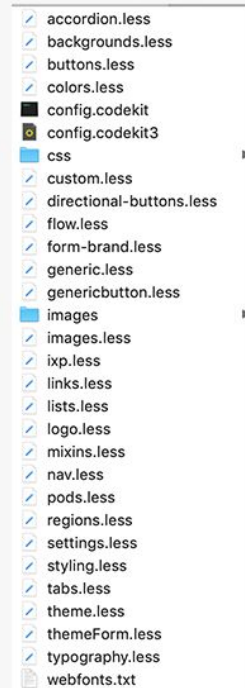


## STEP #1

# Copy the Master Theme files into a new folder for your brand.



This is a good practice to help keep the themes organized, especially if you'll be building multiple themes.



## STEP #2

# Collect the logo(s) and font(s) if applicable.

### Logo files:

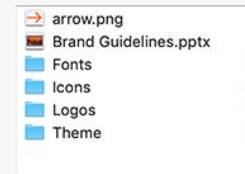
- Logo files should be saved as .svg (preferred) or .png.

### Font files:

- Font files should be uploaded in as many formats as possible for cross browser support.
- The font files that we support are:
  - .eot (IE 6-9)
  - .ttf (Safari, Android, iOS)
  - .woff (pretty modern browsers)
  - .woff2 (really modern browsers)
  - .svg (iOS legacy)



It's a good practice to put the logo(s) and font file(s) into the same overall brand folder.

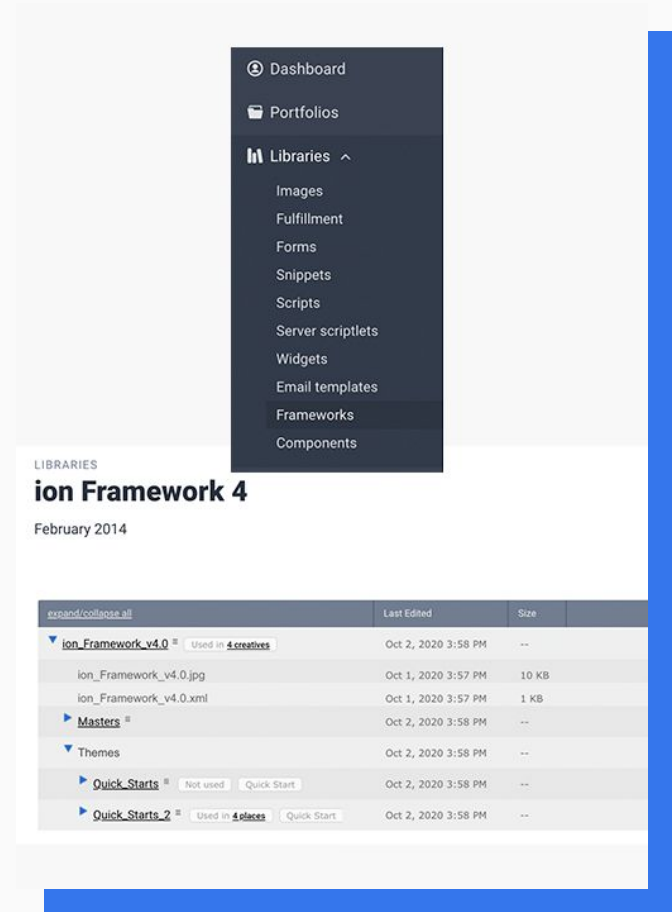




## STEP #3

**Log into the Ion console,**  
and navigate to the Frameworks  
section.

- You'll want to select the ion Framework 4.0
- Open the dropdown "Themes".



The screenshot shows the Ion console interface. A dark navigation menu is open, listing options: Dashboard, Portfolios, Libraries (with a dropdown arrow), Images, Fulfillment, Forms, Snippets, Scripts, Server scriptlets, Widgets, Email templates, Frameworks, and Components. Below the menu, the 'LIBRARIES' section is visible, featuring a header for 'ion Framework 4' dated 'February 2014'. A table below lists various framework components with columns for 'expand/collapse all', 'Last Edited', and 'Size'.

expand/collapse all	Last Edited	Size
▼ ion_Framework_v4.0 <small>Used in 4 creatives</small>	Oct 2, 2020 3:58 PM	--
ion_Framework_v4.0.jpg	Oct 1, 2020 3:57 PM	10 KB
ion_Framework_v4.0.xml	Oct 1, 2020 3:57 PM	1 KB
▶ Masters <small>▯</small>	Oct 2, 2020 3:58 PM	--
▼ Themes	Oct 2, 2020 3:58 PM	--
▶ Quick_Starts <small>▯ Not used Quick Start</small>	Oct 2, 2020 3:58 PM	--
▶ Quick_Starts_2 <small>▯ Used in 4 places Quick Start</small>	Oct 2, 2020 3:58 PM	--



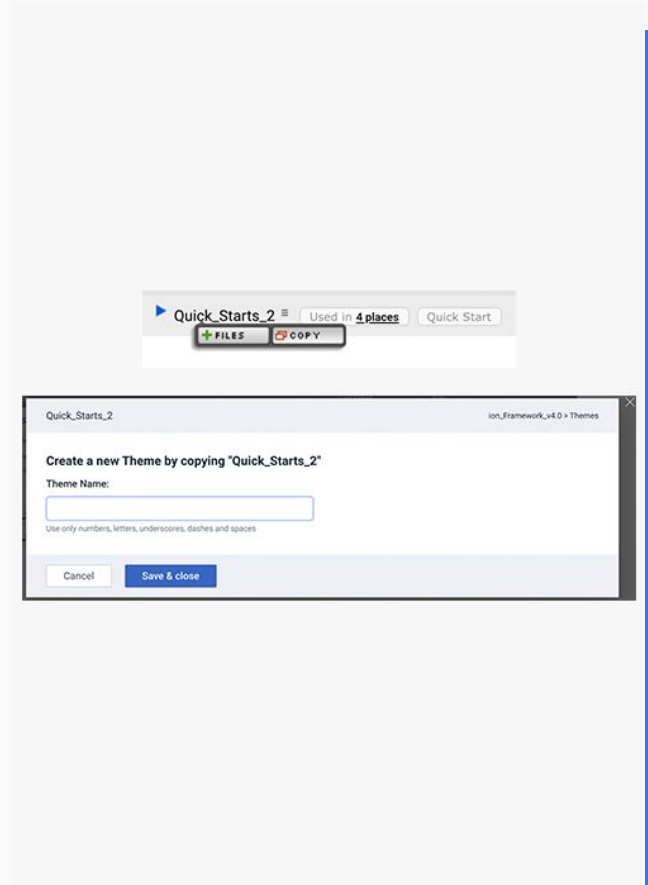
## STEP #4

# Copy the Quick Starts 2 theme and name the new theme.

You'll want to name the new theme after the brand. Themes can not have the same name. We recommended naming your theme after the brand followed by the current year. This will help avoid any confusion if a brand updates their guidelines in the future and requires a new theme.



**Example:** Brand\_Name\_2020



## STEP #5

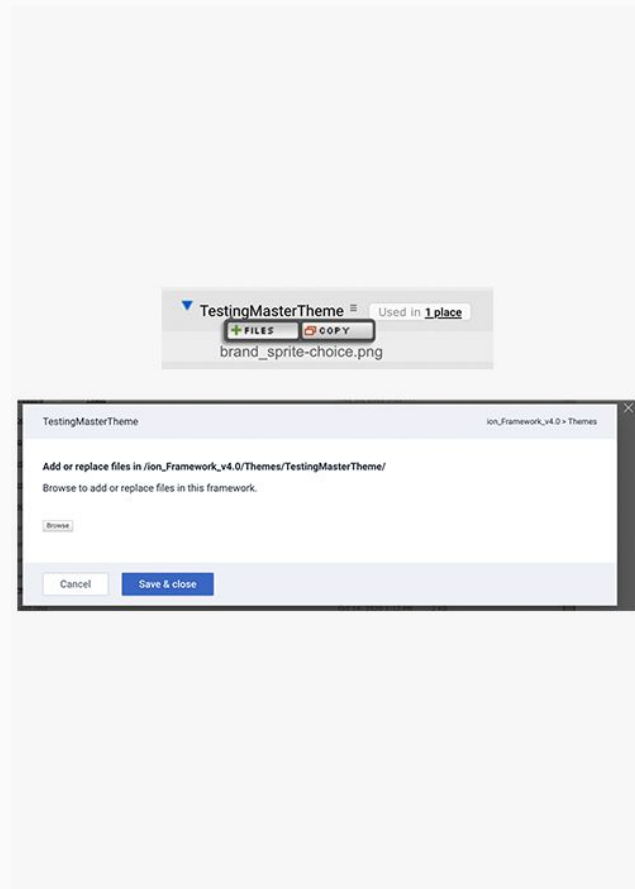
# Upload the logo(s) and font(s), and/or update the webfonts.txt file.

The font files that we support are:

- .eot (IE 6-9)
- .ttf (Safari, Android, iOS)
- .woff (pretty modern browsers)
- .woff2 (really modern browsers)
- .svg (iOS legacy)



We recommend uploading as many file types as possible for the best cross browser compatibility, but any or all of the file types can be included.



## STEP #5

- **Adding fonts through the webfonts.txt**

This method assumes that you don't have font files and are instead using a third party to host the fonts for you. Each third party should provide details for installing the font, but most will use a link reference or the @import rule.

Support post

- **Adding fonts to the framework**

If you have font files that are not being hosted by a third party, you will want to upload them directly to the framework and reference the files within your theme.

Support post



## STEP #6

# Create a new portfolio and campaign for themes, if there is not already one in your console.

When working on themes, we create a Themekit preview page to view theme as we create it and pull down Quick Start pages to test the theme.

All Themekit preview pages and test Quick Starts belong in the Ion Creative Development (portfolio) > Theming (campaign).

You will want to create this portfolio and/or campaign if it doesn't already exist in the console you're working in.

### Add campaign

Start by entering a unique name for this campaign. The "Fallback Url" is the web location you want to send people if for some reason creative are not available (e.g., an external web page that best suits this campaign). You can use the "Description" field to briefly describe the mission of this campaign.

#### Portfolio

Ion Creative Development ▾

#### Name

Theming

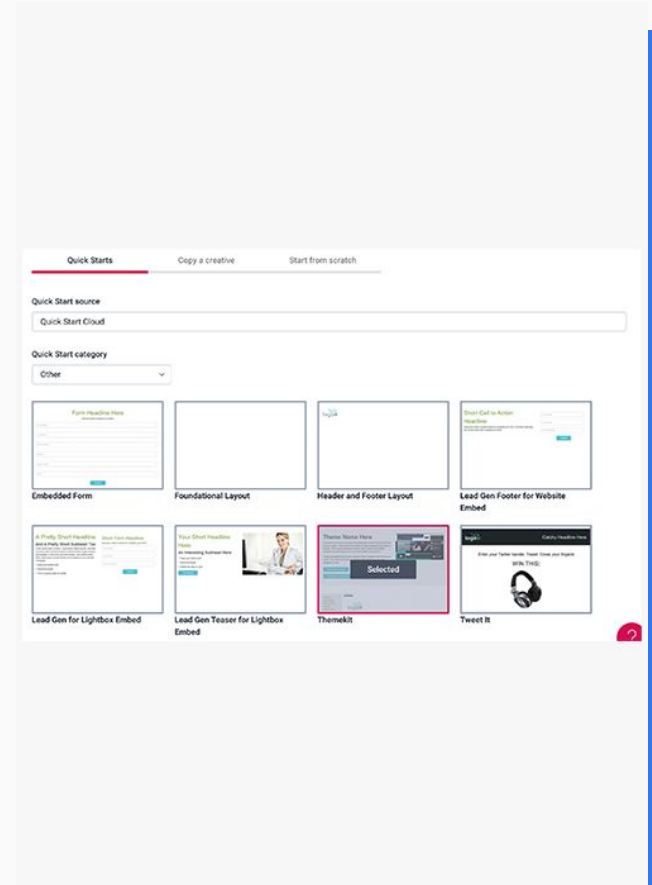
Must be unique



## STEP #7

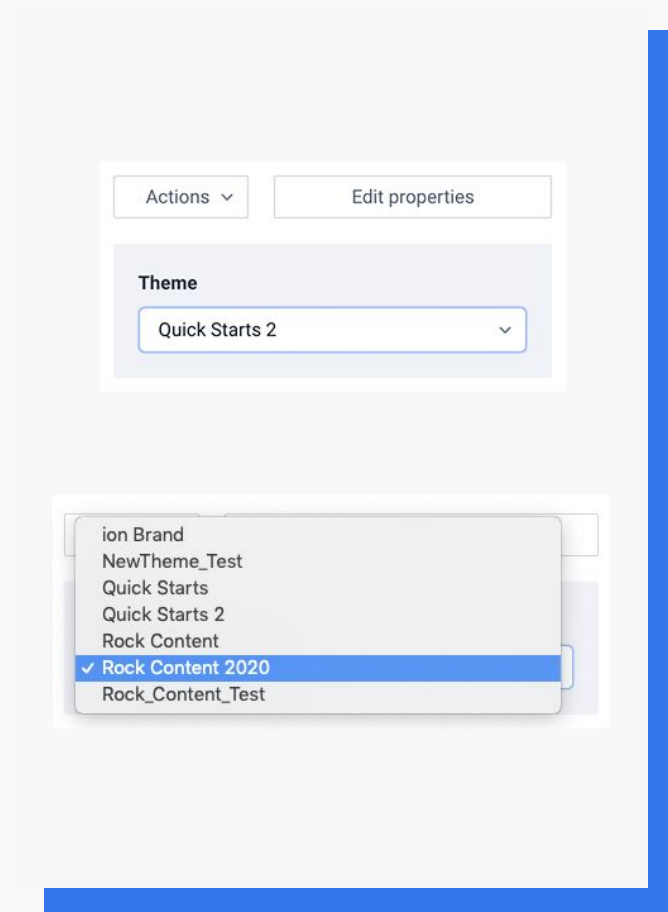
**Pull the Themekit page down**  
from the Quick Start Cloud into  
the Theming campaign.

The Themekit page can be found in Quick Start Cloud > Other > Themekit.



## STEP #8

**Navigate to the Themekit page that you just created,**  
and change the theme to  
the new brand theme.



## STEP #9

# Now you're ready to code the theme.

This is covered in depth within our “Dive Into Less” section.

Jump to **Dive Into Less**

```
() settings.less ▶ ...
1 // =====
2 // SETTINGS INFO:
3 //
4 // This is where all the basic theme settings get changed. Use generic names globally
5 // and reference those to specific names within this file. (E.g., @color-a is global,
6 // @color-blue is a specific color setting. So @color-a = @color-blue in this file.)
7 //
8 // Reference for CSS3: http://css3generator.com/
9 // =====
10 //
11 // Page Defaults
12 // =====
13 //
14
15 @BrandName:          'Brand';
16
```





## STEP #10

Compile the theme.css, form-brand.css, and themeForm.css files.

Master Theme ▾

[Clear Log](#)

● 202	/form-brand.less	Master Theme	2:04:10...
● 203	/form-brand.less	Master Theme	2:05:28...
● 204	/form-brand.less	Master Theme	2:05:5...
● 205	/form-brand.less	Master Theme	2:07:06...
● 206	/form-brand.less	Master Theme	2:07:13...
● 207	/themeForm.less	Master Theme	2:08:0...
● 208	/form-brand.less	Master Theme	2:08:0...
● 209	/theme.less	Master Theme	2:08:0...
● 210	/form-brand.less	Master Theme	2:08:14...
● 211	/form-brand.less	Master Theme	2:09:51...
● 212	/themeForm.less	Master Theme	2:10:46...
● 213	/form-brand.less	Master Theme	2:10:47...
● 214	/theme.less	Master Theme	2:10:47...
● 215	/form-brand.less	Master Theme	2:16:52...
● 216	/themeForm.less	Master Theme	2:16:52...
● 217	/theme.less	Master Theme	2:16:52...
● 218	/form-brand.less	Master Theme	2:17:02...
● 219	/themeForm.less	Master Theme	2:18:24...
● 220	/form-brand.less	Master Theme	2:18:24...
● 221	/theme.less	Master Theme	2:18:24...
● 222	/themeForm.less	Master Theme	2:20:31...
● 223	/form-brand.less	Master Theme	2:20:31...

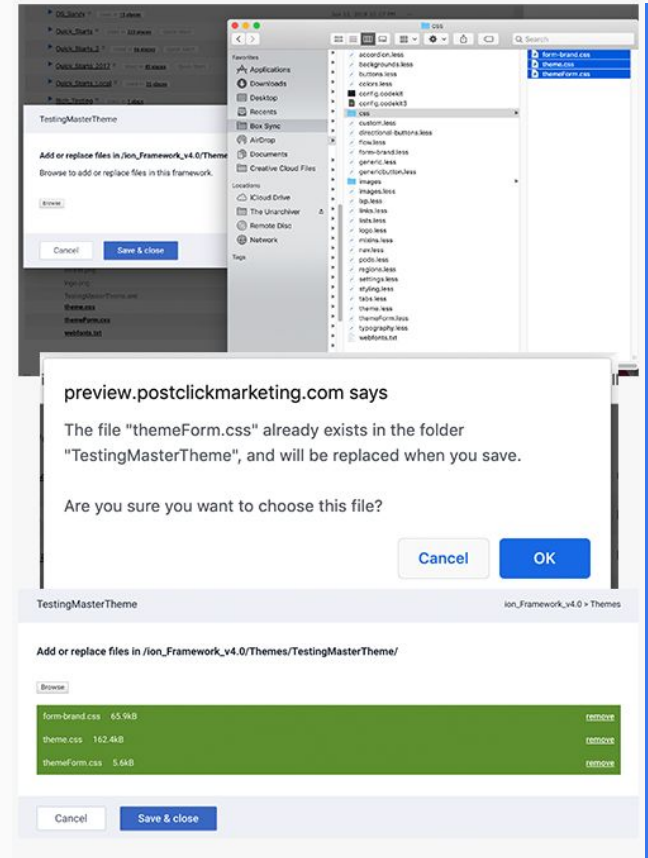


## STEP #11

# Upload the theme.css, themeForm.css, and form-brand.css files to the framework.

You can do this in multiple ways.

1. Locate the folder that holds your compiled CSS on your computer. Open the Frameworks > Themes page in your browser, and hover over your new theme. Click the "+Files" button, and upload the three compiled CSS files into the framework. You will see a warning for the files being overwritten, and you'll want to hit Yes to the warnings.
2. Copy the code within the CSS file(s) from your code editor. Open the Frameworks > Themes page in your browser, and select the applicable .css file. Paste the code into the file, and click "Save & Close".

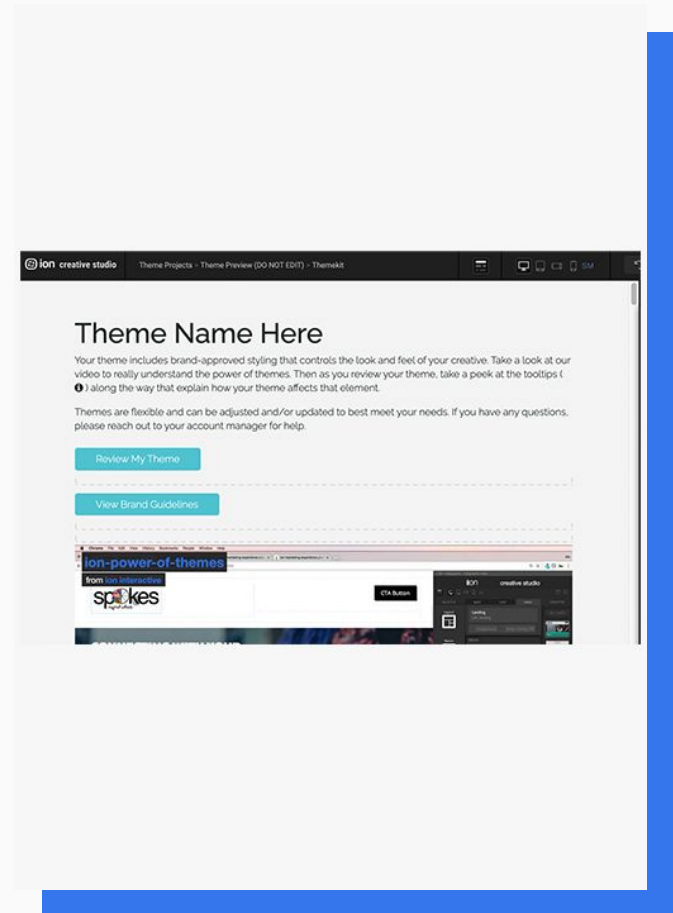


## STEP #12

**Navigate to the Themekit page** that you created and make any/all updates needed to test and show your theme.

This is covered in our “The Themekit” section.

