ion: Developer’s Micro-theme Guide v2.0

Updated February 2019

So you want to create plug-and-play designs for the ion platform? Excellent! Here’s what you need to know to get started with Themes and Micro-themes.

Terminology

First, learn the lingo:

You create one or more **Frameworks** that offer one or more **Themes,** which contain **Micro-themes** that can be applied to pages using the Quick Start Cloud Templates.

What are all these different pieces?

A **Theme** is a collection of images and style sheets that are shared by a creative. The Theme is responsible for providing the look-and-feel that is common across all of the pages in the creative. This often includes colors, fonts, graphics, and branding images (i.e. logo). What is cool is that you can create as many Themes as you want for a Framework—although you need at least one—and the user can pick which Theme they want for a particular creative.

Each Theme contains Theme-specific images and CSS, but you can have these images and class styles differ as subtly or as radically as you want. Maybe there’s a “brand x” Theme and a “brand y” Theme in your Framework; maybe there’s a “conservative” Theme and a “wild, Californian surf party” Theme or a mix of the two using **Micro-themes**. The important principle is that all Themes work with the interactive experiences in your console—so a user can switch the look-and-feel of a creative from “brand x” to “brand y” without changing anything else.

Micro-themes make ion Themes extremely flexible and powerful. Micro-themes are on-brand style variations available within a Theme that can be applied to specific elements on the fly.

By using Micro-themes, you will be able to create as many on-brand page style combinations based off your brand guidelines. Micro-themes are content specific. For instance, if you’re editing a link, you’ll see Micro-themes that are specific to styling links. But, if you’re editing text or a container, you’ll see Micro-themes that are specific to those content types.

Think of the **Framework** as a collection of Themes, that contain Micro-themes, that can be applied to your interactive experiences in the ion platform. In other words, a user should be able to construct a creative using one of the Themes, while sticking to on-brand styles. You should utilize the ‘ion_Framework_v4.0’ to upload your Themes into.

The ‘base_layout’ structure**,** included within the ion Framework in your console, is formatted like an actual HTML file. The user will be able to publish your designs, both responsive and non-responsive, using containers and responsive grids in creative studio with base_layout. Therefore, you shouldn’t need to edit this file. This structure is used by Quick Start Cloud templates each console has access to.

Theme Structure

Five files that are required in every Theme directory are `Theme.css`, `form-brand.css`, `ThemeID.xml`, `webfonts.txt`, and `ThemeForm.css` (**NOTE:** the name of the Theme xml file is the same as the subdirectory name in which it is placed).

`Theme.css` specifies the look and feel of the Theme, which contains powerful Micro-theme style options. This often includes font, color, pod, background, button and ready-made-widget options that the user will be able to choose from when creating their pages in ion interactive.

`ThemeID.xml` specifies the title, description of a Theme as well as some parameters related to a responsive template. The Theme's title appears in the ion platform Theme selection drop-down. The format of this XML file should be as follows:

```
<?xml version="1.0" encoding="utf-8"?>

<Theme_properties>

        <label>Theme name</label>

        <description>description of the Theme</description>

        <responsive>true</responsive>

        <supportsgrid>true</supportsgrid>

</Theme_properties>
```

It is easiest not to create this file and rather allow ion to create it for you by copying over the 'Quick Starts 2' Theme and renaming it. See below on details for creating a new Theme.

`webfonts.txt` specifies 3rd-party font solutions to use for the Theme (i.e. Google Fonts). The code placed within this file will be injected within the <head> of your pages allowing you to specify fonts at the Theme-level. (See "SampleTheme/webfonts.txt" for example code). If the font solution requires hosting font files, you would place these files in the Theme's folder and reference the relative path (i.e. src: url(myfont.eot);) to these files. To ensure browser security, please remove `http://` and/or `https://` from any external URLs included within this file.

`form-brand.css` specifies the look and feel (i.e. form fields, error styles, etc…) of the freestyle form components available within the ion platform, as well as Micro-theme variations of these styles. Read more about the ion platform freestyle form components.