Jump to **The Themekit**



## STEP #13

### Add a URL to the Themekit page.

Once the Themekit has a URL, you'll want to check the page in various browsers to make sure that your elements are working properly.

The screenshot shows a web interface for managing URLs. At the top, there is a dropdown menu set to "Engaged" and three buttons: "Proof", "Preview", and "Add URL". Below this is the "URL" section, which includes a "Domain" dropdown set to "preview.postclickmarketing.com" and a "Path" input field containing "theme/master-theme". A note below the domain states "must be unique within preview.postclickmarketing.com". The "URL tracking" section has two dropdown menus: "Offline" and "Testing". Below these are labels for "Category" and "Sub-category Or, add new sub-category", with a note: "Used to categorize your visitor traffic for reporting and analysis." The "Choose a creative" section has a heading and a note: "Optionally choose a creative for this URL. Note that you can do this later as well. [view as list](#)". Below this is a thumbnail of a Themekit page with a red box around it, and the label "Themekit". At the bottom, there are "Cancel" and "Save" buttons.



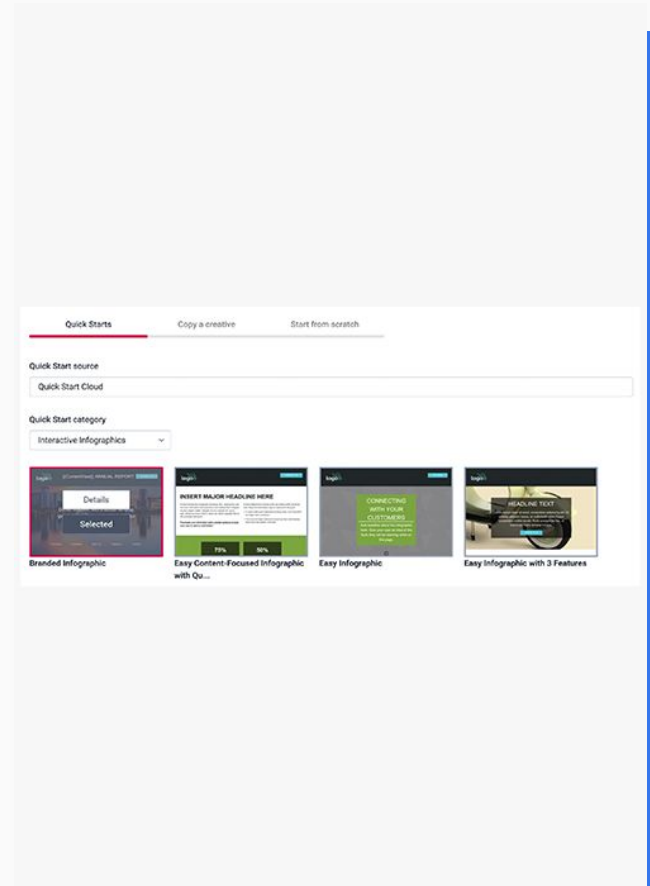
## STEP #14

# Pull down three Quick Starts and apply the new theme to them for testing.

You may want to test Quick Starts while you're coding your theme, so this step in the process may be flexible, depending on your preference.

This is covered in our "Testing & Proofing" section.

Jump to **Testing & Proofing**



## STEP #15

### Quality Assurance.

Internally, we have a QA manager who reviews all of our themes; even if you don't have a QA manager specifically, it's a good idea to have someone else double-check your new theme.



STEP #16

**Celebrate your success,**  
you've finished a new theme!



# The Themekit

---

- What's included
- Structure & what to update
- Adding elements

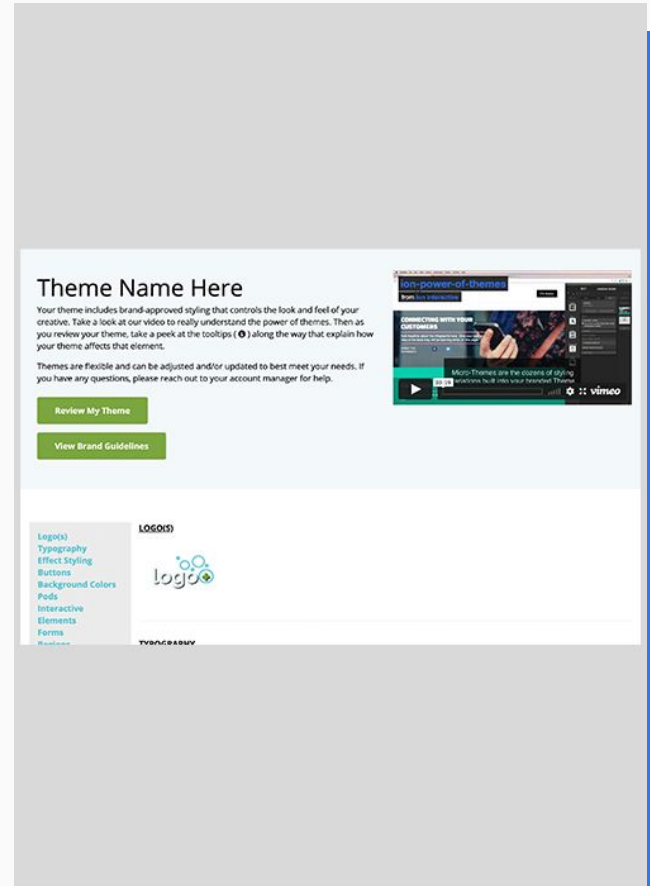




# What's Included?

[The Themekit page](#) (found in the Quick Start Cloud > Other category) showcases everything that is included in our standard code by default.

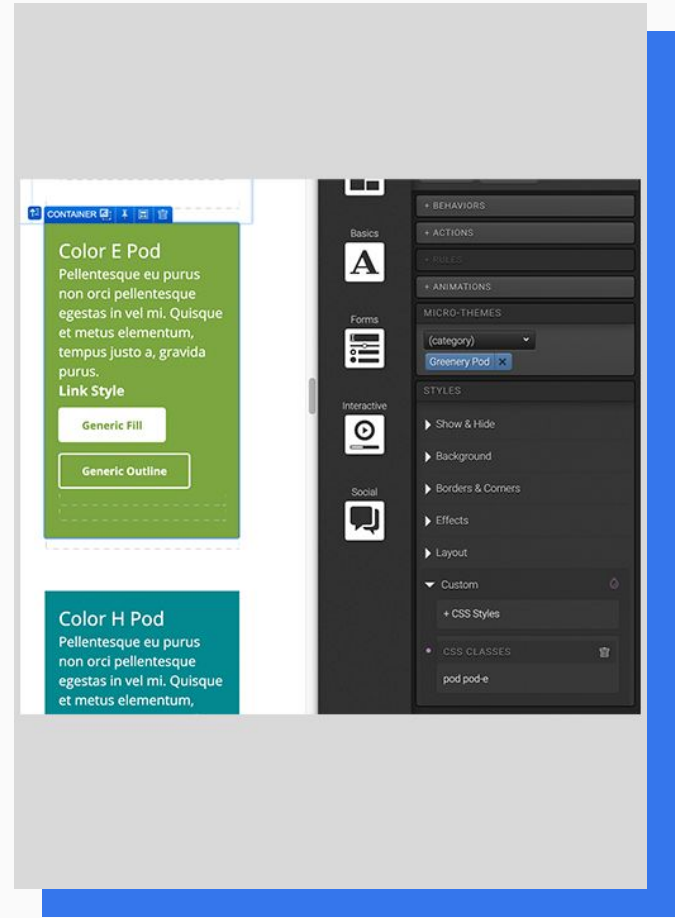
- One (1) logo
- Two (2) font styles (one font family)
- Text styles for headlines/body copy
- Nine (9) color swatches, not including White and Black
- Five (5) link styles
- Two (2) unordered list styles
- One (1) rounded corner, two (2) drop shadow, two (2) border styling
- Six (6) button colors
- Four (4) button sizes
- Four (4) button icons
- Two (2) directional button styles
- Two (2) accordion styles
- Two (2) tab styles
- One (1) numbered navigation, two (2) dot navigation, and two (2) progress bar navigation styles for Flow
- One (1) form styling
- One (1) pre-header, two (2) header, one (1) pre-content, three (3) content, one (1) post-content, one (1) footer, and one (1) post-footer region
- Two (2) navigation styles



# The structure

When you select an element, like a pod, you'll see that the "Custom" dropdown menu in the styles panel has a purple droplet on it, which means that something has been applied. The element also has a blue micro-theme tag.

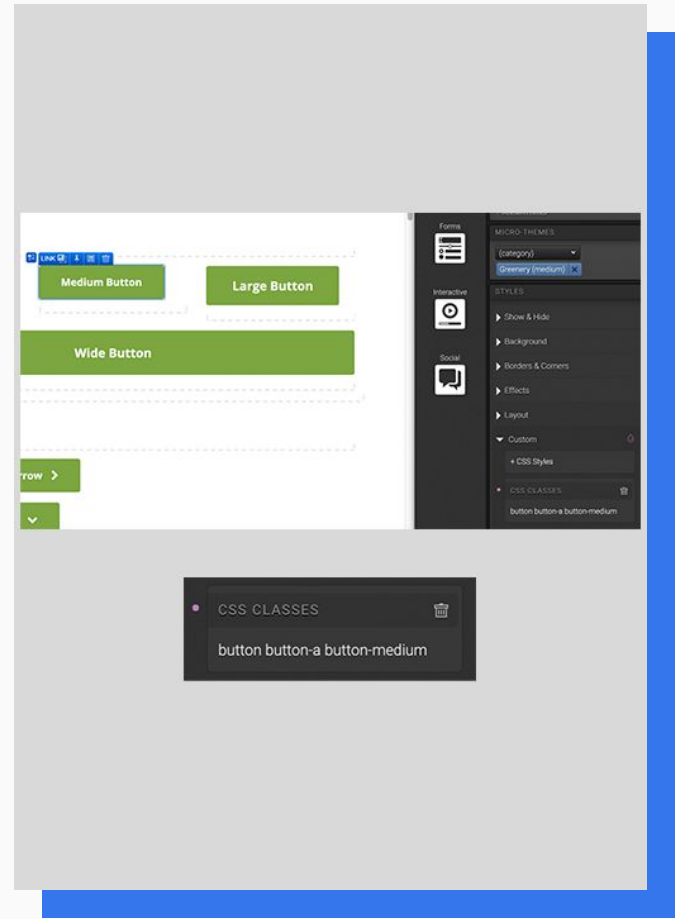
Micro-themes are linked to CSS classes within your theme. Any time a micro-theme is applied, you will see this purple droplet appear beside the Custom dropdown.



# The structure

Every micro-theme in your theme will be attached to at least one CSS class; these CSS classes are the same across all of our templates and any other themes and experiences that you may have.

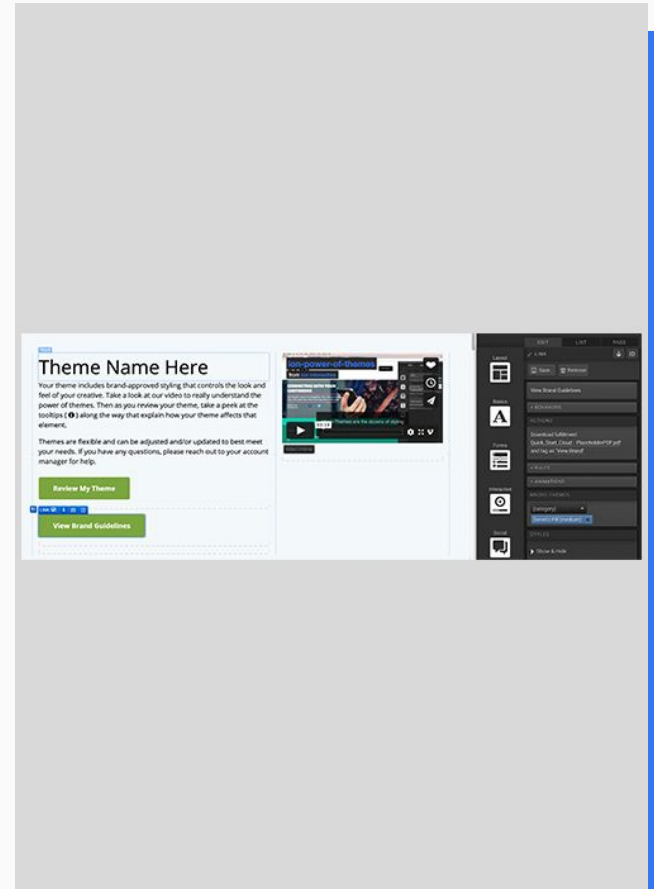
It's very important to note this, as this is what will make a difference in the amount of editing that will need to be done to Quick Starts when your new theme is applied.



# What to update

The first item to update is the name of the theme at the top of the page, and the Brand Guidelines link.

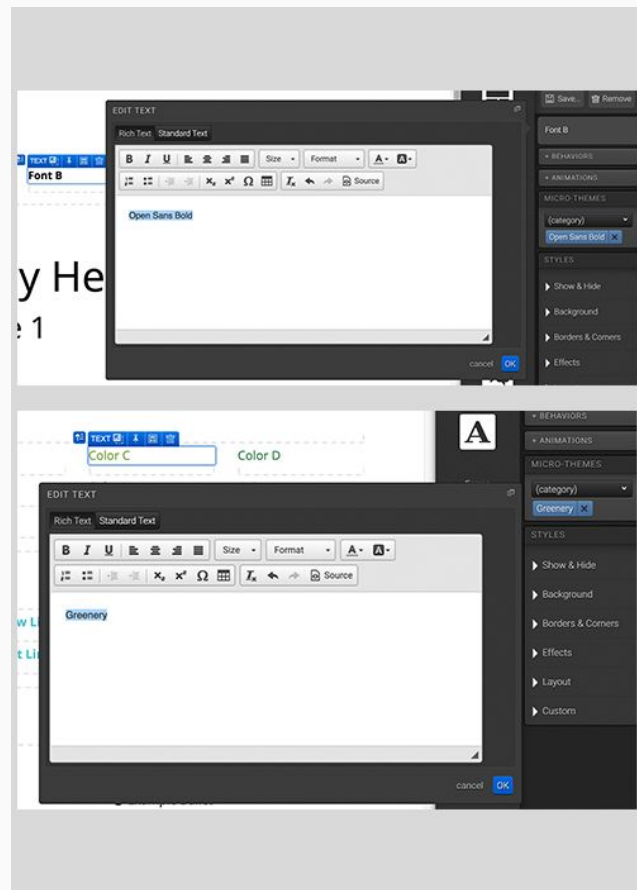
You'll want to make sure to link to the client's brand guidelines or to their website (if they don't have brand guidelines).



# What to update

As you work on the Themekit, you'll want to update all labels on all elements to match your micro-theme labels.

For example, in the Font Styles section, you'll see "Font A" and "Font B" in the text box. You'll want to change these labels in the Rich Text Editor to match the micro-theme label, so that when you or your client looks at the Themekit, they will know what fonts they have available.



# Adding elements

Let's say your brand has three logos — a color version, a black version, and a white version. The Themekit page only has one logo on it, so after adding the logos into your theme code, you'll want to also add those additional logos to the Themekit page and preview them.

You'll want to select our first logo container, and you'll copy that into the next column. Once you have the container copied, you'll update the micro-theme to the second logo that you added. It's a good practice to utilize the micro-theme dropdowns when adding these to the Themekit page, rather than typing the class directly into the Custom CSS Classes panel, to make sure that the micro-theme is hooked up correctly.



## Adding elements

You'll be able to copy other elements in the same way – if your brand has more background colors than the nine that come in the theme by default, for example, you'll select one of the background color columns, and copy and paste it into the row. Then, you'll change the Background micro-theme to your new color, and update the label inside to match your micro-theme name.



# Mapping to Quick Starts

---





## “Required” classes

All of the micro-themes in a theme are linked to at least one CSS class. This is what allows us to change themes on experiences and only need to make minor design adjustments.

Our Quick Starts utilize these required classes. This allows the Quick Start to be pulled down and have the elements rebranded almost seamlessly when switching from theme to theme.

The [required](#) classes in a theme are all delineated in comments in the LESS files. These classes are required because the Quick Starts utilize them; if your theme does not contain these classes, the Quick Starts will be badly styled and will require quite a bit of editing before being usable.



These required classes should rarely ever be adjusted and should never be removed from a theme.

To get a better grasp of how to properly map your theme(s) to the Quick Start theme, let's take a look at a Quick Start and the associated classes.

```
{ } logo.less ▶ ...
1  /* =====
2     Logo
3
4     REQUIRED CLASS: .header-logo-light
5  ===== */
```

```
{ } nav.less ▶ ...
129
130 /* =====
131     Navigation: alt theme(s)
132     REQUIRED CLASS - .ixp-nav-menu-a
133     ===== */
134
```

# Quick Starts & Colors

Open **Branded Infographic**

Looking at the Branded Infographic Quick Start, you'll see that each section has a colored background.

When we click on the Pre-Content container, it has the "Greenery Background" micro-theme applied, which we can see is the `.background-e` class.

The image shows a web design tool interface. The main workspace displays a 'STATISTICS' section with a green background. The section contains a header 'STATISTICS' and a sub-header '5K FLIGHTS'. Below this, there are three columns of data: '8K EMPLOYEES' and '170K CUSTOMERS'. The sidebar on the right shows the 'MICRO-THEMES' panel with 'Greenery Background' selected. The 'STYLES' panel shows the 'Background' property set to 'background-e'.

Category	Value
Flights	5K
Employees	8K
Customers	170K



# Quick Starts & Colors

## Open Branded Infographic

In our LESS files, you'll see that the `.background-e` class, found in the `backgrounds.less` file, utilizes the `@color-a` LESS variable. Going into the `settings.less` file, you'll see all the color swatches set up at the top of the file.

`@color-a` will be the primary color, and will map to that "Greenery" color in our Quick Starts. This means that any color that you set to `@color-a` will have that color applied anywhere you see "Greenery" in the Quick Starts.

Our `settings.less` file has comments for what each color variable will map to in the Quick Starts, and it is a best practice to try and keep the contrast values of these colors similar, so that the Quick Starts pull down with as little editing needed as possible.



**TIP:** `@color-a` should always be the primary color of the brand, because "Greenery" gets used consistently in our Quick Starts, and should be a color dark enough that white text and imagery would look good on.

```
() backgrounds.less ▶ .background-e
30
31 .background-e
32   background: @color-a;
33

() settings.less ▶ ...
26 // Primary
27 @color-a: #86af49; //Main Bright Color (matches to Greenery #86af49)
28 @color-a-name: 'Greenery';
```



## “Required” classes

What to do if the brand doesn't have as many options as the default theme?

For example, our theme requires nine colors. If your brand only has six colors, you'll want to duplicate those colors into the color variables, rather than removing the three additional ones.

This will be the same for any other required elements; if your brand only has one navigation style, you will want to duplicate that style into the alternate style, rather than removing it entirely.

```
{ } logo.less ▶ ...  
1  /* =====  
2     Logo  
3  
4     REQUIRED CLASS: .header-logo-light  
5  ===== */
```

```
{ } nav.less ▶ ...  
129  
130 /* =====  
131     Navigation: alt theme(s)  
132     REQUIRED CLASS - .ixp-nav-menu-a  
133     ===== */  
134
```

# Dive into LESS

---

- Structure walkthrough
- Structure of a micro-theme
- Coding



# Structure Walkthrough

Let's take a look at the Master Themekit folder.

Within the Master Theme folder, you'll see 25 LESS files, a webfonts.txt file, a CSS folder with 3 compiled CSS files inside, and an images folder with some sprite images inside.

You'll also see a codekit configuration file. We use Codekit to compile our LESS files, so if you use a different compiler you may end up having a different config file.

Open **Master Themekit**



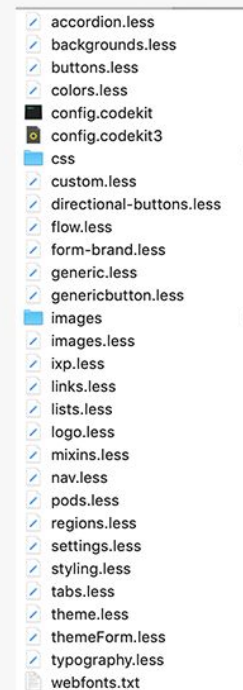
# Structure Walkthrough

The LESS files are broken down into segments that follow the overall structure of our Themekit page and our settings.less file.

The most important files to note are:

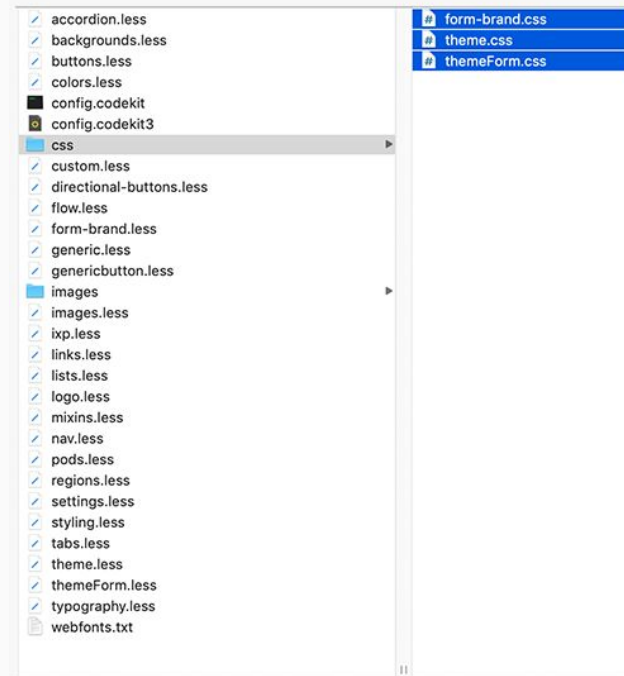
- Theme.less
- themeForm.less
- Form-brand.less
- Settings.less

The theme, themeForm, and form-brand files will be the three files that will compile into CSS files. Looking inside the CSS folder, you'll see the three .css files coincide with those three LESS files.



## Compiled Files

- **theme.less**  
This file is importing the majority of our other LESS files, which will then be compiled into CSS.
- **theme.css**  
The theme.css file will be uploaded to the platform framework.
- **themeForm.less** and **form-brand.less**  
These files are used by our platform to style our forms.
- **themeForm.css** and **form-brand.css**  
The themeForm.css and form-brand.css files will be uploaded to the platform framework.





# Structure Walkthrough

- **Accordion.less:** This file controls the styling of the accordions
- **Backgrounds.less:** This file controls the styling of all the background colors
- **Buttons.less:** This file controls the styling of all the buttons
- **Colors.less:** This file controls the styling of all the text colors
- **Custom.less:** This is an empty file that can contain any miscellaneous styling that might not fit anywhere else
- **Directional-buttons.less:** This file controls the styling of the directional button arrows
- **Flow.less:** This file controls the styling of the flow navigation dots, numbers, and progress bar
- **Form-brand.less:** This file controls the styling of the Freestyle Forms
- **Generic.less:** This file controls a bit of styling on the overall page, the HTML and body styling
- **Genericbutton.less:** This file controls the styling of the “Generic” button classes
- **Images.less:** This file controls the styling for image alignment
- **Ixp.less:** This file is specific to Ixp and the micro-themes. For reference, please see the Developers Guide PDF. This is where all of the references for the micro-theme names and categories are set.
- **Links.less:** This file controls the styling for all the links
- **Lists.less:** This file controls the styling for the ordered and unordered lists
- **Logo.less:** This file controls the styling for the logo(s)
- **Mixins.less:** This file holds the common LESS mixins that our themes utilize
- **Nav.less:** This file controls the styling for the navigation
- **Pods.less:** This file controls the styling of all the pods
- **Regions.less:** This file controls the styling of all the regions
- **Settings.less:** This is the overall settings file, and this is where 95% of your editing will be done. We'll jump into this file in the next video.
- **Styling.less:** This file controls the miscellaneous “Styling” category, which contains things like borders and rounded corners by default
- **Tabs.less:** This file controls the styling of the tabs
- **Theme.less:** This file is the compilation of all the other LESS files
- **themeForm.less:** This file controls an old-school styling of forms
- **Typography.less:** This file controls the styling of the typography overall, including the fonts and font styles, the headlines and paragraphs, any additional styles like captions, etc.
- **Webfonts.txt:** This file will hold your embed link if you're using a third party service to host your fonts, like Google.



# Structure Walkthrough

Micro-theme properties include:

- -ixp-name: "Name";
- -ixp-tags: "tagname";
- -ixp-group: "groupname";
- -ixp-scope: "scope";

To create a new micro-theme in CSS, you would add these lines to your overall CSS class.

Example:

```
.example-style {  
  -ixp-name: "Name";  
  -ixp-tags: "tagname";  
  -ixp-group: "group";  
  -ixp-scope: "scope";  
}
```

- **-ixp-name:** "Name";: The name of the Micro-theme (i.e. ixp-name: "Green Pod";). The value from this property will appear in the 2nd drop-down of the Micro-theme section in creative studio.
- **-ixp-tags:** "tagname";: The value from this property will appear in the Micro-theme category drop-down in creative studio when an element is selected and you're viewing the Edit tab.
- **-ixp-group:** "group";: The value from this property will determine whether or not selecting multiple Micro-themes from the same Micro-theme category will have one Micro-theme override another or apply all Micro-themes selected.
- **-ixp-scope:** "scope";: A comma-separated list of scopes. Micro-themes will only appear in the dropdown when specific elements within the scope are selected. By narrowing a Micro-theme's scope, you control which elements can be styled.

Open [Developer's Guide PDF](#)



## Structure of a micro-theme: LESS

The `ixp.less` file contains LESS mixins for all of the standard scopes / categories of micro-themes that are typically included in a theme.

To create a micro-theme in the LESS files, simply call the appropriate mixin (example: `.ixp-colors()`) and pass in the name parameter (this name will be what is shown in the Micro-theme tag).



**Example:** `ixp-colors("@{color-a-name}");`

```
{ } colors.less ▶ ...
96
97  ▫ /* =====
98     Colors IXP Information
99     ===== */
100
101  .color-a { .ixp-colors("@{color-white-name}"); }
102  .color-b { .ixp-colors("@{color-black-name}"); }
103
104  // Primary
105  .color-c { .ixp-colors("@{color-a-name}"); }
106  .color-d { .ixp-colors("@{color-b-name}"); }
107  .color-e { .ixp-colors("@{color-c-name}"); }
108  .color-f { .ixp-colors("@{color-d-name}"); }
109
110  // Secondary
111  .color-j { .ixp-colors("@{color-h-name}"); }
112  .color-k { .ixp-colors("@{color-i-name}"); }
113
114  // Neutral
115  .color-g { .ixp-colors("@{color-e-name}"); }
116  .color-h { .ixp-colors("@{color-f-name}"); }
117  .color-i { .ixp-colors("@{color-g-name}"); }
```

# Adding new micro-theme categories

It's a best practice to utilize the standard / default categories as much as possible, but sometimes new categories will need to be added.



**Example:** Breaking down background colors into:

- Primary
- Secondary

To do this, you'll add a new micro-theme "Name" and "Tag".



**Example:**

```
-ixp-name: "Backgrounds - Secondary";  
-ixp-tags: "Backgrounds - Secondary";  
-ixp-group: "background";  
-ixp-scope:  
"ContainerLike,StyleOnly,CustomForm,ChoiceGroup,Flow,Flow  
Step,FullPageSection,FullPageSubsection";
```

In this example, the group would be the same for Backgrounds - Primary and Backgrounds - Secondary, because you would want the user to only be able to apply one of these micro-themes at a time.

```
// background  
.ixp-background@name {  
  -ixp-name: @name;  
  -ixp-tags: "backgrounds";  
  -ixp-group: "background";  
  -ixp-scope: "ContainerLike,StyleOnly,CustomForm,ChoiceGroup,Flow,FlowStep,FullPageSection,FullPageSubsection";  
}
```

**Now let's code.**



## STEP #1

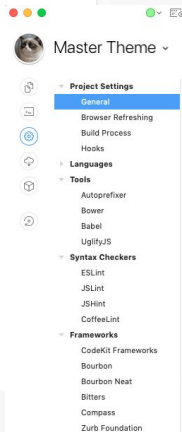
# Add the folder to your compiler and code editor & add files to your framework.

We'll start by making a copy of the Master Theme folder, and naming it with our new theme name. Once we have that copy created, we'll add the theme folder to our compiler — for example, Codekit — and we'll rename it to match the name of our theme.



You'll want to make sure that you have "Enable Javascript in LESS files" checked in whatever compiler you are using; otherwise, you may run into an error and your files may not compile.

- accordion.less
- backgrounds.less
- buttons.less
- colors.less
- config.codekit
- config.codekit3
- css
- custom.less
- directional-buttons.less
- flow.less
- form-brand.less
- generic.less
- genericbutton.less
- images
- images.less
- ixp.less
- links.less
- lists.less
- logo.less
- mixins.less
- nav.less
- pod5.less
- regions.less
- settings.less
- styling.less
- tabs.less
- theme.less
- themeForm.less
- typography.less
- webfonts.txt



### General Project Settings

**Icon and Name:**  Master Theme

**Config File:**  Hide the project's configuration file in the Finder

**Skipped Folders:** log, logs, .cache, cache, /storage/  
framework/sessions, node\_modules

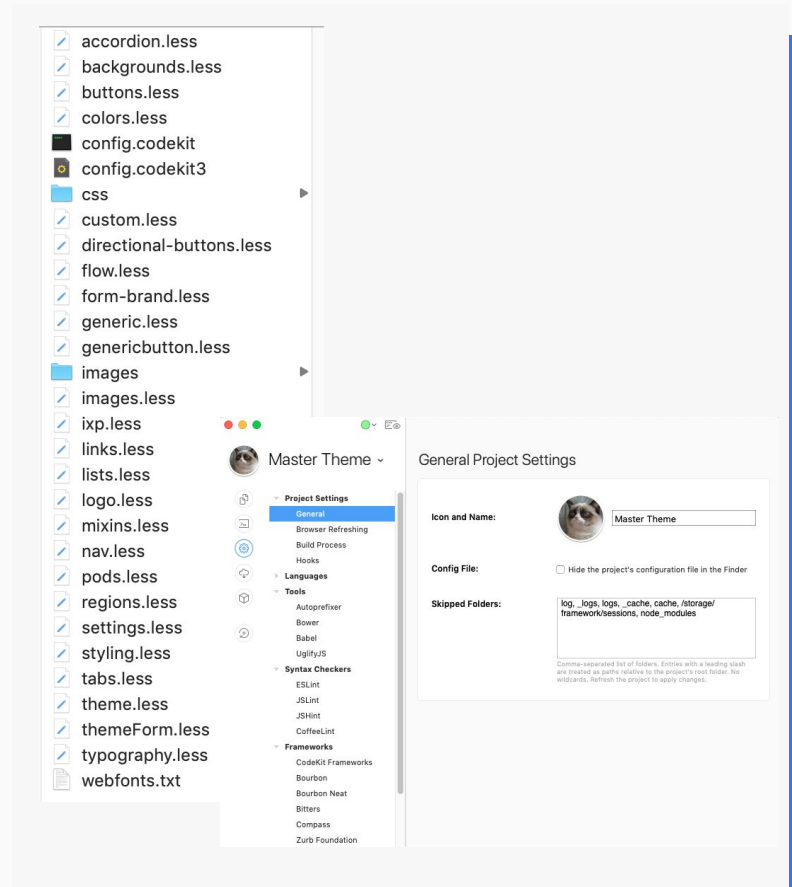
Comma-separated list of folders. Entries with a leading slash are treated as paths relative to the project's root folder. No wildcards. Refresh the project to apply changes.

## STEP #1

**Next, we'll add the theme folder to our code editor.** You'll see the folder structure on the left side panel, with all LESS files.

Finally, we'll open our framework in the Ion console, so that we can upload our logos and font files to reference as we're coding. When you hover over the theme name, you'll see a couple of actions appear.

We'll click the +Files button to then drag and drop the logo files and font files into our framework.



## STEP #2

**Update the comments at the top** of the theme.less, themeForm.less, and form-brand.less files.

You'll want to update the Theme Name.css, the Author, and the date.

```
{ } theme.less
1 // =====
2 // Name: ION Master Theme 2020
3 //
4 //
5 // NOTE: LESS comment slashes do not
6 // compile to css - internal use only
7 // =====
8
9 /* =====
10 ThemeName Theme.css
11
12 Created : Author 04/02/2020
13 Last Updated : Author 04/02/2020
14 ===== */
15

{} themeForm.less ▶ ...
1 /* =====
2 ThemeName Theme.css
3
4 Created : Author 04/02/2020
5 Last Updated : Author 04/02/2020
6 ===== */
7

{} form-brand.less ▶ ...
1 1 /* =====
2 ThemeName FormBrand.css
3
4 Last Updated : Author 04/02/2020
5 Created : Author 04/02/2020
6 ===== */
7
```





## STEP #3

# Open the settings.less file and update the Brand Name.

The first thing we'll be updating is the brand name. This will be what shows up as the name on the Form micro-theme. You'll want this to be the name of the brand that you're building.

```
() settings.less ▶ ...
1 // =====
2 // SETTINGS INFO:
3 //
4 // This is where all the basic theme settings get changed. Use generic names globally
5 // and reference those to specific names within this file. (E.g., @color-a is global,
6 // @color-blue is a specific color setting. So @color-a = @color-blue in this file.)
7 //
8 // Reference for CSS3: http://css3generator.com/
9 // =====
10 //
11 // Page Defaults
12 // =====
13 //
14
15 @BrandName:          'Brand';
16
```



## STEP #4

# Update the color swatches & understand the variables.

The second section in the settings.less file is the Swatches. This is where you will set the color variables that will be used throughout the rest of the brand.



It's very important to note, as you're looking at this, that these are setting the LESS variables for the colors, and will not necessarily coincide with the CSS class names.

As you work through the settings.less file, you'll notice that the variables come in sets, and almost all the major variables that you set will have a 'name' associated with them. This name variable will be what populates in the micro-themes in Creative Studio.

The first two color variables are white and black. The next four are typically the "primary" color or colors for the brand. The next two are the "secondary" colors, and then the last three are the "neutral", or typically gray, colors.

```
50 // Form Colors
51 @error: #B32F3D; //form error color - red or similar
52 @active: @color-c; //form active color - generally a main brand color
53 @placeholder-color: @color-f; // placeholder color - generally a lighter gray;
```



## STEP #4

# Update the color swatches & understand the variables.

As mentioned in our Mapping to Quick Starts section, it is very important to map these color swatches to similar shades that are noted in the comments beside each variable.

For example, `@color-a` is the main “Greenery” color in our Quick Start theme. As you look at the Quick Starts, any container / element that has the “Greenery” color applied will pull down with the color that you set as the `@color-a` variable. Occasionally you may want to set multiple variables to one color, and then add additional variables, to make Quick Starts pull down more seamlessly.



**Example:** Brand’s main colors are navy blue and orange, and they have a robust secondary palette that should only be used in very specific cases. Instead of setting the secondary colors to `@color-c`, `@color-d`, etc. (which would make the Quick Starts pull down with these colors as main background colors), you could set `@color-a` and `@color-d` to navy blue, and `@color-b` and `@color-c` to orange, and then add additional variables for the secondary palette.

```
50 // Form Colors
51 @error: #B32F3D; //form error color - red or similar
52 @active: @color-c; //form active color - generally a main brand color
53 @placeholder-color: @color-f; // placeholder color - generally a lighter gray;
```



## STEP #4.1

# Adding additional colors/swatches

If your theme has more than nine colors, you'll want to add your additional color swatches as variables in this section. We recommend using the same overall naming convention that we have already set up when adding new variables.



**Example:** Add `@color-j` and `@color-j-name`, and `@color-k` and `@color-k-name`.

In the next slide, we'll go over adding these new colors to the `colors.less` and `backgrounds.less` files so that we will have them as text and background color options; if you don't need to add additional colors to your theme, you can move on to Step 6, updating the logo.

```
( ) settings.less > @color-j
34 @color-o-name:      real;
35
36 // Secondary
37 @color-h-name:      #285124; //Dark Color (matches to Dark Moss #285124)
38 @color-h-name:      'Dark Moss';
39 @color-i-name:      #c2d7a4; //Light Color (matches to Grass #c2d7a4)
40 @color-i-name:      'Grass';
41
42 @color-j:           #ff0000;
43 @color-j-name:      'Red';
44 @color-k:           #2d62c4;
45 @color-k-name:      'Blue';
46
```

## STEP #5

# Update the colors.less and backgrounds.less files (if you've added color variables)

Let's start with colors.less. You'll see that we have the CSS classes set up in this file, and we're pulling the variable names from the settings.less file.



**Remember**, the variable names in the settings.less file will not always coincide with the names of the CSS classes. `.color-a` – the CSS class – is always set to white in our themes. `.color-b` is always set to black. Because of this, `.color-c` (the CSS class) will be using the `@color-a` variable.

You'll add your new colors at the bottom of the CSS section, using an updated class name and the variables that you set in your settings.less file.

```
{ } colors.less > ...
9
10 .color-a {
11     color: @color-white !important;
12
13     h1,h2,h3,h4,h5,h6,a,p {
14         color: @color-white !important;
15     }
16 }
17
18 .color-b {
19     color: @color-black !important;
20
21     h1,h2,h3,h4,h5,h6,a,p {
22         color: @color-black !important;
23     }
24 }
25
26 .color-c {
27     color: @color-a !important;
28     h1,h2,h3,h4,h5,h6,a,p {
29         color: @color-a !important;
30     }
31 }
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88 .color-k {
89     color: @color-i !important;
90
91     h1,h2,h3,h4,h5,h6,a,p {
92         color: @color-i !important;
93     }
94 }
95
96
```



## STEP #5

# Update the colors.less and backgrounds.less files (if you've added color variables)

(in colors.less)

Underneath the CSS section, you'll see a "Colors IXP Information" section.

This is where the micro-theme name and category will be set. You'll notice that this follows the same order that we've set up in the settings.less file — .color-a and .color-b, white and black, are first, then the 4 primary colors, the 2 secondary colors, and the 3 neutral colors. This is the order the colors will show up in your micro-theme dropdown list.

You will want to add your additional colors to this section as well, updating the class and variable names accordingly, so that the micro-themes will show up in Creative Studio.

Save your colors.less file, and open the backgrounds.less file.

```
{ } colors.less ▶ ...
96
97  ▫ /* =====
98     Colors IXP Information
99     ===== */
100
101  .color-a { .ixp-colors("@{color-white-name}"); }
102  .color-b { .ixp-colors("@{color-black-name}"); }
103
104  // Primary
105  .color-c { .ixp-colors("@{color-a-name}"); }
106  .color-d { .ixp-colors("@{color-b-name}"); }
107  .color-e { .ixp-colors("@{color-c-name}"); }
108  .color-f { .ixp-colors("@{color-d-name}"); }
109
110  // Secondary
111  .color-j { .ixp-colors("@{color-h-name}"); }
112  .color-k { .ixp-colors("@{color-i-name}"); }
113
114  // Neutral
115  .color-g { .ixp-colors("@{color-e-name}"); }
116  .color-h { .ixp-colors("@{color-f-name}"); }
117  .color-i { .ixp-colors("@{color-g-name}"); }
```



## STEP #5

# Update the colors.less and backgrounds.less files (if you've added color variables)

(in backgrounds.less)

You'll see the exact same overall structure in this file that we have in the colors.less file. Again, the variable names won't coincide with the CSS class names; .color-a through .color-d are reserved for Black, Black Transparent, White, and White Transparent. Your brand background color CSS classes will start with .color-e.

Add your additional background CSS class(es), and add the additional micro-themes that coincide.