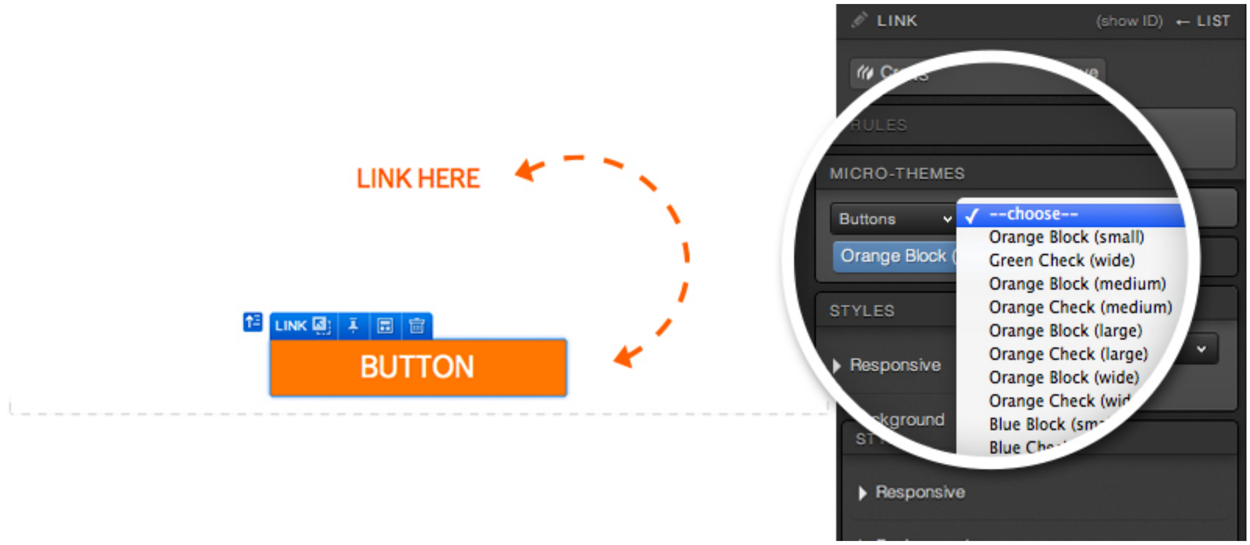
`ThemeForm.css` specifies the look and feel (i.e. form fields, error styles, etc…) of your ion platform library forms, as well as Micro-theme variations of these styles.

NOTE: The `ThemeForm.css` refers to legacy library forms that you might not use. It is a best practice to utilize our freestyle form components instead. Read more about the ion platform library forms.

Creating a Theme using Micro-themes

Now the fun part! Developing a new Theme that includes a wide variety of Micro-themes.

With the ion interactive Quick Start Cloud, your Themes can be very flexible and powerful! Within your Themes, Micro-themes can be defined which will apply styling to specific page elements without requiring additional unique Themes or tedious inline styling. In the following example, the user could add a regular text link to their page and apply the "Buttons" Micro-theme, then select the type of button to apply (i.e. Greenery (medium), Greenery Checked (medium), etc).



By using Micro-themes, you will be able to create hundreds of on-brand page style combinations using a single Theme that is based off of your site's brand guidelines.

Want to implement your own Micro-themes? To get started, we recommend you copy the "Quick Start 2" Theme located in the "ion_Framework_v4.0" Framework in your ion platform. This is the most up-to-date Theme used by the Quick Start Cloud templates. Or you can base your new Theme off of the "SampleTheme" located in the companion files.

An alternative to starting with the CSS files would be to utilize the LESS files we have available. To download the files and learn how to use them, please reference the Master Theme LESS Guide.

Within the Theme.css, platform Micro-theme properties can be applied to the `.style`, `style.substyle`, `:before`, and `:after` type CSS selectors. When the user applies a Micro-theme to an element, the platform will add its class to this element.

The list of Micro-theme properties include:

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

The format of this should be as follows (See "SampleTheme/Theme.css" in the example Theme for additional comments/ examples of all the different Micro-theme properties):

```
.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.

Available scopes:

- **Accordion:** Micro-theme can only be applied to the Accordion component.
- **Any:** Micro-theme can be applied to ANY element.
- **CodeBlock:** Micro-theme can only be applied to code blocks.
- **Column:** Micro-theme can only be applied to columns.
- **Container:** Micro-theme can only be applied to container elements.
- **ContainerLike:** Micro-theme can only be applied to containers, form containers, columns, rows, grids and responsive grids.
- **Flow:** Micro-theme can only be applied to the Flow component.
- **FlowStep:** Micro-theme can only be applied to the interior Flow Step of the Flow component.
- **Form:** Micro-theme can only be applied to forms.
- **FormContainer:** Micro-theme can only be applied to the form's container.
- **FullPageSlider:** Micro-theme can only be applied to the overarching Full Page Slider component.
- **FullPageSection:** Micro-theme can only be applied to the Section or Sub-Section within the Full Page Slider component.
- **Grid:** Micro-theme can only be applied to grids.
- **Iframe:** Micro-theme can only be applied to iframes.
- **Image:** Micro-theme can only be applied to images.
- **Link:** Micro-theme can only be applied to link elements. **NOTE:** A good use for this scope would be for buttons or cta-type link styles.
- **LightboxContainer:** Micro-theme can only be applied to the Lightbox container element.
- **Navigation:** Micro-theme can only be applied the Navigation component and the Navigation Item containers within it.

- **ReadyMadeWidget:** Micro-theme can only be applied to ANY ready-made-widget.
- **ResponsiveGrid:** Micro-theme can only be applied to responsive grids.
- **Reveal:** Micro-theme can only be applied to the Reveal component.
- **Row:** Micro-theme can only be applied to rows.
- **SocialShare:** Micro-theme can only be applied to the SocialShare component.
- **StyleOnly:** Micro-theme can only be applied to styleonly elements that are called Styleable containers within Creative Studio. **NOTES:** Stylelable containers can only contain dynamic elements, they are not the dynamic element itself. For instance, the <body> and the "content_wrap" (from the "Base Layout" master) are Styleable containers.