Save your backgrounds.less file, and go back to your settings.less file.

```
{ } backgrounds.less ▶ ...
12
13 .background-a {
14     background: @color-white;
15 }
16
17 .background-b {
18     background: @color-white;
19     background: rgba(255, 255, 255, .50);
20 }
21
22 .background-c {
23     background: @color-black;
24     background: rgba(0, 0, 0, .40);
25 }
26
27 .background-d {
28     background: @color-black;
29 }
30
31 .background-e {
32     background: @color-a;
33 }

{ } backgrounds.less ▶ ...
67 /* =====
68     Background IXP Information
69     ===== */
70
71 .background-a { .ixp-background("@{color-white-name} Background"); }
72 .background-b { .ixp-background("@{color-white-name} Transparent Background"); }
73 .background-d { .ixp-background("@{color-black-name} Background"); }
74 .background-c { .ixp-background("@{color-black-name} Transparent Background"); }
75
76 // Primary
77 .background-e { .ixp-background("@{color-a-name} Background"); }
78 .background-f { .ixp-background("@{color-b-name} Background"); }
79 .background-g { .ixp-background("@{color-c-name} Background"); }
80 .background-h { .ixp-background("@{color-d-name} Background"); }
81
82 // Secondary
83 .background-l { .ixp-background("@{color-h-name} Background"); }
84 .background-m { .ixp-background("@{color-i-name} Background"); }
85
86 // Neutral
87 .background-i { .ixp-background("@{color-e-name} Background"); }
88 .background-j { .ixp-background("@{color-f-name} Background"); }
89 .background-k { .ixp-background("@{color-g-name} Background"); }
```



## Update the logo

- The next section in the settings.less file is the logo.
- The logo URL will be found in your framework. You don't need a folder structure prefacing it. Copy the file name from your framework, then paste it into your settings file.
- The next two variables are the logo height and width.
- The next two variables are the logo height and width in the SM and XS viewports, if the logo should be resized in smaller viewports.
- The last variable is the name that will show up in the micro-theme for the logo.

```
{ } settings.less ▶ ...  
56 // =====  
57 // Logo  
58 // =====  
59  
60 @logo-a-URL:          'logo.png';  
61 @logo-a-height:      ~'80px';  
62 @logo-a-width:       ~'136px';  
63 @logo-a-height-sm:   ~'54px';  
64 @logo-a-width-sm:    ~'96px';  
65 @logo-a-name:        'Logo';  
66
```



## STEP #6.1

# Add additional logo(s)

If your theme has more than one logo, you'll want to add your additional logo variables. We recommend using the same overall naming convention that we have already set up when adding new variables.



**Example:** Add @logo-b-URL, @logo-b-name, etc.



**TIP:** If your logos are the same size, you can set up the size variables once, rather than setting them for each individual logo.

In the next slide, we'll go over adding additional logos to the logo.less file; if you don't need to add additional logos to your theme, you can move on to Step 8, updating the typography.

```
{ } settings.less ▾ @logo-b-URL
59
60
61 // =====
62 // Logo
63 // =====
64
65 @logo-a-URL:          'logo.png';
66 @logo-a-height:      ~'80px';
67 @logo-a-width:       ~'136px';
68 @logo-a-height-sm:   ~'54px';
69 @logo-a-width-sm:    ~'96px';
70 @logo-a-name:        'Logo';
71
72 @logo-b-URL:          'logo2.png';
73 @logo-b-height:      ~'50px';
74 @logo-b-width:       ~'150px';
75 @logo-b-height-sm:   ~'50px';
76 @logo-b-width-sm:    ~'150px';
77 @logo-b-name:        'Logo - Alternate';
78
79 @logo-c-URL:          'logo3.png';
80 @logo-c-name:        'Logo - Alternate Black';
81
```

## STEP #7

# Update the logo.less file (if you've added logo variables)

In the logo.less file, you'll see the same structure as in the colors.less and backgrounds.less files, a CSS section at the top, and an IXP section below.



If your logos are the same size, you can add your additional logo class to the existing CSS block, and then just override the background image URL with your new variable.

If your logos aren't the same size, you'll want to add your CSS block for the new logo beneath the one that already exists.

```
() logo.less ▶ header-logo-c
1  /* =====
2  Logo
3  4  REQUIRED CLASS: .header-logo-light
5  ===== */
6
7  .header-logo-light, .header-logo-b, .header-logo-c {
8    background-image: url('@logo-a-url');
9    background-repeat: no-repeat;
10   background-position: 0 0;
11   min-height: @logo-a-height;
12   min-width: @logo-a-width;
13   background-size: @logo-a-width @logo-a-height;
14   display: inline-block;
15   text-indent: -9999px;
16 }
17 @media (max-width: 768px) { // media query to display a different logo image or logo specs for mobile and/or tablet
18   .header-logo-light, .header-logo-b, .header-logo-c {
19     min-height: @logo-a-height-sm;
20     min-width: @logo-a-width-sm;
21     background-size: @logo-a-width-sm @logo-a-height-sm;
22   }
23 }
24 .header-logo-b {
25   background-image: url('@logo-b-url');
26 }
27 .header-logo-c {
28   background-image: url('@logo-c-url');
29 }
30
```

## STEP #7

# Update the logo.less file (if you've added logo variables)

Once the CSS has been added, you'll scroll down to the IXP section and add your new logo(s) to the micro-themes.

Save your logo.less file, and go back to your settings.less file.

```
{ } logo.less ▶ ...
25  /* =====
26     Logo IXP Information
27
28     REQUIRED CLASS: .header-logo-light
29     ===== */
30
31  .header-logo-light { .ixp-logo("@{logo-a-name}"); }
```



## STEP #8

# Update the typography

The next section in the settings.less file is the Typography section.

You'll start by setting your font family in the `@font-family-base` variable.

The `@font-base-name` variable will be the name that is seen in the micro-theme for this font.



If your brand has more than one font family, you'll want to add additional font family and font name variables below the `@font-family-base`.

The next two variables will set the base font size, which will be what all the other variables reference. The two variables after that will set the default text color and the default background color for the entire body container. Typically the font-color-base does not change, and we do not recommend changing the background-color-base. These will be adjustable within publishing.

The next variable sets the default font weight, which most of the time should be "normal".

```
() settings.less > ...
68 // =====
69 // Typography
70 // =====
71
72 @font-family-base: 'Open Sans', sans-serif;
73 @font-base-name: 'Open Sans';
74
75 @font-size-base-px: 16px; // All theme font em is based on this (1em = n:px);
76 @font-size-base-em: 1em; // Default font size
77
78 @font-color-base: @color-black; // Default text color for all page
79 @background-color-base: @color-white; //Color for body BG
80
81 @font-weight-base: normal; //default font weight
```



## STEP #8

# Update the typography

The next set of variables are for the font sizes. The first two are the size for the default paragraph text, and the default form input field text. Then, you'll see that each headline tag, h1 through h6, have their own size which should be set in "em" units – these will reference that base pixel size you set above.

Next, you'll set the font weights. If you need more than two, you can add additional font weight variables here.

The next set of variables are for the font family and the font weight of the headlines.

The following three variables are for the "Text" category micro-themes – by default, we have a "Caption", a "Fine Print", and a "Display Headline", which change the size of the text element when they are applied. Typically, these don't need to be changed, but if your brand has a specific styling for these elements, the size can be updated here.

```
() settings.less ▶ ...
83  | //Font Sizes
84  @font-size-default:      1em;
85  @font-size-input:       1em;
86
87  @font-size-h1:           2.8em;
88  @font-size-h2:           2.2em;
89  @font-size-h3:           2em;
90  @font-size-h4:           1.6em;
91  @font-size-h5:           1.4em;
92  @font-size-h6:           1.2em;
93
94  // Font weights: Set if not using different font files for weights
95  @font-weight-1:          400;
96  @font-weight-2:          700;
97
98  // Headline font families and weights
99  @font-family-h1:          @font-family-base;
100 @font-weight-h1:          @font-weight-1;
101
102 @font-family-h2:          @font-family-base;
103 @font-weight-h2:          @font-weight-1;
104
105 @font-family-h3:          @font-family-base;
106 @font-weight-h3:          @font-weight-1;
107
108 @font-family-h4:          @font-family-base;
109 @font-weight-h4:          @font-weight-1;
110
111 @font-family-h5:          @font-family-base;
112 @font-weight-h5:          @font-weight-1;
113
114 @font-family-h6:          @font-family-base;
115 @font-weight-h6:          @font-weight-1;
116
117 // For the "Text" category microthemes
118 @font-size-caption:       0.85em;
119 @font-size-fineprint:     0.75em;
120 @font-size-display-headline: 140%;
```



## STEP #8

# Update the typography

The next variable will set the amount of indent space on the “Indent” micro-theme.

Then we’ll see the mobile specific font size variables. Most of the time, these will not need to be adjusted, but if your brand specifies mobile font sizes, they can be applied here.

Lastly, we’ll see the variables for line height. These should use a pure number value, so that the line-height will dynamically adjust based on the size set in the Rich Text Editor if it has a set value. We have a separate line height value for the base font size and for headlines.

```
() settings.less ▶ ...
...
122 // For the "indent" microtheme
123 @indent-left: .85em;
124
125 // MOBILE FONT SIZES: Do not change unless brand guidelines specify mobile sizes
126 @font-size-h1-xs: 1.8em;
127 @font-size-h2-xs: 1.6em;
128 @font-size-h3-xs: 1.4em;
129 @font-size-h4-xs: 1.2em;
130 @font-size-h5-xs: 1.1em;
131 @font-size-h6-xs: 1em;

() settings.less ▶ ...
133 // =====
134 // Specify default line height as a pure number value (no px/em), adjusting the value
135 // as desired. This will allow the line-height to flex depending on what size the user
136 // specifies their text within the rich text editor.
137 // =====
138
139 @line-height-base: 1.45;
140 @line-height-headline: 1.45;
```



## STEP #8.1

# Add an additional font and font weight

If your theme has more than one font family, and/or more than two font weights, you'll want to add your additional font family and font weight variables to the appropriate sections in the settings.less file.



**Example:** Add `@font-family-b` and `@font-b-name`, add `@font-weight-3`, etc.

In the next slide, we'll go over adding the additional font families and font weights to the typography.less file; if you don't need to add additional fonts or font weights to your theme, you can move on to Step 10, updating the links and CTA links.

```
settings.less
settings.less | @font-family-b
82
83 // =====
84 // Typography
85 // =====
86
87 @font-family-base: 'Open Sans', sans-serif;
88 @font-base-name: 'Open Sans';
89 @font-family-b: 'Roboto', sans-serif;
90 @font-b-name: 'Roboto';
```

## STEP #9

# Update the typography.less file (if you've added additional font or font weight variables)

(If using a Third Party to host the font files, like Google)

To add a third-party hosted font file, you'll want to start by adding the embed link to the hosted font into the webfonts.txt file. You can do this with the webfonts.txt file that is included in the Master Theme folder, or you can go into the framework in the Ion console and add it directly to the webfonts.txt file.

Copy the embed code from Google, and then open up the webfonts.txt file in the framework. Replace the <link> line with your new embed code, and click Save & Close.

Support post

The screenshot shows a 'Selected family' dialog box for the Roboto font. The 'Review' section shows 'Roboto' selected with 'Regular 400' weight. Below that, there are options to 'Add more styles' and 'Remove all'. The 'Use on the web' section has radio buttons for '<link>' (selected) and '@import'. A code block shows the generated HTML: 

```
<link rel="preconnect" href="https://fonts.gstatic.com">
<link href="https://fonts.googleapis.com/css?family=Roboto&display=swap" rel="stylesheet">
```

 Below this, there are 'CSS rules to specify families' with the rule: 

```
font-family: 'Roboto', sans-serif;
```

 At the bottom, the 'webfonts.txt' file is open in an editor, showing the replacement of the existing code with the new one: 

```
<link href="https://fonts.googleapis.com/css?family=Raleway:300,500" rel="stylesheet">
```





## STEP #9

# Update the typography.less file (if you've added additional font or font weight variables)

(If using font files hosted in the Ion framework)

Start by adding the font face declarations to the top of our typography.less file.

You'll update the font family name, which will be used to reference the font, and the URLs to your files, by copying the file names from your framework and pasting them into the code. The folder structure is not required for the URL of the font files.

Support post

```
() typography.less | ...
1  | -----
2  | Font Faces
3  | ----- */
4  @font-face {
5  |   font-family: 'MyWebFont';
6  |   src: url('webfont.eot'); /* IE9 Compat Modes */
7  |   src: url('webfont.eot?#iefix') format('embedded-opentype'), /* IE6-IE8 */
8  |   url('webfont.woff2') format('woff2'), /* Super Modern Browsers */
9  |   url('webfont.woff') format('woff'), /* Pretty Modern Browsers */
10 |   url('webfont.ttf') format('truetype'), /* Safari, Android, iOS */
11 |   url('webfont.svg#svgFontName') format('svg'); /* Legacy iOS */
12 | }
13
```



## STEP #9

# Update the typography.less file (if you've added additional font or font weight variables)

Once you have your font families imported or declared, you'll add your CSS for the new fonts to the Fonts section, beneath the "Typography" section.

If you have additional font weights, you'll also add your CSS classes for the new font weights in this section.

After the CSS has been added, you'll scroll down to the IXP section and add your new font(s) to the micro-themes.

Save your typography.less file, and go back to your settings.less file.

```
{ } typography.less ▶ ...  
73  /* =====  
74     Fonts  
75  
76     REQUIRED CLASSES: None  
77  ===== */  
78  
79  .font-base {  
80      font-family: @font-family-base !important;  
81      font-weight: @font-weight-1;  
82      h1, h2, h3, h4, h5, h6, .nav, .button {  
83          font-family: @font-family-base !important;  
84          font-weight: @font-weight-1;  
85      }  
86  }  
87  
88  .font-b {  
89      font-family: @font-family-base !important;  
90      font-weight: @font-weight-2;  
91      h1, h2, h3, h4, h5, h6, .nav, .button {  
92          font-family: @font-family-base !important;  
93          font-weight: @font-weight-2;  
94      }  
95  }  
96  
97  /* =====  
98     Font IXP Information  
99  ===== */  
100  
101  .font-base { .ixp-font("@{font-base-name} Regular");}  
102  .font-b { .ixp-font("@{font-base-name} Bold");}  
103
```



## STEP #10

# Update the links

The next two sections in our settings.less file are the Links and Icon Links sections. The first four variables are the inactive link color, the link hover color, the default link font family, and the default link font weight.

The next variable sets the link transition between the inactive state and the hover state.

The last two variables in the default links section set the text decoration of the links, on the inactive and the hover states.

```
() settings.less ▶ ...
141
142 // =====
143 // Link - Default
144 // =====
145
146 @link-color:           @color-c;
147 @link-color-hover:    @color-d;
148 @link-font-family:    @font-family-base;
149 @link-font-weight:    @font-weight-2;
150
151 @link-transition:     all 0.3s ease;
152
153 @link-text:           none; // For text decoration. Underline or none
154 @link-hover-text:    underline; // For text decoration. Underline or none
155
```



## STEP #10

# Update the CTA links

The CTA links section is next. These control the specific CTA link micro-themes, which add icons to the :before or the :after of the link. We use FontAwesome version 4.7 for our icons by default (which can be found in the webfonts.txt file). These are the codes that you will see beside each icon variable - example: @cta-icon-a: '\f105';. There are five options for CTA links by default - arrow, check, link out, caret, and back.

Typically, nothing more needs to be updated. If you have a specific link styling that is beyond these standard options, the links.less file contains all of the styling that targets links.

FontAwesome version 4.7

```
() settings.less > ...
156 |// =====
157 // CTA Links
158 // =====
159
160 @cta-link-text:      none; // For text decoration
161 @cta-link-hover-text: underline;
162
163 @icon-cta-spacing:  ~'.5em'; // default spacing between icon and text
164
165 @cta-pos-x:          ~'100%'; // default cta icon horizontal placement
166 @cta-pos-y:          ~'50%'; // default cta icon vertical placement
167
168 @cta-icon-a-color:   @link-color;
169 @cta-icon-a-color-hover: @link-color-hover;
170 @cta-icon-a-name:    'Arrow';
171 @cta-icon-a:         '\f105';
172
173 @cta-icon-b-color:   @link-color;
174 @cta-icon-b-color-hover: @link-color-hover;
175 @cta-icon-b-name:    'Check';
176 @cta-icon-b:         '\f00c';
177
178 @cta-icon-c-color:   @link-color;
179 @cta-icon-c-color-hover: @link-color-hover;
180 @cta-icon-c-name:    'Link Out';
181 @cta-icon-c:         '\f08e';
182
183 @cta-icon-d-color:   @link-color;
184 @cta-icon-d-color-hover: @link-color-hover;
185 @cta-icon-d-name:    'Caret';
186 @cta-icon-d:         '\f0da';
187
188 @cta-icon-e-color:   @link-color;
189 @cta-icon-e-color-hover: @link-color-hover;
190 @cta-icon-e-name:    'Back';
191 @cta-icon-e:         '\f104';
```



## STEP #11

# Update the list styles

The Bullets section in the Settings.less file will control the default styling of numbered and unordered lists, as well as the two additional Bullet List micro-themes.

The first variable sets the margin for both the unordered and ordered list items. The next variable sets the style for the unordered list default.

The following two variables are specific to the ordered lists. The left margin for the ordered list as a whole is specified as well as the style for the ordered list.

After that there are variables for the Bullet List micro-themes. The first one will set the amount of padding on the list items. Next, we'll see a set of three variables for bullet a, and a set of three variables for bullet b. These will specify the icon, the color, and the micro-theme name for each of the Bullet List styles.

Normally, nothing more will need to be updated, but if there is a specific styling that is beyond these standard options, the lists.less file contains all of the styling that targets list styles.

```
{ } settings.less x
{ } settings.less > ...
192
193 // =====
194 // Bullets
195 // =====
196
197 @li-margin:          10px 0 10px 1em;
198 @li-style:           disc;
199
200 @ol-margin:          10px;
201 @ol-style:           decimal;
202
203 @li-icon-padding:    0 0 0 1.2em;
204
205 @li-a-icon:          '\f138';
206 @bullet-a-color:     @color-black;
207 @bullet-a-name:      'Bullet List - Circle Arrow';
208
209 @li-b-icon:          '\f00c';
210 @bullet-b-color:     @color-black;
211 @bullet-b-name:      'Bullet List - Checkmark';
```

## STEP #12

# Update the “Styling” category

The “Styling” category is the next section in our settings.less file. This section contains the miscellaneous styling elements, like rounded corners, drop shadows, borders, and potentially anything else without a specific category that may be needed in the theme.

The first set of two variables will set the border radius value for the “Rounded Corners” micro-theme, as well as the name of that micro-theme.

The next four variables will set the two default drop shadow styles that we include.

The following four variables will set the two default border styles that we include.

The final two variables set the style of <hr> tags.

Typically, these don't need to be adjusted, but if you have additional stylings that you'd like to include in the “Styling” micro-theme category, the styling.less file contains all of the CSS and IXP information for this category.

```
(1) settings.less ▶ ...
213 // =====
214 // Styling
215 // =====
216
217 @rounded-corners: 10px;
218 @rounded-corners-name: 'Rounded Corners';
219
220 @shadow-a: 0 4px 2px -2px rgba(0,0,0,0.4);
221 @shadow-a-name: 'Drop Shadow Bottom';
222
223 @shadow-b: 1px 1px @color-black, 2px 2px @color-black, 3px 3px @color-black;
224 @shadow-b-name: '@{color-black-name} 3D Shadow';
225
226 @border-style-a: 2px solid @color-a;
227 @border-style-a-name: '@{color-a-name} Border';
228
229 @border-style-b: 2px solid @color-c;
230 @border-style-b-name: '@{color-c-name} Border';
231
232 @hr-color: @color-e;
233 @hr-height: 1px;
234
```



## STEP #13

# Update the buttons

The next section in our settings.less file is the section for all the button stylings.

Buttons have several moving parts involved with them, so let's take a quick look at the overall structure of the buttons variables, and then we'll dig in further.

We start off with some global styles that will be applied to all buttons. Then, we delineate the colors for the six buttons that we have in the code by default.

After that, we set up the sizes for the buttons: small, medium, large, and wide. Then, we'll set up the button icon options, and the padding for those icons.

And lastly, we'll set up the directional buttons.

```
{ } settings.less ▾ ...
236 // =====
237 // Button Styles
238 // =====

{ } settings.less ▾ ...
299 // =====
300 // Button Sizes
301 // =====
302

{ } settings.less ▾ ...
320 // =====
321 // Button Icons
322 // =====
323

{ } settings.less ▾ ...
342 // =====
343 // Button Icon Padding
344 // =====
345

{ } settings.less ▾ ...
350
351 // =====
352 // Directional Button Styles
353 // =====
354
```



## STEP #13

# Update the buttons

## Global Styling

The first variable will set the margin on the button as a whole. The next three variables will be for the border of the button, setting the size, style, and radius. Note: If there is not an obvious border style, a border should still be included – just set the border color and the background color to the same value.

Next, we'll set the transition between the inactive and the hover state of the button.

The next three variables will set up the text for the buttons – the text transform, the font family, and the font weight.

Once these variables have been updated, we'll move on to setting up the button colors.

```
{ } settings.less ▸ ...
236 // =====
237 // Button Styles
238 // =====
239
240 // Global Styles
241 @btn-margin:          0.5em 0;
242 @btn-border-width:    2px;
243 @btn-border-style:    solid;
244 @btn-border-radius:   4px;
245 @btn-transition:      all 0.4s ease-in-out;
246 @btn-text-transform:  none;
247 @btn-font-family:     @font-family-base;
248 @btn-font-weight:     @font-weight-2;
249
```





## STEP #13

# Update the buttons

## Button Colors

You'll see groupings of variables for each button – buttons A through F. By default, our code is set up with button A being the 'main' color of the brand, @color-a; button-b is the outline version of that main color, button C is white, button D is the outline version of white, button E is black, and button F is the outline version of black.

The first two variables in the button grouping are for the background color and the hover background color of the button.

The next two variables in the button grouping are for the text color and the hover text color.

The next two variables in the button grouping are the border color, and the border color on hover.

The last variable in the button grouping is the name of the button, which will show up in the button micro-theme.

```
{ } settings.less ▶ ...
250 // Specific Styles
251 @btn-color-a: @color-a;
252 @btn-color-a-hover: @color-white;
253 @btn-text-a: @color-white;
254 @btn-text-a-hover: @color-a;
255 @btn-border-a: @color-a;
256 @btn-border-a-hover: @color-a;
257 @button-a-name: @color-a-name;
258
259 @btn-color-b: transparent;
260 @btn-color-b-hover: @color-a;
261 @btn-text-b: @color-a;
262 @btn-text-b-hover: @color-white;
263 @btn-border-b: @color-a;
264 @btn-border-b-hover: @color-a;
265 @button-b-name: '{color-a-name} Outline';
266
```



## STEP #13

# Update the buttons

## Button Sizes

By default, size a is “small”, size b is “medium”, size c is “large”, and size d is “wide”, and you’ll see that the variables are structured using this a-b-c-d naming convention.

The first set of four variables will set the font sizes for each of the respective button sizes. The next set of four variables will set the padding for each of the respective sizes. The next variable will set the line height for the “wide” button, specifically — by default, this is set to the line-height-base variable, but can be changed if needed.

The last set of four variables is to set the name of the button size that will be shown in the button micro-themes.

```
{ } settings.less ▶ ...
--
299 // =====
300 // Button Sizes
301 // =====
302
303 @size-a-font-size:      0.8em;
304 @size-b-font-size:      1em;
305 @size-c-font-size:      1.2em;
306 @size-d-font-size:      1.2em;
307
308 @size-a-padding:        1em 2em;
309 @size-b-padding:        1em 2em;
310 @size-c-padding:        1em 2em;
311 @size-d-padding:        1em 2em;
312
313 @size-d-line-height:    @line-height-base;
314
315 @size-a:                ~'small';
316 @size-b:                ~'medium';
317 @size-c:                ~'large';
318 @size-d:                ~'wide';
319
```



## STEP #13

# Update the buttons

## Button Icons

There are four icon buttons by default in a theme, and you'll see the four groups of variables that coincide with each icon button style. The first variable in this section will set the margin on the icon itself, and will apply to all four icon buttons.

The four groups of variables for the icons will control the icon itself, which uses FontAwesome and the name of the icon button.

In the next variable section, we can set a different size for the padding on the icon buttons. By default, they will be the same padding that our normal buttons have.

FontAwesome version 4.7

```
() settings.less > ...
319
320 // =====
321 // Button Icons
322 // =====
323
324 @icon-btn-margin:          0 0 0 .85em;
325
326 @icon-btn-a-icon:         '\f00c';
327 @icon-btn-a-LABEL:       ~'checked'; //descriptive label name here i.e. checked
328 @icon-btn-a:              @icon-btn-a-LABEL;
329
330 @icon-btn-b-icon:         '\f054';
331 @icon-btn-b-LABEL:       ~'arrow';
332 @icon-btn-b:              @icon-btn-b-LABEL;
333
334 @icon-btn-c-icon:         '\f053';
335 @icon-btn-c-LABEL:       ~'back';
336 @icon-btn-c:              @icon-btn-c-LABEL;
337
338 @icon-btn-d-icon:         '\f078';
339 @icon-btn-d-LABEL:       ~'down';
340 @icon-btn-d:              @icon-btn-d-LABEL;
341
342 // =====
343 // Button Icon Padding
344 // =====
345
346 @size-a-icon-padding:     @size-a-padding;
347 @size-b-icon-padding:     @size-b-padding;
348 @size-c-icon-padding:     @size-c-padding;
349 @size-d-icon-padding:     @size-d-padding;
350
```



## STEP #13

# Update the buttons

## Directional Buttons

The final section in the button variables is the directional buttons. These are special buttons that show only arrows when the micro-theme is applied.

The first set of variables control the margin on each arrow and the overall size of the arrow. The next set of variables are the icon codes for each arrow.

Then, we have the variables for the color of the “dark” version of the arrow, and the “light” version of the arrow.

Finally, we have the name of each of the arrows, which will show up in the micro-theme.

Typically, the only option that will be updated in this section is the color of the arrows.

```
{ } settings.less ▶ ...
351 // =====
352 // Directional Button Styles
353 // =====
354
355 @btn-directional-margin: 0;
356 @directional-icon-height: 40px;
357 @directional-icon-width: 40px;
358 @directional-icon-font-size:40px;
359
360 @directional-icon-down: '\f107';
361 @directional-icon-up: '\f106';
362 @directional-icon-left: '\f104';
363 @directional-icon-right: '\f105';
364
365 @dark-arrow-color: @color-black;
366 @light-arrow-color: @color-white;
367
368 @button-up-dark: 'Dark Arrow Up';
369 @button-down-dark: 'Dark Arrow Down';
370 @button-left-dark: 'Dark Arrow Left';
371 @button-right-dark: 'Dark Arrow Right';
372 @button-up-light: 'Light Arrow Up';
373 @button-down-light: 'Light Arrow Down';
374 @button-left-light: 'Light Arrow Left';
375 @button-right-light: 'Light Arrow Right';
376
```



## STEP #13.1

# Add additional buttons (or remove buttons)

If your theme has more than six button styles, you'll want to add additional button variables to the Specific Styles button section in the settings.less file. Alternatively, if your theme requires less than six button styles, you'll also want to update into the buttons.less file.



**Example:** Add a new block of variables starting with @btn-color-g.

In the next slide, we'll go over adding the additional button to the buttons.less file; if you don't need to add additional buttons to your theme, you can move on to Step 15, updating the pods.

```
() settings.less | @btn-color-g
287 @btn-text-c-hover: @color-white;
288 @btn-border-c: @color-white;
289 @btn-border-c-hover: @color-white;
290 @button-c-name: @color-white-name;
291
292 @btn-color-d: transparent;
293 @btn-color-d-hover: @color-white;
294 @btn-text-d: @color-white;
295 @btn-text-d-hover: @color-a;
296 @btn-border-d: @color-white;
297 @btn-border-d-hover: @color-white;
298 @button-d-name: '@{color-white-name} Outline';
299
300 @btn-color-e: @color-black;
301 @btn-color-e-hover: @color-white;
302 @btn-text-e: @color-white;
303 @btn-text-e-hover: @color-black;
304 @btn-border-e: @color-black;
305 @btn-border-e-hover: @color-black;
306 @button-e-name: @color-black-name;
307
308 @btn-color-f: transparent;
309 @btn-color-f-hover: @color-black;
310 @btn-text-f: @color-black;
311 @btn-text-f-hover: @color-white;
312 @btn-border-f: @color-black;
313 @btn-border-f-hover: @color-black;
314 @button-f-name: '@{color-black-name} Outline';
315
316 @btn-color-g: transparent;
317 @btn-color-g-hover: @color-black;
318 @btn-text-g: @color-black;
319 @btn-text-g-hover: @color-white;
320 @btn-border-g: @color-black;
321 @btn-border-g-hover: @color-black;
322 @button-g-name: '@{color-black-name} Outline';
323
```



## STEP #14

# Update the buttons.less file (if you've added additional button variables)

Because buttons have several moving pieces to them, there will be a couple different spots that will need updates within the buttons.less file.

The first section will not be changed — this is the base button styling.

The next section is for the button mixins. When adding a new button, you'll start by copying the `.button-f()` mixin block, and updating it to your new button variables (which we recommend would be `button-g`). If any additional CSS styling is required, that would also go in this block.

The next section is for setting the text color overrides, so that the buttons will always have the correct text color. You'll copy the `.button-f` block here and update it to the new button variables.

```
{ } buttons.less ▶ ...
41  /* =====
42  Button - Mixins
43  ===== */
44  .button-a() { -
54  }
55
56  .button-b() { -
66  }
67
68  .button-c() { -
78  }
79
80  .button-d() { -
90  }
91
92  .button-e() { -
102 }
103
104 .button-f() {
105   background: @btn-co!
106   color: @btn-text-f
107   border-color: @btn-l
108
109   &:hover {
110     background: @btr
111     color: @btn-text!
112     border-color: @
113   }
114 }

{} buttons.less ▶ ...
116 /* =====
117 Button
118 REQUIRED CLASS: .button-a
119 ===== */
120
121 .button-a,
122 .button-container-a @({form-button}) {
123   .button-a();
124
125   &:hover p,
126   &:hover h1,
127   &:hover h2,
128   &:hover h3,
129   &:hover h4,
130   &:hover h5,
131   &:hover h6 {
132     .transition(@btn-transition);
133     color: @btn-text-a-hover !important;
134   }
135 }
136
137 .button-a p,
138 .button-a h1,
139 .button-a h2,
140 .button-a h3,
141 .button-a h4,
142 .button-a h5,
143 .button-a h6 {
144   color: @btn-text-a !important;
145 }
```



## STEP #14

# Update the buttons.less file (if you've added additional button variables)

Nothing will need to be updated in the sizes or icons sections, so the last update will be to the IXP information where you will want to add the additional micro-themes. You'll copy the Button F blocks (including all the Icon micro-themes), and paste them at the bottom. Then, you'll update all the variable letters from "f" to "g" (or whatever you've named your variable).

Save your buttons.less file, and go back to your settings.less file.

```
( ) buttons.less ▶ ...
395 /* =====
396 Button IXP Information
397
398 REQUIRED CLASSES:
399 .button.button-a.button-small
400 .button.button-a.button-medium
401 .button.button-a.button-large
402 .button.button-a.button-wide
403 ===== */
404
405 // Button A
406 .ixp-button('@{button-a-name}') (@{size-a}), a, @size-a;
407 .ixp-button('@{button-a-name}') (@{size-b}), a, @size-b;
408 .ixp-button('@{button-a-name}') (@{size-c}), a, @size-c;
409 .ixp-button('@{button-a-name}') (@{size-d}), a, @size-d;
410
411 // Button A, Icon A
412 .ixp-button-icon('@{button-a-name}@{icon-btn-a}') (@{size-a}), a, @size-a, @icon-btn-a;
413 .ixp-button-icon('@{button-a-name}@{icon-btn-a}') (@{size-b}), a, @size-b, @icon-btn-a;
414 .ixp-button-icon('@{button-a-name}@{icon-btn-a}') (@{size-c}), a, @size-c, @icon-btn-a;
415 .ixp-button-icon('@{button-a-name}@{icon-btn-a}') (@{size-d}), a, @size-d, @icon-btn-a;
416
417 // Button B
418 .ixp-button('@{button-b-name}') (@{size-a}), @size-a;
419 .ixp-button('@{button-b-name}') (@{size-b}), @size-b;
420 .ixp-button('@{button-b-name}') (@{size-c}), @size-c;
421 .ixp-button('@{button-b-name}') (@{size-d}), @size-d;
422
423 // Button B, Icon A
424 .ixp-button-icon('@{button-b-name}@{icon-btn-a}') (@{size-a}), @size-a, @icon-btn-a;
425 .ixp-button-icon('@{button-b-name}@{icon-btn-a}') (@{size-b}), @size-b, @icon-btn-a;
426 .ixp-button-icon('@{button-b-name}@{icon-btn-a}') (@{size-c}), @size-c, @icon-btn-a;
427 .ixp-button-icon('@{button-b-name}@{icon-btn-a}') (@{size-d}), @size-d, @icon-btn-a;
428
429 // Button B, Icon B
430 .ixp-button-icon('@{button-b-name}@{icon-btn-b}') (@{size-a}), @size-a, @icon-btn-b;
431 .ixp-button-icon('@{button-b-name}@{icon-btn-b}') (@{size-b}), @size-b, @icon-btn-b;
432 .ixp-button-icon('@{button-b-name}@{icon-btn-b}') (@{size-c}), @size-c, @icon-btn-b;
433 .ixp-button-icon('@{button-b-name}@{icon-btn-b}') (@{size-d}), @size-d, @icon-btn-b;
434
435 // Button C
436 .ixp-button('@{button-c-name}') (@{size-a}), @size-a;
437 .ixp-button('@{button-c-name}') (@{size-b}), @size-b;
438 .ixp-button('@{button-c-name}') (@{size-c}), @size-c;
439 .ixp-button('@{button-c-name}') (@{size-d}), @size-d;
440
441 // Button C, Icon A
442 .ixp-button-icon('@{button-c-name}@{icon-btn-a}') (@{size-a}), @size-a, @icon-btn-a;
443 .ixp-button-icon('@{button-c-name}@{icon-btn-a}') (@{size-b}), @size-b, @icon-btn-a;
444 .ixp-button-icon('@{button-c-name}@{icon-btn-a}') (@{size-c}), @size-c, @icon-btn-a;
445 .ixp-button-icon('@{button-c-name}@{icon-btn-a}') (@{size-d}), @size-d, @icon-btn-a;
446
447 // Button C, Icon B
448 .ixp-button-icon('@{button-c-name}@{icon-btn-b}') (@{size-a}), @size-a, @icon-btn-b;
449 .ixp-button-icon('@{button-c-name}@{icon-btn-b}') (@{size-b}), @size-b, @icon-btn-b;
450 .ixp-button-icon('@{button-c-name}@{icon-btn-b}') (@{size-c}), @size-c, @icon-btn-b;
451 .ixp-button-icon('@{button-c-name}@{icon-btn-b}') (@{size-d}), @size-d, @icon-btn-b;
452
453 // Button D
454 .ixp-button('@{button-d-name}') (@{size-a}), @size-a;
455 .ixp-button('@{button-d-name}') (@{size-b}), @size-b;
456 .ixp-button('@{button-d-name}') (@{size-c}), @size-c;
457 .ixp-button('@{button-d-name}') (@{size-d}), @size-d;
458
459 // Button D, Icon A
460 .ixp-button-icon('@{button-d-name}@{icon-btn-a}') (@{size-a}), @size-a, @icon-btn-a;
461 .ixp-button-icon('@{button-d-name}@{icon-btn-a}') (@{size-b}), @size-b, @icon-btn-a;
462 .ixp-button-icon('@{button-d-name}@{icon-btn-a}') (@{size-c}), @size-c, @icon-btn-a;
463 .ixp-button-icon('@{button-d-name}@{icon-btn-a}') (@{size-d}), @size-d, @icon-btn-a;
464
465 // Button D, Icon B
466 .ixp-button-icon('@{button-d-name}@{icon-btn-b}') (@{size-a}), @size-a, @icon-btn-b;
467 .ixp-button-icon('@{button-d-name}@{icon-btn-b}') (@{size-b}), @size-b, @icon-btn-b;
468 .ixp-button-icon('@{button-d-name}@{icon-btn-b}') (@{size-c}), @size-c, @icon-btn-b;
469 .ixp-button-icon('@{button-d-name}@{icon-btn-b}') (@{size-d}), @size-d, @icon-btn-b;
470
471 // Button E
472 .ixp-button('@{button-e-name}') (@{size-a}), @size-a;
473 .ixp-button('@{button-e-name}') (@{size-b}), @size-b;
474 .ixp-button('@{button-e-name}') (@{size-c}), @size-c;
475 .ixp-button('@{button-e-name}') (@{size-d}), @size-d;
476
477 // Button E, Icon A
478 .ixp-button-icon('@{button-e-name}@{icon-btn-a}') (@{size-a}), @size-a, @icon-btn-a;
479 .ixp-button-icon('@{button-e-name}@{icon-btn-a}') (@{size-b}), @size-b, @icon-btn-a;
480 .ixp-button-icon('@{button-e-name}@{icon-btn-a}') (@{size-c}), @size-c, @icon-btn-a;
481 .ixp-button-icon('@{button-e-name}@{icon-btn-a}') (@{size-d}), @size-d, @icon-btn-a;
482
483 // Button E, Icon B
484 .ixp-button-icon('@{button-e-name}@{icon-btn-b}') (@{size-a}), @size-a, @icon-btn-b;
485 .ixp-button-icon('@{button-e-name}@{icon-btn-b}') (@{size-b}), @size-b, @icon-btn-b;
486 .ixp-button-icon('@{button-e-name}@{icon-btn-b}') (@{size-c}), @size-c, @icon-btn-b;
487 .ixp-button-icon('@{button-e-name}@{icon-btn-b}') (@{size-d}), @size-d, @icon-btn-b;
488
489 // Button F
490 .ixp-button('@{button-f-name}') (@{size-a}), @size-a;
491 .ixp-button('@{button-f-name}') (@{size-b}), @size-b;
492 .ixp-button('@{button-f-name}') (@{size-c}), @size-c;
493 .ixp-button('@{button-f-name}') (@{size-d}), @size-d;
494
495 // Button F, Icon A
496 .ixp-button-icon('@{button-f-name}@{icon-btn-a}') (@{size-a}), @size-a, @icon-btn-a;
497 .ixp-button-icon('@{button-f-name}@{icon-btn-a}') (@{size-b}), @size-b, @icon-btn-a;
498 .ixp-button-icon('@{button-f-name}@{icon-btn-a}') (@{size-c}), @size-c, @icon-btn-a;
499 .ixp-button-icon('@{button-f-name}@{icon-btn-a}') (@{size-d}), @size-d, @icon-btn-a;
500
501 // Button F, Icon B
502 .ixp-button-icon('@{button-f-name}@{icon-btn-b}') (@{size-a}), @size-a, @icon-btn-b;
503 .ixp-button-icon('@{button-f-name}@{icon-btn-b}') (@{size-b}), @size-b, @icon-btn-b;
504 .ixp-button-icon('@{button-f-name}@{icon-btn-b}') (@{size-c}), @size-c, @icon-btn-b;
505 .ixp-button-icon('@{button-f-name}@{icon-btn-b}') (@{size-d}), @size-d, @icon-btn-b;
506
507 // Button G
508 .ixp-button('@{button-g-name}') (@{size-a}), @size-a;
509 .ixp-button('@{button-g-name}') (@{size-b}), @size-b;
510 .ixp-button('@{button-g-name}') (@{size-c}), @size-c;
511 .ixp-button('@{button-g-name}') (@{size-d}), @size-d;
512
513 // Button G, Icon A
514 .ixp-button-icon('@{button-g-name}@{icon-btn-a}') (@{size-a}), @size-a, @icon-btn-a;
515 .ixp-button-icon('@{button-g-name}@{icon-btn-a}') (@{size-b}), @size-b, @icon-btn-a;
516 .ixp-button-icon('@{button-g-name}@{icon-btn-a}') (@{size-c}), @size-c, @icon-btn-a;
517 .ixp-button-icon('@{button-g-name}@{icon-btn-a}') (@{size-d}), @size-d, @icon-btn-a;
518
519 // Button G, Icon B
520 .ixp-button-icon('@{button-g-name}@{icon-btn-b}') (@{size-a}), @size-a, @icon-btn-b;
521 .ixp-button-icon('@{button-g-name}@{icon-btn-b}') (@{size-b}), @size-b, @icon-btn-b;
522 .ixp-button-icon('@{button-g-name}@{icon-btn-b}') (@{size-c}), @size-c, @icon-btn-b;
523 .ixp-button-icon('@{button-g-name}@{icon-btn-b}') (@{size-d}), @size-d, @icon-btn-b;
524
525 // Button H
526 .ixp-button('@{button-h-name}') (@{size-a}), @size-a;
527 .ixp-button('@{button-h-name}') (@{size-b}), @size-b;
528 .ixp-button('@{button-h-name}') (@{size-c}), @size-c;
529 .ixp-button('@{button-h-name}') (@{size-d}), @size-d;
530
531 // Button H, Icon A
532 .ixp-button-icon('@{button-h-name}@{icon-btn-a}') (@{size-a}), @size-a, @icon-btn-a;
533 .ixp-button-icon('@{button-h-name}@{icon-btn-a}') (@{size-b}), @size-b, @icon-btn-a;
534 .ixp-button-icon('@{button-h-name}@{icon-btn-a}') (@{size-c}), @size-c, @icon-btn-a;
535 .ixp-button-icon('@{button-h-name}@{icon-btn-a}') (@{size-d}), @size-d, @icon-btn-a;
536
537 // Button H, Icon B
538 .ixp-button-icon('@{button-h-name}@{icon-btn-b}') (@{size-a}), @size-a, @icon-btn-b;
539 .ixp-button-icon('@{button-h-name}@{icon-btn-b}') (@{size-b}), @size-b, @icon-btn-b;
540 .ixp-button-icon('@{button-h-name}@{icon-btn-b}') (@{size-c}), @size-c, @icon-btn-b;
541 .ixp-button-icon('@{button-h-name}@{icon-btn-b}') (@{size-d}), @size-d, @icon-btn-b;
542
543 // Button I
544 .ixp-button('@{button-i-name}') (@{size-a}), @size-a;
545 .ixp-button('@{button-i-name}') (@{size-b}), @size-b;
546 .ixp-button('@{button-i-name}') (@{size-c}), @size-c;
547 .ixp-button('@{button-i-name}') (@{size-d}), @size-d;
548
549 // Button I, Icon A
550 .ixp-button-icon('@{button-i-name}@{icon-btn-a}') (@{size-a}), @size-a, @icon-btn-a;
551 .ixp-button-icon('@{button-i-name}@{icon-btn-a}') (@{size-b}), @size-b, @icon-btn-a;
552 .ixp-button-icon('@{button-i-name}@{icon-btn-a}') (@{size-c}), @size-c, @icon-btn-a;
553 .ixp-button-icon('@{button-i-name}@{icon-btn-a}') (@{size-d}), @size-d, @icon-btn-a;
554
555 // Button I, Icon B
556 .ixp-button-icon('@{button-i-name}@{icon-btn-b}') (@{size-a}), @size-a, @icon-btn-b;
557 .ixp-button-icon('@{button-i-name}@{icon-btn-b}') (@{size-b}), @size-b, @icon-btn-b;
558 .ixp-button-icon('@{button-i-name}@{icon-btn-b}') (@{size-c}), @size-c, @icon-btn-b;
559 .ixp-button-icon('@{button-i-name}@{icon-btn-b}') (@{size-d}), @size-d, @icon-btn-b;
560
561 // Button J
562 .ixp-button('@{button-j-name}') (@{size-a}), @size-a;
563 .ixp-button('@{button-j-name}') (@{size-b}), @size-b;
564 .ixp-button('@{button-j-name}') (@{size-c}), @size-c;
565 .ixp-button('@{button-j-name}') (@{size-d}), @size-d;
566
567 // Button J, Icon A
568 .ixp-button-icon('@{button-j-name}@{icon-btn-a}') (@{size-a}), @size-a, @icon-btn-a;
569 .ixp-button-icon('@{button-j-name}@{icon-btn-a}') (@{size-b}), @size-b, @icon-btn-a;
570 .ixp-button-icon('@{button-j-name}@{icon-btn-a}') (@{size-c}), @size-c, @icon-btn-a;
571 .ixp-button-icon('@{button-j-name}@{icon-btn-a}') (@{size-d}), @size-d, @icon-btn-a;
572
573 // Button J, Icon B
574 .ixp-button-icon('@{button-j-name}@{icon-btn-b}') (@{size-a}), @size-a, @icon-btn-b;
575 .ixp-button-icon('@{button-j-name}@{icon-btn-b}') (@{size-b}), @size-b, @icon-btn-b;
576 .ixp-button-icon('@{button-j-name}@{icon-btn-b}') (@{size-c}), @size-c, @icon-btn-b;
577 .ixp-button-icon('@{button-j-name}@{icon-btn-b}') (@{size-d}), @size-d, @icon-btn-b;
578
579 // Button K
580 .ixp-button('@{button-k-name}') (@{size-a}), @size-a;
581 .ixp-button('@{button-k-name}') (@{size-b}), @size-b;
582 .ixp-button('@{button-k-name}') (@{size-c}), @size-c;
583 .ixp-button('@{button-k-name}') (@{size-d}), @size-d;
```