- **Tabs:** Micro-theme can only be applied to the Tabs component.
- **Text:** Micro-theme can only be applied to text elements.
- **Video:** Micro-theme can only be applied to the Video component.
- **Widget:** Micro-theme can only be applied to widgets.

NOTES:

- Don't forget to review the comments in "/Sample_FW_v1-0/Themes/SampleTheme/Theme.css" file for examples using the Micro-theme properties above.

- When applying your Theme to any of the Quick Start Cloud templates available within your console, please note the required classes (see css comments) within the Theme.css file. Here is a list of some of the required classes for reference:

    - Accordion required classes:

        - `.accordion-a`

    - Backgrounds required classes:

        - `.background-a, .background-b`, and `.background-c`

        - **NOTE:** These 3 required classes (a,b, and c) style values should not change. They should always be White, Transparent White and Transparent Black, respectively.

    - Buttons required classes:

        - `.button-a, .button-small, .button-medium, .button-large, .button-wide, .button.button-a.button-small, .button.button-a.button-medium, .button.button-a.button-large`, and `.button.button-a.button-wide`

    - Colors (for any text or link elements) required classes:

        - `.color-a` and `.color-b`

            - **NOTE:** These 2 required classes (a and b) style values should not be changed. They should always be White and Black, respectively.

    - Images required classes:

        - `.image-a, .image-b`, and `.image-c`

- Link (CTA Links) required classes:
  - `.cta_link.link-a`
- List Items (Bullets) required classes:
  - `.bullet-a`
- Logo required classes:
  - `.header-logo-light`
- Navigation required classes:
  - `.ixp-nav—menu-a` and `.ixp-navitem-active-state-a`
- Pods required classes:
  - `.pod`, `.pod-a`, `.pod-b`, `.pod-c`, and `.pod.form-pod-a`
    - **NOTE:** `.pod`, `.pod-a`, `.pod-b`, and `.pod-c` style values should not change. They should always be Transparent, Transparent White and Transparent Black, respectively.
- Regions required classes:
  - `.pre-header` and `.pre-header-a`
  - `.header` and `.header-a`