## STEP #15

# Update the pods

The next section in our settings.less file are the Pods. Pods are unique to the Ion platform themes; they are a container (div) that by default adds padding to all four sides, and controls the background color, text color, link color, and button color(s) of the elements inside of it.

In the Pods section, our first variable controls the padding that will be applied to all the pods, on all four sides of the Pod.

The next two sections are the global settings for the “light” pods and the “dark” pods. These will set the text colors, link colors, and button colors that will be referenced for each individual background color, which you will see a bit further down in the code.

```
{ settings.less } ...
376
377 // =====
378 // Pods
379 // =====
380
381 // Pods
382
383 @pod-padding: 1em;
384 //pod a, pod b, pod c, and pod d should not be changed and therefore do not have variables
385
386 // Sets global "light" pod colors
387 @pod-color-light: @color-white;
388 @pod-link-color-light: @color-white;
389 @pod-link-color-hover-light: @color-white;
390 @pod-button-1-light: button-c;
391 @pod-button-2-light: button-d;
392
393 // Sets global "dark" pod colors
394 @pod-color-dark: @font-color-base;
395 @pod-link-color-dark: @link-color;
396 @pod-link-color-hover-dark: @link-color-hover;
397 @pod-button-1-dark: button-a;
398 @pod-button-2-dark: button-b;
```





## STEP #15

# Update the pods

Pods a, b, c, and d are reserved for black, black transparent, white, and white transparent in our platform, so we do not have variables set for those classes in the settings.less file.

Starting with pod e, we'll see a group of variables similar to the buttons. The first variable will set the background color on the pod. The second variable should be set to either "light", or "dark". This will reference the global values that you set above, selecting one of those two blocks for the text, link, and button colors. The last variable will be the name of the pod, which will be shown in the micro-theme.

.form-pod-a is a required CSS class, but you can treat this as either a form-specific pod, or as any other pod, depending on the brand.

Example: If your brand utilizes a white pod behind your forms, you can duplicate the white pod in the form-pod-a. This would ensure that the Quick Starts that use the form-pod styling would pull down with your form-specific pod.

```
() settings.less + ...
400 // Use "light" or "dark" to change the color for text, link, and button styles based on the
    variables set above
401 @pod-e-bg:                @color-a;
402 @pod-e-color:             light;
403 @pod-e-name:               @color-a-name;
404
405 @pod-f-bg:                 @color-b;
406 @pod-f-color:             light;
407 @pod-f-name:               @color-b-name;
408
409 @pod-g-bg:                 @color-c;
410 @pod-g-color:             light;
411 @pod-g-name:               @color-c-name;
412
413 @pod-h-bg:                 @color-d;
414 @pod-h-color:             light;
415 @pod-h-name:               @color-d-name;
416
417 @pod-i-bg:                 @color-e;
418 @pod-i-color:             dark;
419 @pod-i-name:               @color-e-name;
420
421 @pod-j-bg:                 @color-f;
422 @pod-j-color:             light;
423 @pod-j-name:               @color-f-name;
424
425 @pod-k-bg:                 @color-g;
426 @pod-k-color:             light;
427 @pod-k-name:               @color-g-name;
428
429 @pod-l-bg:                 @color-h;
430 @pod-l-color:             light;
431 @pod-l-name:               @color-h-name;
432
433 @form-pod-a-bg:            @color-i;
434 @form-pod-a-color:        dark;
435 @form-pod-a-name:         @color-i-name;
436
```



## STEP #15.1

### Add additional pods

If you added colors to your swatches and those colors should be available as Pods, you'll want to add pod variables for those additional colors.



**Example:** Add a new block of variables starting with `@pod-m-bg`, beneath the `.form-pod-a` block.

In the next slide, we'll go over adding the additional pods to the `podless` file; if you don't need to add additional pods to your theme, you can move on to Step 17, updating the accordions & tabs.

```
() settings.less | @pod-m-bg
437
438 @pod-h-bg: @color-d;
439 @pod-h-color: light;
440 @pod-h-name: @color-d-name;
441
442 @pod-i-bg: @color-e;
443 @pod-i-color: dark;
444 @pod-i-name: @color-e-name;
445
446 @pod-j-bg: @color-f;
447 @pod-j-color: light;
448 @pod-j-name: @color-f-name;
449
450 @pod-k-bg: @color-g;
451 @pod-k-color: light;
452 @pod-k-name: @color-g-name;
453
454 @pod-l-bg: @color-h;
455 @pod-l-color: light;
456 @pod-l-name: @color-h-name;
457
458 @form-pod-a-bg: @color-i;
459 @form-pod-a-color: dark;
460 @form-pod-a-name: @color-i-name;
461
462 @pod-m-bg: @color-j;
463 @pod-m-color: light;
464 @pod-m-name: @color-j-name;
465
466 @pod-n-bg: @color-k;
467 @pod-n-color: light;
468 @pod-n-name: @color-k-name;
469
```

## STEP #16

# Update the pods.less file

(if you've added additional pod variables)

In the pods.less file, you'll see the same overall structure setup that the other interior less files have, with the CSS first, and then the IXP information.

Pods have a slightly different CSS structure within them — they utilize a mixin with two parameters. The first parameter should be the pod color (the “light” or “dark” that will set the text color, link color, and button color), and the second should be the background color of the pod. This mixin can be found in the mixins.less file.

To add a new pod, simply copy the `&.pod-l` block of code (the mixin), and paste it inside the overall `.pod` class, below the `&.pod-l` block. Then, update the values in the mixin to match the variables in your settings.less file.

```
() pods.less ▶ ...
1  | * =====
2  |   Pods
3  |   REQUIRED CLASS: .pod, .pod-a, .pod-b, .pod-c, .pod-form-pod-a
4  |   NOTE: .pod-a, .pod-b, .pod-c style values should not change.
5  |   They should be Transparent (.pod), Transparent White (.pod-b) and Transparent Black (.pod-c)
6  |   always.
7  |   ===== */
8  |   // Pods use a mixin with two parameters.
9  |   // The first parameter is whether the pod should have "light" or "dark" text, set in the
   |   settings.less file.
10 |   // The second parameter is the background color of the pod.
11 |
12 |   .pod {
13 |     padding: @pod-padding;
14 |     .generic-fill {
15 |       .button-a();
16 |     }
17 |     .generic-outline {
18 |       .button-b();
19 |     }
20 |
21 |     &.pod-a {
22 |       .pod-theme-selector(dark, @color-white);
23 |     }
24 |
25 |     &.pod-b {
26 |       .pod-theme-selector(dark, rgba(255, 255, 255, .60));
27 |     }
28 |
29 |     &.pod-c {
30 |       .pod-theme-selector(light, rgba(0, 0, 0, .60));
31 |     }
32 |
33 |     &.pod-d {
34 |       .pod-theme-selector(light, @color-black);
35 |     }
36 |
37 |     &.pod-e {
38 |       .pod-theme-selector(@pod-e-color, @pod-e-bg);
39 |     }
40 |
41 |     &.pod-f {
42 |       .pod-theme-selector(@pod-f-color, @pod-f-bg);
43 |     }
44 |     &.pod-g {
45 |       .pod-theme-selector(@pod-g-color, @pod-g-bg);
46 |     }
47 |   }
48 | }
```

## STEP #16

# Update the pods.less file (if you've added additional pod variables)

Scroll down to the IXP information. You'll see that this is organized in the same order that the Colors and Backgrounds files are; remember, the order of this list will be the order that these micro-themes appear.

Copy one of the pod micro-theme lines, and paste it. Update your variables to match, then save the pods.less file, and go back to your settings.less file.

```
() pods.less ▶ ...
76  /* =====
77     Pod IXP Information
78
79     REQUIRED CLASS: .pod, .pod-a, .pod.form-pod-a
80     ===== */
81
82     .pod { .ixp-pod("Transparent"); }
83     .pod.pod-a { .ixp-pod("@{color-white-name} Pod"); }
84     .pod.pod-b { .ixp-pod("@{color-white-name} Transparent Pod"); }
85     .pod.pod-c { .ixp-pod("@{color-black-name} Transparent Pod"); }
86     .pod.pod-d { .ixp-pod("@{color-black-name} Pod"); }
87
88     // Primary
89     .pod.pod-e { .ixp-pod("@{pod-e-name} Pod"); }
90     .pod.pod-f { .ixp-pod("@{pod-f-name} Pod"); }
91     .pod.pod-g { .ixp-pod("@{pod-g-name} Pod"); }
92     .pod.pod-h { .ixp-pod("@{pod-h-name} Pod"); }
93
94     // Secondary
95     .pod.pod-l { .ixp-pod("@{pod-l-name} Pod"); }
96     .pod.form-pod-a { .ixp-pod("@{form-pod-a-name} Pod"); }
97
98     // Neutral
99     .pod.pod-i { .ixp-pod("@{pod-i-name} Pod"); }
100    .pod.pod-j { .ixp-pod("@{pod-j-name} Pod"); }
101    .pod.pod-k { .ixp-pod("@{pod-k-name} Pod"); }
```



## Update the accordions & tabs

The next sections in the settings.less file are the Accordion settings and the Tab settings. These are very similar in the way they are structured.

Both Accordions and Tabs have a default set of stylings that will be applied automatically without needing to add a micro-theme, and an alternate set of stylings that is shown when a micro-theme is applied to the element.

In the settings, the default styles will come first, then the alternate styles.

```
() settings.less > ...
436
437 |// =====
438 // Accordion RHM
439 // =====
440
441 // Default Accordion
442 @accordion-font-size: ~'1em';
443 @accordion-slide-font-size: ~'1em';
444 @accordion-padding: 5px 1.75em 5px 1em;
445 @accordion-margin: 0 0 5px 0;
446
447 @accordion-open-icon: '\f08d';
448 @accordion-closed-icon: '\f078';
449
450 @accordion-border: none; //Define as none or solid 3px @color
451 @accordion-active-border: none;
452 @accordion-hover-border: none;
453 @accordion-border-radius: 0; //Define as 0 or pixels
454
455 @accordion-toggle-background: @color-e;
456 @accordion-active-background: @color-a;
457 @accordion-hover-background: @color-e;
458
459 @accordion-toggle-color: @font-color-base;
460 @accordion-active-color: @color-white;
461 @accordion-hover-color: @color-a;
462
463 // Alternate Accordion
464 @accordion-a-name: "@(color-a-name) Alternate";
465 @accordion-padding-a: 5px 16px 5px 40px;
466
467 @accordion-a-open-icon: '\f078';
468 @accordion-a-closed-icon: '\f054';
469
470 @accordion-a-border: solid 1px @color-e; //Define as none or solid 3px @color - applies to
bottom
471 @accordion-a-box-shadow: none;
472
473 @accordion-a-toggle-background: @color-white;
474 @accordion-a-active-background: @color-a;
475 @accordion-a-hover-background: @color-a;
476
477 @accordion-a-toggle-color: @font-color-base;
478 @accordion-a-active-color: @color-white;
479 @accordion-a-hover-color: @color-white;
480
```

## STEP #17

# Update the accordions & tabs

## Accordions - Default

The first variable in the Accordion section will set the font size of the accordion toggle. The second variable sets the base font size of the content inside the accordion itself. The next variable sets the padding on the accordion toggle, and the final variable in the first set of variables will set the margin on the toggle.

The next two variables refer to the icon that is shown on the right side of the toggle, and uses FontAwesome by default.

The next set of variables will set a border on the accordion toggle; the inactive state, the active state, and the hover state, as well as delineating whether the border should be rounded or not.

The next three variables will set the toggle background color; the inactive state, the active state, and the hover state.

The last three variables in the default accordion will set the toggle text color for the inactive state, the active state, and the hover state.

```
{ } settings.less ▶ ...
436
437 // =====
438 // Accordion R/W
439 // =====
440
441 // Default Accordion
442 @accordion-font-size: ~'1em';
443 @accordion-slide-font-size: ~'1em';
444 @accordion-padding: 5px 1.75em 5px 1em;
445 @accordion-margin: 0 0 5px 0;
446
447 @accordion-open-icon: '\f00d';
448 @accordion-closed-icon: '\f078';
449
450 @accordion-border: none; //Define as none or solid 3px @color
451 @accordion-active-border: none;
452 @accordion-hover-border: none;
453 @accordion-border-radius: 0; //Define as 0 or pixels
454
455 @accordion-toggle-background: @color-e;
456 @accordion-active-background: @color-a;
457 @accordion-hover-background: @color-e;
458
459 @accordion-toggle-color: @font-color-base;
460 @accordion-active-color: @color-white;
461 @accordion-hover-color: @color-a;
462
```

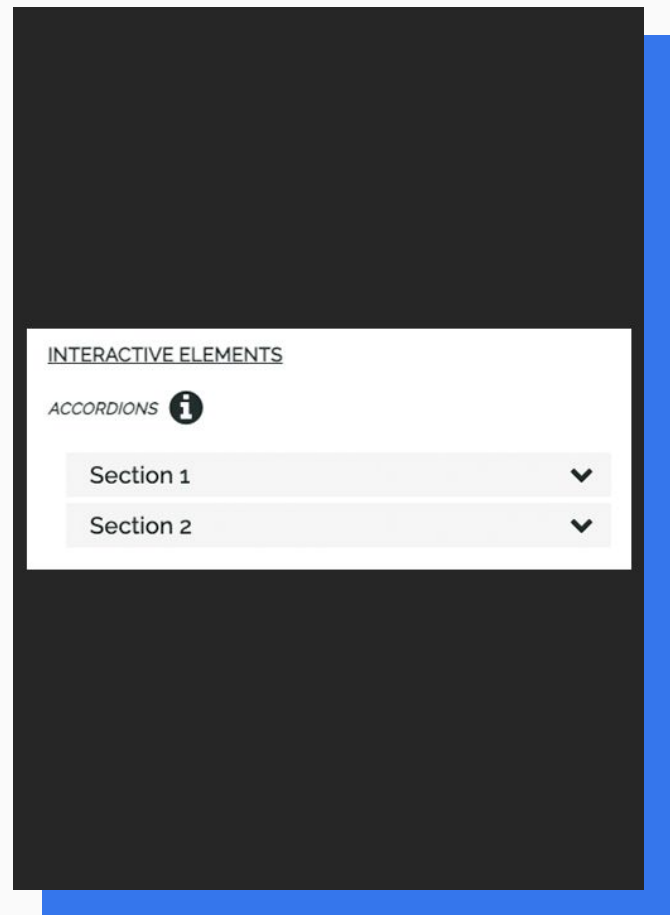


## STEP #17

# Update the accordions & tabs

### Terminology

- The **accordion toggle referenced** is the default state for the clickable accordion bar.
- The **accordion hover referenced** is the hover state for the clickable accordion bar.
- The **accordion active referenced** is the active, or open, state for the clickable accordion bar.



## STEP #17

# Update the accordions & tabs

The accordion toggle icon can be adjusted to be on either the right or left side of the accordion toggle, dependent on brand.

To update this, you'll want to go into the accordions.less file. You'll see an `:after` on the default accordion, and a `:before` on the alternate accordion.

The `:after` placement will be used if you want the icon to be on the right side of the toggle. The `:before` placement will be used if you want the icon to be on the left side of the toggle.

To update the default accordion to have the icon on the left side, copy the `:before` from the alternate CSS code, and paste it into the default, before the `:after`, and replace the `:after` code with `"display:none;"`.



Display:none is important to keep in the code, as this overrides the platform default `:after` on accordion elements.

```
{} accordion.less ▶ .lbul_accordion_v3 ▶ >.lbul_accordion_content ▶ >div ▶ >h2.lbul_trigger ▶ &:after
1
2  -----
3  |     |     |     |     |
4  |     |     |     |     |
5  |     |     |     |     |
6  |     |     |     |     |
7  |     |     |     |     |
8  |     |     |     |     |
9  |     |     |     |     |
10 |     |     |     |     |
11 |     |     |     |     |
12 |     |     |     |     |
13 |     |     |     |     |
14 |     |     |     |     |
15 |     |     |     |     |
16 |     |     |     |     |
17 |     |     |     |     |
18 |     |     |     |     |
19 |     |     |     |     |
20 |     |     |     |     |
21 |     |     |     |     |
22 |     |     |     |     |
23 |     |     |     |     |
24 |     |     |     |     |
25 |     |     |     |     |
26 |     |     |     |     |
27 |     |     |     |     |
28 |     |     |     |     |
29 |     |     |     |     |
30 |     |     |     |     |
31 |     |     |     |     |
32 |     |     |     |     |
33 |     |     |     |     |
34 |     |     |     |     |
35 |     |     |     |     |
36 |     |     |     |     |
37 |     |     |     |     |
38 |     |     |     |     |
39 |     |     |     |     |
40 |     |     |     |     |
41 |     |     |     |     |
```



## Update the accordions & tabs

### Accordions - Alternate

Next is the alternate styling for the accordion. This uses the same font size as the default accordion. The first variable will be the name that is shown on the micro-theme for the accordion. The next variable is the padding for the accordion toggle.

Next, we'll set the variables for the open and close icons that are on the left side of the toggle. The alternate accordion border is only set on the bottom by default, and the next variable contains a box-shadow that can be added if desired.

The next six variables will be the same as the default accordion, setting the toggle background colors in all three states, and the toggle text colors in all three states.

Typically, adjustments don't need to be made beyond these settings, but if your brand guidelines require a specific accordion styling, the `accordion.less` file contains all the CSS styling for the accordions.

```
() settings.less p ...
463 // Alternate Accordion
464 @accordion-a-name:    "@{color-a-name} Alternate";
465 @accordion-padding-a: 5px 16px 5px 40px;
466
467 @accordion-a-open-icon: '\f078';
468 @accordion-a-closed-icon: '\f054';
469
470 @accordion-a-border:  solid 1px @color-e; //Define as none or solid 3px @color - applies to
    bottom
471 @accordion-a-box-shadow: none;
472
473 @accordion-a-toggle-background: @color-white;
474 @accordion-a-active-background: @color-a;
475 @accordion-a-hover-background: @color-a;
476
477 @accordion-a-toggle-color:    @font-color-base;
478 @accordion-a-active-color:    @color-white;
479 @accordion-a-hover-color:    @color-white;
480
```

## STEP #17

# Update the accordions & tabs

## Tabs - Default

Tabs have a similar setup to accordions in their variables. The first variable will set the margin on each individual tab. The second will set the padding for each tab. The third will set the border radius for the tabs themselves.

The next variable is the transition setting between tabs.

The next two variables set the font size for the tab itself, and the font size for the content inside the tab element. The final variable in this group will set a border radius on the tab content, if desired.

The next six variables will be similar to the accordions; they will set the background color of the tabs in all three states, and the text color of the tabs in all three states.

The last group of variables will apply a border, if desired, to the tabs, and a border to the content inside the tabs. The final variable will set a background color on the content inside the tabs.

```
{ } settings.less ▶ ...
481 // =====
482 // Tabs RMW
483 // =====
484
485 // Default Tabs
486 @tab-margin:      0 0 0px 5px;
487 @tab-padding:     12px 25px;
488 @tab-border-radius: 0;
489 @tab-transition:  all 0.3s ease;
490 @tab-font-size:   1em;
491 @tab-slide-font-size: 1em;
492 @tab-slide-border-radius: 0;
493
494 @tab-toggle-background: @color-e;
495 @tab-active-background: @color-a;
496 @tab-hover-background: @color-e;
497
498 @tab-toggle-color:      @font-color-base;
499 @tab-active-color:      @color-white;
500 @tab-hover-color:       @color-a;
501
502 @tab-toggle-border:     none;
503 @tab-active-border:     none;
504 @tab-hover-border:      none;
505 @tab-slides-border:     none;
506 @tab-slide-bg:          @color-white;
507
```

## STEP #17

# Update the accordions & tabs

## Tabs - Alternate

The alternate variables will follow that same structure – the name of the tab is first (which will show up in the micro-theme), then the margin that should be on each tab. The background colors for the tabs is then specified. Followed by the bottom border of the tab, similar to the accordions. A box-shadow can also be added, if desired.

Next, we'll set the text colors for the tabs. And finally, we'll set a border color on the tab content, if desired, and a background color on the tab content.

Usually, nothing will need to be edited beyond this, but if your brand guidelines require a specific tab styling, the tabs.less file contains all the CSS styling for the tabs.

```
{ } settings.less ▶ ...
507
508 // Alternate Tabs
509 @tab-a-name:           '@{color-a-name} Alternate';
510
511 @tab-a-margin:        0 0 5px 5px;
512
513 @tab-a-toggle-background: @color-white;
514 @tab-a-active-background: @color-a;
515 @tab-a-hover-background:  @color-a;
516
517 @tab-a-border-bottom:  solid 2px @color-e;
518 @tab-a-box-shadow:    none;
519
520 @tab-a-toggle-color:   @font-color-base;
521 @tab-a-active-color:   @color-white;
522 @tab-a-hover-color:    @color-white;
523
524 @tab-a-slides-border-color: none;
525 @tab-a-slide-bg:      @color-white;
526
```



## STEP #18

# Update the Flow element

The Flow settings will control the styling for the flow dots, numbers, and progress bar.

The first set of variables are the variables for the colors overall. The first variable will set the color of the off state for the light; the second will set the color of the off state for the dark version. The third will set the color of the hover state and the fourth will set the color of the active state.

The next three will set the color of the text on the numbers progress bar, off state, hover state, and active states respectively.

Then, you'll see the variables for the three specific types of Flow indicator bars – the numbers, the dots, and the progress bar.

```
() settings_less ▶ ...
249
527 // =====
528 // Flow Settings
529 // =====
530
531 @flow-dot-bg:           @color-e; // off state light
532 @flow-dot-bg-dark:     @color-g; // off state dark
533 @flow-dot-bg-hover:    @color-c; // hover state - can be same as active or similar
534 @flow-dot-bg-active:   @color-a; // preferably main color
535 @flow-dot-text:        @color-white;
536 @flow-dot-text-hover:  @color-white;
537 @flow-dot-text-active: @color-white;
538
539 // STYLES FOR NUMBERS
540 @toggle-size: 24px;
541 @toggle-gutter: 6px;
542 @toggle-radius: 3px;
543 @flow-number-font-size: 14px;
544
545 // STYLES FOR SOLID DOTS
546 @dot-size: 20px;
547 @dot-gutter: 4px;
548 @dot-focus-border-size: 2px;
549
550 // STYLES FOR PROGRESS
551 @progress-border-radius: 0;
552 @progress-border-radius-corners: 3px;
553 @progress-height: 12px;
554 @progress-border: none;
555 @progress-margin: 0 1px 0 0;
```



## STEP #18

# Update the Flow element

The numbers variables come first, and they will determine the size of the entire number block, the space between the numbers, the rounding on the corners, and the text size of the numbers.

Next are the dots variables, which will adjust the size of the dot, the space between the dot, and the border size on the active border.

Lastly are the variables for the progress bar styling. The first variable will adjust the radius of each of the individual bars. The second variable will adjust the radius of the first and last bar.

Next is the variable for setting the height of the progress bars; then setting the border; and lastly setting the margin between each bar.

Typically, nothing will need to be edited beyond this, but if your brand guidelines require a specific flow styling, the flow.less file contains all the CSS styling for the flow element.

```
() settings_less > ...
249
527 // =====
528 // Flow Settings
529 // =====
530
531 @flow-dot-bg:           @color-e; // off state light
532 @flow-dot-bg-dark:     @color-g; // off state dark
533 @flow-dot-bg-hover:    @color-c; // hover state - can be same as active or similar
534 @flow-dot-bg-active:   @color-a; // preferably main color
535 @flow-dot-text:        @color-white;
536 @flow-dot-text-hover:  @color-white;
537 @flow-dot-text-active: @color-white;
538
539 // STYLES FOR NUMBERS
540 @toggle-size: 24px;
541 @toggle-gutter: 6px;
542 @toggle-radius: 3px;
543 @flow-number-font-size: 14px;
544
545 // STYLES FOR SOLID DOTS
546 @dot-size: 20px;
547 @dot-gutter: 4px;
548 @dot-focus-border-size: 2px;
549
550 // STYLES FOR PROGRESS
551 @progress-border-radius: 0;
552 @progress-border-radius-corners: 3px;
553 @progress-height: 12px;
554 @progress-border: none;
555 @progress-margin: 0 1px 0 0;
```

## STEP #19

# Update the form stylings

The next section in our settings.less file will handle the styling of the forms within your brand. The majority of the time, you won't need to adjust much in this section.

The first set of variables in this section will be some global variables that will be referenced further down in the form settings, as well as within the form-brand.less file.

```
() settings.less ▶ ...  
557 |// =====  
558 // Freestyle Forms & Forms  
559 // =====  
560  
561 // =====  
562 // CSS and specific style variables for the form elements can be found in the  
563 // form-brand.less file. The variables in the settings.less file are the most common  
564 // variables that need to be adjusted, but if more granular control of styling is  
565 // required, use the form-brand.less file.  
566 // =====  
567
```



## STEP #19

# Update the form stylings

The first grouping of variables will be styling for the input fields.

The first variable is the color for the text in the input fields – not the placeholder color, which is set within our Swatches section at the top of the settings file, but the color of the text when the field has a value.

The next variable is the background color for the input fields. Next, we have two variables for the borders on inputs – a border color, and a border variable that should contain the size, type, and color of the intended border. The color and the color in the full border variable should generally be the same.

The next variable controls the padding of the input fields.

After that is a hover color for the text in the input field, if desired. Then, the hover color for the background of the input fields, if desired. Note: these are rarely updated.

Lastly in this group of variables is the border radius for the input fields.

```
() settings.less | @input-border
257
568 //Global Variables
569 @input-color: @font-color-base; //color for form input (text boxes, etc)
570 @input-bg: @color-white;
571 @input-border-color: @color-f;
572 @input-border: 2px solid @input-border-color;
573 @input-padding: 10px;
574 @input-color-hover: @input-color; //color for form input (text boxes, etc)
575 @input-bg-hover: @color-white;
576 @input-border-radius: 0;
```

## STEP #19

# Update the form stylings

The next group of settings are mainly for support of legacy / older forms (used in the themeForm.less file), and shouldn't really need to be adjusted. The only variable that is used in the Freestyle form (form-brand.less) is the @form-padding variable, which will control the size of the dropdown input padding. This is slightly different padding than the @input-padding variable because of the way that dropdowns are built in the backend of the Ion platform.

```
{ } settings.less | @form-margin
578 @form-margin: 1em 0; // Old Theme Form setting
579 @form-padding: 10px 7px; // Old Theme Form setting -- also used in Freestyle form dropdown
      size
580 @form-cell-padding: .3em .6em; // Old Theme Form setting
581 @form-cell-align: right; // Old Theme Form setting
582 @error-icon-URL: ~'formError.png'; //required for library forms
583 @submit-padding: .4em .5em; // Old Theme Form setting
...

```





## STEP #19

# Update the form stylings

The next three variables will be utilized for the Freestyle forms, and will control the Focus border style, the padding on the validation asterisk, and whether or not the form should have support for UE8 radio/check buttons.

Next, you'll see variables for all the different types of inputs available within Freestyle forms. The majority of the time, none of these values should be changed, as they will be controlled by those global variables that you've set in the above settings, but if your brand requires a specific styling, you can make updates to padding, font sizes, colors, border colors, and validation asterisks as needed.

```
( ) settings.less > ...
584
585 //Freestyle Forms Only
586 @focus: 1px solid @rgba(0,0,0,.5);
587 @validation-padding: 20px;
588 @support-lt-ie9: true; // true = ie8 radio/check support, false = no ie8 support
589
590 //Textbox
591 @form-brand-textbox-padding: @input-padding;
592 @form-brand-textbox-font-size: @font-size-input;
593 @form-brand-textbox-font-color: @input-color;
594 @form-brand-textbox-border-color: @input-border-color;
595 @form-brand-textbox-background-color: @input-bg;
596 @form-brand-textbox-focus-border-color: @active;
597 @form-brand-textbox-focus-background: @input-bg-hover;
598 @form-brand-textbox-error-color: @error;
599 @form-brand-textbox-error-background: @input-bg;
600 @form-brand-textbox-validation-icon-padding: @validation-padding;
601 @form-brand-textbox-validation-asterick-padding: @validation-padding;
602 @form-brand-textbox-placeholder-color: @placeholder-color;
603
604 //Slider
605 @form-brand-slider-bar-color: @input-border-color;
606 @form-brand-slider-border-color: @input-border-color;
607 @form-brand-slider-active-color: @active;
608 @form-brand-slider-handle-bg-color: @input-bg;
609 @form-brand-slider-handle-dot-bg-color: @input-bg;
610 @form-brand-slider-value-bg-color: @input-bg;
611 @form-brand-slider-value-font-color: @input-color;
612
613 //Choice Group
614 @form-brand-choice-group-border-color: @input-border-color;
615 @form-brand-choice-group-background-color: @input-bg;
616 @form-brand-choice-group-selected-color: @active;
617 @form-brand-choice-group-hover-color: @input-border-color;
618 @form-brand-choice-group-error-color: @error;
619 @form-brand-choice-group-focus-color: @focus;
620 @form-brand-choice-group-focus-background: @input-bg;
621 @form-brand-choice-group-checkbox-border-radius: 0;
622 @form-brand-choice-group-radio-outline-color: @peaseaea;
623 @form-brand-choice-group-dot-sprite-image: 'brand_sprite-choice.png'; //for ie 8 and below
624
625 //Choice Group - Border Box
626 @form-brand-choice-group-box-border-color: @color-f;
627 @form-brand-choice-group-box-background-color: transparent;
628 @form-brand-choice-group-box-border-selected-color: @active;
```



## STEP #19

# Update the form stylings

To note, in the Dropdown section of the settings, you'll see a slightly different structure than the rest of the input types. There are two "styles" for the Dropdown fields by default. We use FontAwesome version 4.7 for our icons by default (which can be found in the webfonts.txt file). One will use a FontAwesome icon as the dropdown symbol, and the other will use border styling to create two triangles (one up, and one down) as the dropdown symbol.

@form-brand-dropdown-arrow-style should be set to either "icon" or "border". Icon will use a FontAwesome icon to create the dropdown symbol, and the variables for those should be set in this "FontAwesome icon arrow" section below.

Border will set the up and down arrows as the dropdown symbol, and will use the border color as the arrow color. The size of the arrows can be adjusted by changing the variable under the "CSS border arrows" section below.

FontAwesome version 4.7

```
() settings.less > ...
650 //Dropdown
651 @form-brand-dropdown-sprite: 'brand_sprite-dropdown.png'; //for ie 8 and below
652 @form-brand-dropdown-sprite-2x: 'brand_sprite-dropdown@2x.png'; //for ie 8 and below
653 @form-brand-dropdown-padding: @form-padding;
654 @form-brand-dropdown-font-size: @font-size-input;
655 @form-brand-dropdown-font-color: @input-color;
656 @form-brand-dropdown-border-color: @input-border-color;
657 @form-brand-dropdown-background-color: @input-bg;
658 @form-brand-dropdown-focus-arrow-color: @input-border-color;
659 @form-brand-dropdown-focus-arrow-background: @input-bg-hover;
660 @form-brand-dropdown-error-color: @error;
661 @form-brand-dropdown-placeholder-color: @placeholder-color;
662
663 // ARROWS
664 @form-brand-dropdown-arrow-style: icon; // Set this to "icon" or "border" for the style of
the dropdown arrow
665
666 // FontAwesome icon arrow — USED IF STYLE ABOVE IS SET TO "icon"
667 @form-brand-dropdown-arrow-content: '\f107';
668 @form-brand-dropdown-arrow-container-width: 22px;
669 @form-brand-dropdown-arrow-position-top: 30%;
670 @form-brand-dropdown-arrow-position-right: 0;
671
672 // CSS border arrows (filled-in triangles) — USED IF STYLE ABOVE IS SET TO "border"
673 @form-brand-dropdown-arrow-size: 9px;
674
675 // menu
676 @form-brand-dropdown-menu-color: inherit;
677 @form-brand-dropdown-menu-background-hover: #f0f0f0;
678 @form-brand-dropdown-menu-border-style: solid;
679 @form-brand-dropdown-menu-border-width: 1px;
680 @form-brand-dropdown-menu-border-color: @input-border-color;
681 @form-brand-dropdown-menu-highlighted-background: transparent;
682 @form-brand-dropdown-menu-highlighted-color: @active;
683
684 // search
685 @form-brand-dropdown-search-border: 1px solid #e0e0e0;
686 @form-brand-dropdown-search-input-border: 1px solid #e0e0e0;
687 @form-brand-dropdown-search-input-background: #f0f0f0;
```

## STEP #19

# Update the form stylings

The Library forms section just underneath this should not need to be adjusted.

Most commonly, nothing will need to be edited beyond this, but if your brand guidelines requires a more granular or specific form field styling, the form-brand.less file contains all the CSS styling for the Freestyle form elements.

```
() settings.less ▶ ...
689 // =====
690 // Library Form Drop Downs
691 // =====
692 // =====
693
694 @dropdown-border-radius: 0;
695 @select-icon-URL: ~'icon_select.png'; //required for library forms
696
697 @error-border: 1px solid @error;
698 @error-select-icon-URL: ~'icon_select.png'; //required for library forms
699
700 @select-icon-x: ~'99%'; // default select icon horizontal placement
701 @select-icon-y: ~'48%'; // default select icon vertical placement
702 @select-height: ~'8px'; // The (top padding + bottom padding + font size)
    determines height cross browser
703
```

## Update the regions

The next section in our settings.less file handles what we call “Regions”. Regions are unique to the Ion platform themes. They are a container (div) that by default adds padding to the top and bottom of the container and controls the background color, text color, link color, and button color(s) of the elements inside of it.

In the Regions section, the first two sections are the global settings for the “light” regions and the “dark” regions. These will set the text colors, link colors, and button colors that will be referenced for each individual background color, which you will see a bit further down in the code.

```
{ } settings.less ▶ ...
703
704 // =====
705 // Regions
706 // =====
707
708 // Sets global "light" region colors
709 @region-color-light: @color-white;
710 @region-link-color-light: @color-white;
711 @region-link-color-hover-light: @color-white;
712 @region-button-1-light: button-c;
713 @region-button-2-light: button-d;
714
715 // Sets global "dark" region colors
716 @region-color-dark: @font-color-base;
717 @region-link-color-dark: @link-color;
718 @region-link-color-hover-dark: @link-color-hover;
719 @region-button-1-dark: button-a;
720 @region-button-2-dark: button-b;
721
```

## STEP #20

# Update the regions

The next sections are the variables for each Region. They are structured in the same way as the Pods, with one addition. Each set of Region variables will have its own padding.



As a best practice, these Regions should not have left or right padding – if they do, they will push the Responsive Grid off the browser screen in the SM and XS viewports.

The Regions are:

- Pre-Header
- Header (x2)
- Inner Content Wrapper / Pre-Content
- Content (x3)
- Post-Content
- Footer
- Post-Footer

```
(1) settings.less ▶ ...
721
722 // Use "Light" or "dark" to change the color for text, link, and button styles based on the
723 // variables set above
724
725 // Pre-Header
726 @pre-header-padding: 0.5em 0;
727
728 @pre-header-a-bg: @color-white;
729 @pre-header-a-color: dark;
730 @pre-header-a-name: @color-white-name;
731
732 // Header
733 @header-padding: 1.5em 0;
734
735 @header-a-bg: @color-white;
736 @header-a-color: dark;
737 @header-a-name: @color-white-name;
738
739 @header-b-bg: @color-black;
740 @header-b-color: light;
741 @header-b-name: @color-black-name;
742
743 // Inner Content Wrapper
744 @innercontent-a-bg: @color-white;
745 @innercontent-a-name: @color-white-name;
746
747 // Pre-Content
748 @pre-content-padding: 3em 0;
749
750 // Content
751 @content-padding: 3em 0;
752
753 @content-a-bg: @color-white;
754 @content-a-color: dark;
755 @content-a-name: @color-white-name;
756
757 @content-b-bg: @color-e;
758 @content-b-color: dark;
759 @content-b-name: @color-e-name;
760
761 @content-c-bg: @color-a;
762 @content-c-color: light;
763 @content-c-name: @color-a-name;
```



## STEP #20

# Add additional regions

If you would like to add additional regions, you will want to add the variables to these sections.