  - `.inner-content-wrapper`
    - **NOTE:** This is the container that includes the Pre-Content, Content and Post-Content containers within it.
  - `.pre-content`
  - `.content`, `.content-a`, `.content-b`, and `.content-c`
    - **NOTE:** `.content-a` should always be a white background, `.content-b` should be a light gray or a very pale color, and `.content-c` should be the main brand color or a dark color. Each class should also contain the appropriate styles for the text, buttons and links within it that don't conflict with the background color of the container the Micro-theme is applied to.
  - `.post-content` and `.post-content-a`
  - `.footer` and `.footer-a`
  - `.post-footer` and `.post-footer-a`
- Styling required classes:
  - `.rounded-corners`
- Typography required classes:

- `h1, h2, h3`
- `.font-a`
- `.caption` and `.fine-print`
- `.align-right`, `.align-left`, `.align-center`, and `.indent`

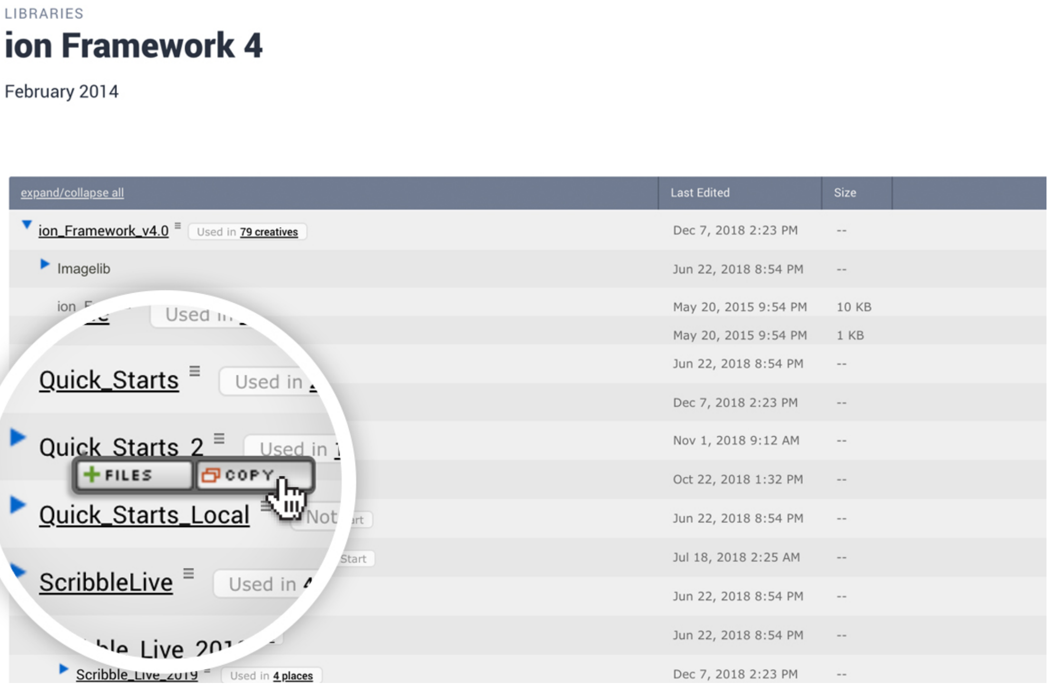
It's best to copy the 'Quick Starts 2' Theme from your console Framework and use this as your base when developing new Themes. The 'Quick Starts 2' Theme is used as the sample Theme for all Quick Start Cloud templates. Basing your Theme off of this one will ensure these templates get stylized with your branded Micro-themes.

That's it, you're done—you've successfully built an ion interactive Theme with Micro-theme options!

Creating a New Theme in the ion console

To create a new Theme, navigate to the 'ion_Framework_v4.0' via the 'Frameworks' link in the 'Libraries' dropdown. Open this Framework, and open the 'Themes' folder within.

Find the 'Quick Starts 2' Theme and hover over the Theme name. You will see a button that says 'COPY'. Press this and insert the name of your new Theme. **NOTE:** the name of the Theme in the Framework cannot be changed once created.



LIBRARIES

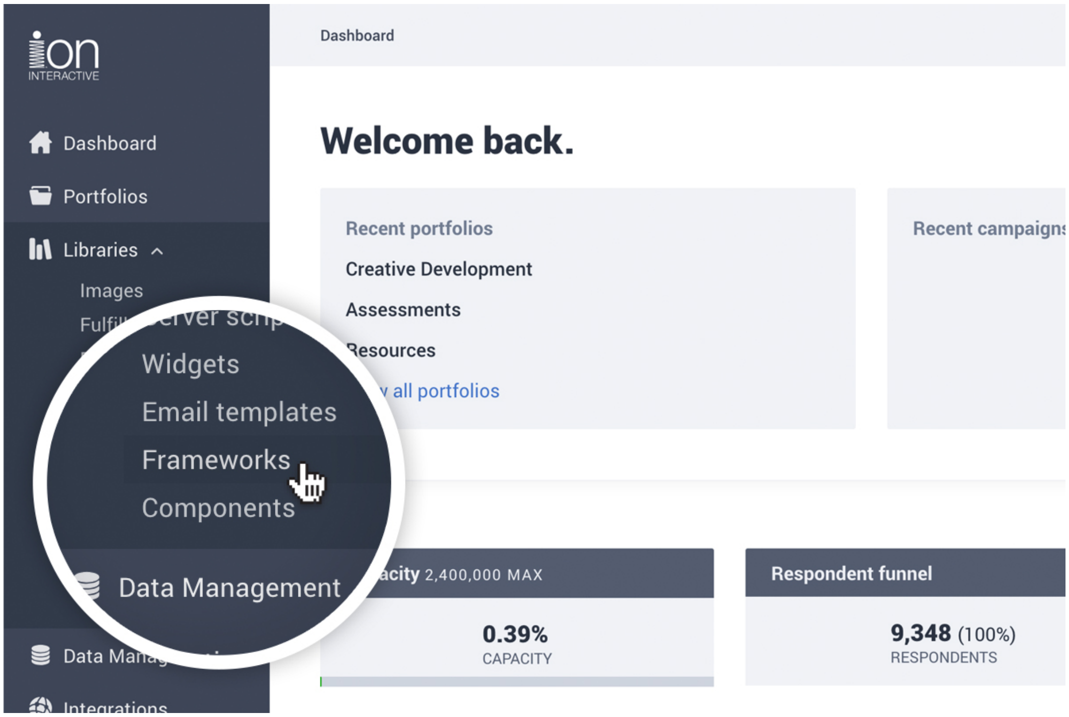
ion Framework 4

February 2014

expand/collapse all	Last Edited	Size
ion_Framework_v4.0 Used in 79 creatives	Dec 7, 2018 2:23 PM	--
Imagelib	Jun 22, 2018 8:54 PM	--
ion	May 20, 2015 9:54 PM	10 KB
	May 20, 2015 9:54 PM	1 KB
	Jun 22, 2018 8:54 PM	--
	Dec 7, 2018 2:23 PM	--
	Nov 1, 2018 9:12 AM	--
	Oct 22, 2018 1:32 PM	--
	Jun 22, 2018 8:54 PM	--
	Jul 18, 2018 2:25 AM	--
	Jun 22, 2018 8:54 PM	--
	Jun 22, 2018 8:54 PM	--
Scribble_Live_2019 Used in 4 places	Dec 7, 2018 2:23 PM	--