**Example:** Add a new block of variables in the Content regions section, starting with `@content-d-bg`.

In the next slide, we'll go over adding the additional regions to the `regions.less` file; if you don't need to add additional regions to your theme, you can move on to Step 22, updating the navigation.

```
{ } settings.less ▶ @content-d-bg
782 // Content
783 @content-padding: 3em 0;
784
785 @content-a-bg: @color-white;
786 @content-a-color: dark;
787 @content-a-name: @color-white-name;
788
789 @content-b-bg: @color-e;
790 @content-b-color: dark;
791 @content-b-name: @color-e-name;
792
793 @content-c-bg: @color-a;
794 @content-c-color: light;
795 @content-c-name: @color-a-name;
796
797 @content-d-bg: @color-j;
798 @content-d-color: light;
799 @content-d-name: @color-j-name;
```



## STEP #21

# Update the regions.less file (if you've added additional region variables)

In the regions.less file, you'll see the same overall structure setup that the other interior files have, with the CSS first, then the IXP information. The regions.less file is also split into sections the way that the settings are, with each Region's CSS and IXP information grouped together.

Regions have the same CSS structure that the pods do – they utilize a mixin with four parameters. The first parameter should be the region color (the “light” or “dark” that will set the text color, link colors, and button colors), the second should be the background color of the region, the third should be the padding, and the fourth should be the font size. This mixin can be found in the mixins.less file.

```
() regions.less ▶ ...
99
100 /* =====
101 Content
102 REQUIRED CLASSES: .content, .content-a, .content-b, .content-c
103 NOTE: Content A should be a white background with styles, content B should be a light gray
104 or very pale color with styles, and content C should be the main brand color or dark color
105 with styles.
106 ===== */
107
108 .content (
109   padding: @content-padding;
110
111   &.content-a {
112     .region-theme-selector(@content-a-color, @content-a-bg, @content-padding,
113       @font-size-base-em);
114   }
115   &.content-b {
116     .region-theme-selector(@content-b-color, @content-b-bg, @content-padding,
117       @font-size-base-em);
118   }
119   &.content-c {
120     .region-theme-selector(@content-c-color, @content-c-bg, @content-padding,
121       @font-size-base-em);
122   }
123 )
124
125 /* =====
126 Content IXP Information
127 REQUIRED CLASSES: .content, .content-a, .content-b, .content-c
128 NOTE: Content A should be a white background with styles, content B should be a light gray
129 or very pale color with styles, and content C should be the main brand color or dark color
130 with styles.
131 ===== */
132
133 .content { .ixp-regions-cl("Content"); }
134 .content.a { .ixp-regions-cl("Content - @({content-a-name})"); }
135 .content.b { .ixp-regions-cl("Content - @({content-b-name})"); }
136 .content.c { .ixp-regions-cl("Content - @({content-c-name})"); }
137
138 ---
```



## STEP #21

# Update the regions.less file (if you've added additional region variables)

To add a new region, scroll to the appropriate region section.



**Example:** If adding a new content region, scroll to the Content section.

Copy the `&.content-c` block of code (the mixin), and paste it inside the overall `.content` class, below the `&.content-c` block. Update your variables to match the variables that you added.

Beneath the CSS, you'll see the IXP information. Copy the `.content.content-c` line, and paste it below. Update your variables, then save your `regions.less` file, and go back to the `settings.less` file.

```
() regions.less ▶ ...
99
100 /* =====
101 Content
102 REQUIRED CLASSES: .content, .content-a, .content-b, .content-c
103 NOTE: Content A should be a white background with styles, content B should be a light gray
104 or very pale color with styles, and content C should be the main brand color or dark color
105 with styles.
106 ===== */
107 .content {
108   padding: @content-padding;
109
110   &.content-a {
111     .region-theme-selector(@content-a-color, @content-a-bg, @content-padding,
112       @font-size-base-em);
113   }
114   &.content-b {
115     .region-theme-selector(@content-b-color, @content-b-bg, @content-padding,
116       @font-size-base-em);
117   }
118   &.content-c {
119     .region-theme-selector(@content-c-color, @content-c-bg, @content-padding,
120       @font-size-base-em);
121   }
122 }
123
124 /* =====
125 Content IXP Information
126 REQUIRED CLASSES: .content, .content-a, .content-b, .content-c
127 NOTE: Content A should be a white background with styles, content B should be a light gray
128 or very pale color with styles, and content C should be the main brand color or dark color
129 with styles.
130 ===== */
131 .content { .ixp-regions-cl("Content");}
132 .content.content-a { .ixp-regions-cl("Content - @({content-a-name})");}
133 .content.content-b { .ixp-regions-cl("Content - @({content-b-name})");}
134 .content.content-c { .ixp-regions-cl("Content - @({content-c-name})");}
135 ---
```





## STEP #22

# Update the navigation

The last section in our settings.less file is the navigation.

There are two navigation styles included in the theme, and the settings for them are structured similarly to the Accordions and Tabs; the default navigation will come first, and then the secondary/alternate navigation.

The first set of variables will be for the overall navigation styles. The first variable will set a background color on the entire navigation element – we recommend this remain transparent, so that the background color can be controlled by the region or container that the navigation is inside.

The next two variables will set the color of the text/links and the font size. The next two will set the margin and padding for the entire navigation element. Then you'll set a text transform, if desired, and the alignment of the text inside the navigation (as a whole), as well as the float of the entire navigation element.

Finally, you'll set the font family and the font weight.

```
( ) settings.less ▶ ...
788 | // =====
789 | // Navigation
790 | // =====
791 |
792 | //Overall nav styles
793 | @nav-background:          transparent; // Entire Nav BG Color
794 | @nav-color:               @color-black; //Text and link color for nav
795 | @nav-font-size:           1em;
796 | @nav-margin:              0;
797 | @nav-padding:             0;
798 | @nav-text-transform:      uppercase;
799 | @nav-text-align:          center;
800 | @nav-float:               right; //floats navigation container to the right side
801 | @nav-font-family:         @font-family-base;
802 | @nav-font-weight:         @font-weight-2;
```



## STEP #22

# Update the navigation

Navigations are built with a couple different pieces, so the next settings that you will see have some “duplicates”. The next set of variables will determine the styling for the <li> element. This is the element that will hold the links inside it, and you’ll have the option to set the float on those specifically.

The next variable will be either true or false, and will set a border between each navigation item.

The next two variables will set the padding and margin on the <li> navigation items.

```
() settings.less > ...
003
004 //NavItem (li) styles
005 @navitem-floats: left; //Floats the lis within the nav UL to the left or right. Avoid
    setting this to right, choose left, none, or inherit.
006 @navitem-border: false; //Will add a border to the right of the link that is the same
    color as @nav-color
007 @navitem-padding: 0;
008 @navitem-margin: 0;
009
```



## STEP #22

# Update the navigation

The next set of variables will affect the navigation `<a>` element. This element is also referred to as the Nav Link, and it sits inside the `<li>` element.

The first variable will set the padding on the `<a>`.

The next six variables will set the background color, the background hover color, the background active color, and then the text color on hover, active, and active hover, respectively.

After the `<a>` variables, you'll have settings for the mobile specific navigation. Typically the only settings that may need to be adjusted in this section are the hamburger bar colors. You'll also have the option to set up the margin on the hamburger menu, the font size, the padding, and the text alignment.

The last option for the default navigation will be the name for the micro-theme on the active state. Typically this shouldn't need to be updated.

```
{ } settings.less > ...
809
810 //Navlink (a) styles
811 @navlink-padding: 0.25em 1.25em;
812 @navlink-background: transparent;
813 @navlink-background-hover: transparent;
814 @navlink-background-active: transparent;
815 @navlink-color-hover: @color-a; //Color for the link hover state
816 @navlink-color-active: @color-a; //Color for the link active state
817 @navlink-color-active-hover:@color-black;
818
819 //Collapsed nav styles (mobile)
820 @nav-collapsed-margin: 0.5em 0;
821 @nav-collapsed-font-size: 110%;
822 @navlink-collapsed-padding: 10px;
823 @nav-text-align-collapsed: center;
824
825 //Hamburger Icon Styles
826 @hamburger-color-open: @color-black;
827 @hamburger-color-closed: @color-black;
828 @hamburger-color-hover: @color-black;
829
830 //Active State Name
831 @active-state-name: 'Active Primary';
832
```



## Update the navigation

The secondary/alternate navigation will have the same settings as the primary/default navigation, with the addition of the first variable, which will set the name for the navigation style that will be shown on the micro-theme.

Usually, nothing will need to be edited beyond this, but if your brand guidelines requires a more granular or specific navigation styling, the navigation.less file contains all the CSS styling for the navigation elements.

```
() settings.less > ...  
  
833 // Navigation A Styling (Secondary Navigation)  
834 @navigation-a-name:      'Secondary Navigation';  
835 @nav-background-a:       transparent; // Entire Nav BG Color  
836 @nav-color-a:            @color-white; //Text and link color for nav  
837 @nav-font-size-a:        1em;  
838 @nav-margin-a:           0;  
839 @nav-padding-a:          0;  
840 @nav-float-a:            right; //floats navigation container to the right side  
841 @nav-text-align-a:       center;  
842 @nav-font-family-a:      @font-family-base;  
843 @nav-font-weight-a:      @font-weight-2;  
844  
845 //Navitem (li) styles  
846 @navitem-float-a:         none;  
847 @navitem-padding-a:       0;  
848 @navitem-margin-a:        0;  
849  
850 //NavLink (a) styles  
851 @navlink-padding-a:       0.25em 1.25em;  
852 @navlink-background-a:    transparent;  
853 @navlink-background-hover-a: transparent;  
854 @navlink-background-active-a: transparent;  
855 @navlink-color-hover-a:   @color-a; //Color for the link hover state  
856 @navlink-color-active-a:  @color-a; //Color for the link active state  
857 @navlink-color-active-hover-a:@color-white;  
858  
859  
860 //Collapsed nav styles (mobile)  
861 @nav-collapsed-margin-a:   0.5em 0;  
862 @navlink-collapsed-padding-a: 10px;  
863 @nav-collapsed-font-size-a: 110%;  
864 @nav-text-align-collapsed-a: center;  
865  
866 //Hamburger Icon Styles  
867 @hamburger-color-open-a:   @color-white;  
868 @hamburger-color-closed-a: @color-white;  
869 @hamburger-color-hover-a:  @color-white;  
870  
871 @active-state-name-a:      'Active Secondary';
```

# Testing & Proofing

---



# Themekit Testing & Proofing

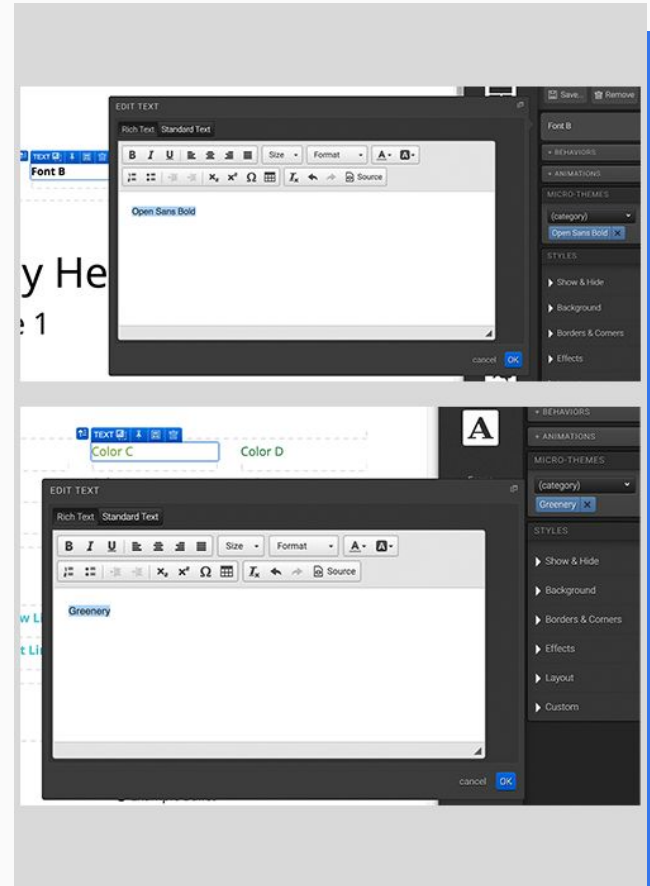
After uploading your CSS files and updating your themekit, it is time to test your new theme.

Updating the themekit page will be the themekit test – you'll want to make sure that every element has the correct micro-theme applied, that the micro-theme is labeled correctly, and that it is styled as expected. You'll want to click every element on this page while testing and proofing.

If you've added additional micro-themes that are not included by default, you'll also want to ensure that you have added those to your themekit page, and applied the micro-themes by using the Micro-theme dropdown (rather than adding the CSS class directly to the element).



**TIP:** Themes can cache, so if you update your CSS files and notice that the themekit page doesn't appear to be updating your styling, try testing in an incognito window, clearing your cache, or doing a hard refresh on your page to clear the cache.



# Browser Testing

If your theme is a standard/typical theme, and you haven't made extensive CSS updates or adjustments, then browser testing will be optional - all of our default settings and stylings are compatible across all major browser versions.

If you've made more specific styling updates, however, we would strongly recommend that you test all major browsers for compatibility.

- Chrome
- Firefox
- Safari
- Edge
- Android
- iOS

```
(.) settings.less > ...
17 // =====
18 // Swatches - For reference in font and background colors
19 // =====
20
21 @color-white:           #FFFFFF; // always must be white
22 @color-white-name:     'White';
23 @color-black:         #000000; // always must be black
24 @color-black-name:    'Black';
25
26 // Primary
27 @color-a:               #86af49; //Main Bright Color (matches to Greenery #86af49)
28 @color-a-name:         'Greenery';
29 @color-b:               #388054; //2nd Bright Color (matches to Deep Lime #388054)
30 @color-b-name:         'Dark Lime';
31 @color-c:               #59c9d5; //3rd Bright Color (matches to Aqua #59c9d5)
32 @color-c-name:         'Aqua';
33 @color-d:               #02939a; //4th Bright Color (matches to Teal #02939a)
34 @color-d-name:         'Teal';
35
36 // Secondary
37 @color-h:               #285124; //Dark Color (matches to Dark Moss #285124)
38 @color-h-name:         'Dark Moss';
39 @color-l:               #c2d7a4; //Light Color (matches to Grass #c2d7a4)
40 @color-l-name:         'Grass';
41
42 // Neutral
43 @color-e:               #f5f9fb; //Off White (matches to #f5f9fb)
44 @color-e-name:         'Light Gray';
45 @color-f:               #8a9a9a; //Gray (matches to #8a9a9a)
46 @color-f-name:         'Medium Gray';
47 @color-g:               #517278; //Mid-Gray (matches to #517278)
48 @color-g-name:         'Dark Gray';
49
```

# Quick Start Testing

## What to look for?

Once your themekit page has been updated and tested, you'll pull three Quick Starts into your Theme campaign and apply your new theme to them. Internally, we typically test the Branded Infographic, Basic Long Page White Paper, and an assessment (like the Assessment with Benchmark Results), to test a variety of experience types.

### 1. Make sure that the colors are mapped correctly.

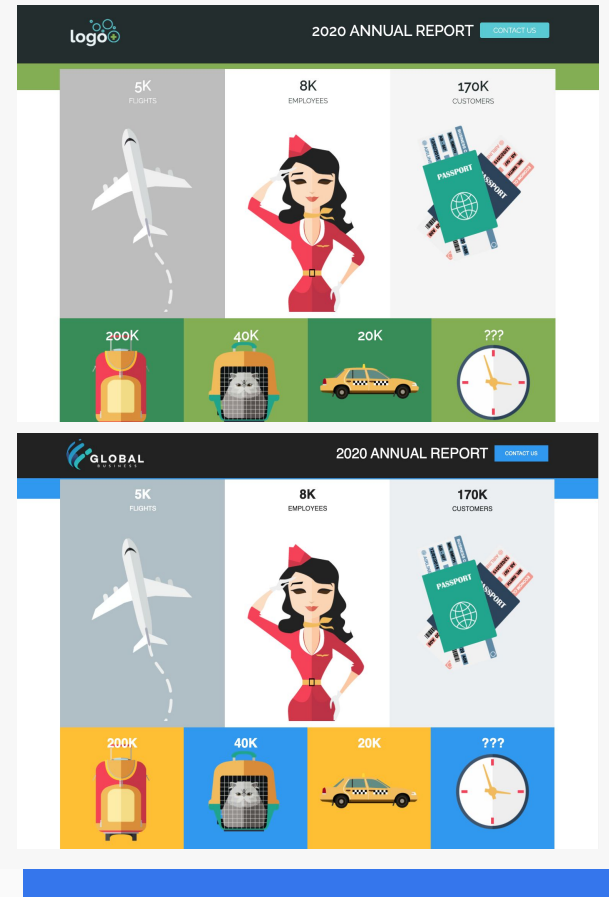
This is where understanding the color mappings will be important. When pulling down Quick Starts to test, you'll want to check that the colors don't conflict with each other, and that the theme has avoided things like having white text on a light background.

### 2. Make sure that the stylings apply properly.

If you've made any special stylings, like with the navigation, tabs, or accordions, you'll want to double check that those apply properly.

### 3. Make sure that the brand looks correct based on guidelines.

Some things will need to be edited during publishing, but you'll want to make sure that the brand's secondary or tertiary colors aren't pulling in as a primary color, that the default fonts are correct, etc.





# How to fix common issues

## My logo isn't sized correctly / the aspect ratio looks wrong.

This can occur if you haven't updated the dimensions of the logo in the settings.less file. There are two sets of dimensions, one for the desktop viewport, and one for the mobile viewport. You'll want to make sure that you've updated both of these dimensions to the correct logo sizes.

If you've added additional logos, you'll want to make sure that you've updated your logo.less file, and used the correct LESS variables for your sizes.

```
{ } settings.less ▶ ...  
56 // =====  
57 // Logo  
58 // =====  
59  
60 @logo-a-URL:          'logo.png';  
61 @logo-a-height:      ~'80px';  
62 @logo-a-width:       ~'136px';  
63 @logo-a-height-sm:   ~'54px';  
64 @logo-a-width-sm:    ~'96px';  
65 @logo-a-name:        'Logo';  
66
```



# How to fix common issues

**My colors aren't looking great on the Quick Starts / I'm seeing light text on light backgrounds or dark text on dark backgrounds.**

This can happen if you haven't set up your Swatches at the top of the settings.less file to match the Quick Start class colors. You'll want to take note of the comments beside each color variable; they will tell you what color they map to in the Quick Start theme, to give you an idea of whether that color should be a light color or a dark color.

If your color palette is smaller or you only have one main primary color, with others being very secondary / sparse, we recommend mapping the main colors all to the same primary color(s), and then adding additional color variables for the secondary colors. This way, the Quick Starts will pull down using just that primary color, and when the designer is publishing, they can use the additional micro-theme colors sparingly as they wish.

```
(.) settings.less > ...
17 // =====
18 // Swatches - For reference in font and background colors
19 // =====
20
21 @color-white:           ■ #FFF; // always must be white
22 @color-white-name:     'White';
23 @color-black:         □ #000; // always must be black
24 @color-black-name:    'Black';
25
26 // Primary
27 @color-a:               ■ #86af49; //Main Bright Color (matches to Greenery #86af49)
28 @color-a-name:         'Greenery';
29 @color-b:               ■ #388054; //2nd Bright Color (matches to Deep Lime #388054)
30 @color-b-name:         'Dark Lime';
31 @color-c:               ■ #59c9d5; //3rd Bright Color (matches to Aqua #59c9d5)
32 @color-c-name:         'Aqua';
33 @color-d:               ■ #02939a; //4th Bright Color (matches to Teal #02939a)
34 @color-d-name:         'Teal';
35
36 // Secondary
37 @color-h:               □ #285124; //Dark Color (matches to Dark Moss #285124)
38 @color-h-name:         'Dark Moss';
39 @color-l:               ■ #c2d7a4; //Light Color (matches to Grass #c2d7a4)
40 @color-l-name:         'Grass';
41
42 // Neutral
43 @color-e:               ■ #f5f9fb; //Off White (matches to #f5f9fb)
44 @color-e-name:         'Light Gray';
45 @color-f:               ■ #8a9a9a; //Gray (matches to #8a9a9a)
46 @color-f-name:         'Medium Gray';
47 @color-g:               ■ #5f7278; //Mid-Gray (matches to #5f7278)
48 @color-g-name:         'Dark Gray';
```

**Thank**  
*you*



**JAMES LUCIA**

Creative & QA Lead  
[james.lucia@rockcontent.com](mailto:james.lucia@rockcontent.com)