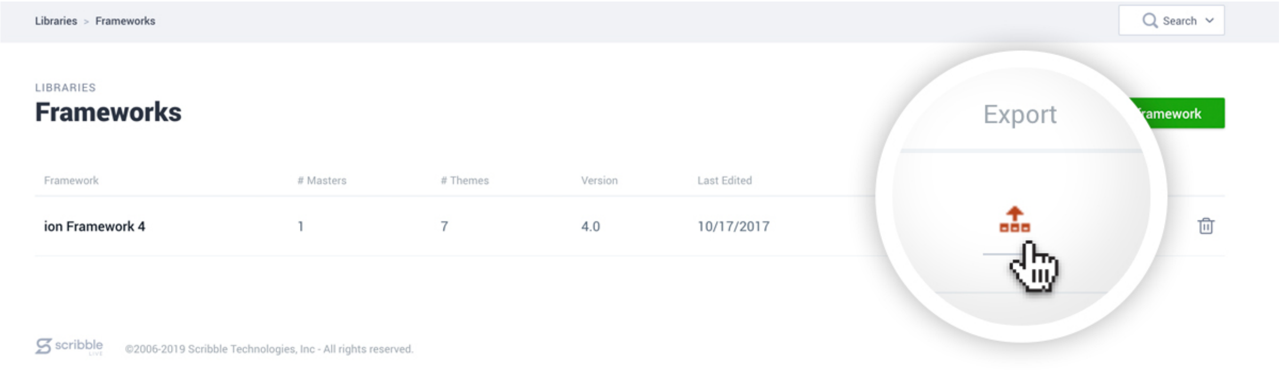
Once the Theme is copied, you can upload files into the Theme by hovering over the name and pressing '+ FILES'. Any files like Theme.css will be overwritten with your new files. You can also copy and paste your Theme directly into the file with the file navigator in ion. To do this, expand the Theme by clicking the blue arrow. Then click the CSS file you want to replace. This will open a code editor that you can copy and paste into. You can also directly edit code in this editor, though this is suggested only for quick, easy edits and not for coding your entire Theme as Themes are not versioned.

Editing the Framework



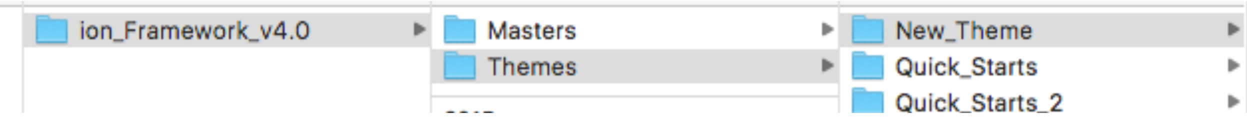
You can edit the Framework to add and create new Masters and Themes. To begin you will want to first download the Framework by navigating to the ‘Frameworks’ library via the ‘Libraries’ dropdown.

Once there, you can then click the ‘Export’ icon at the end of the row of the Framework you’re looking to edit.

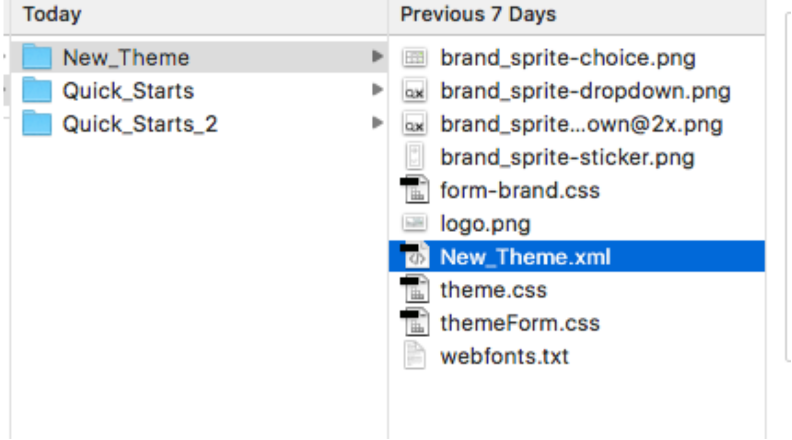


Framework	# Masters	# Themes	Version	Last Edited
ion Framework 4	1	7	4.0	10/17/2017

Your Framework will export in a zipped folder. When you unzip that folder, you will find 2 subfolders, Masters and Themes. To create new Masters and Themes, you will want to add a new folder within those subfolders.



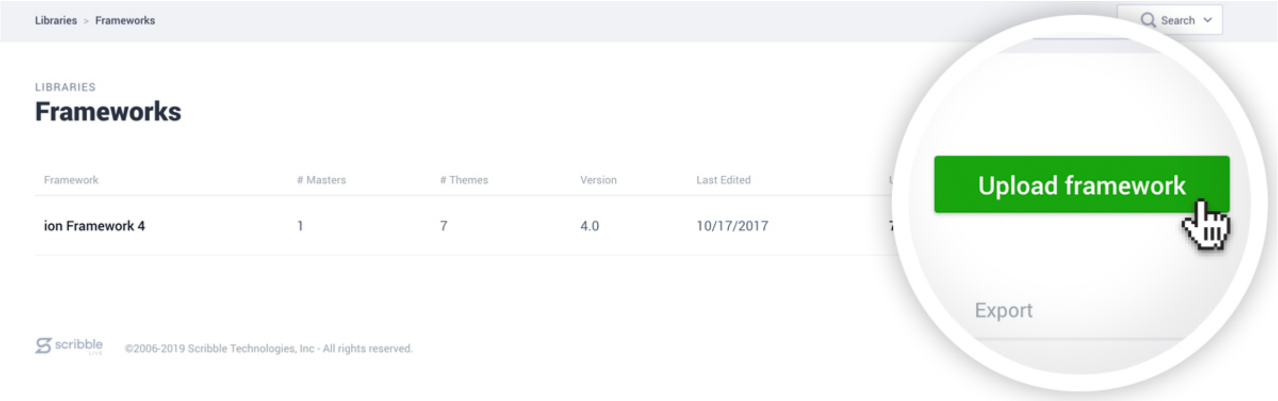
When creating a new theme, you can include compiled CSS files, images, logos, `webfonts.txt` and an `xml` file. **NOTES:** Your xml file must have the same name as your Theme folder. You can copy an xml file from an existing theme to start with, but be sure to update the label and the description tag. The label will be the name of the Theme listed in the dropdown to select when working on a creative.


Today
New_Theme
Quick_Starts
Quick_Starts_2

Previous 7 Days
brand_sprite-choice.png
brand_sprite-dropdown.png
brand_sprite...own@2x.png
brand_sprite-sticker.png
form-brand.css
logo.png
New_Theme.xml
theme.css
themeForm.css
webfonts.txt

Before you upload your updated Framework to the ion platform, you must first select all of your folders, right click and zip those folders. **NOTE**: Some zip/compress utilities add an additional directory in the root of the .zip, which will prevent you from uploading your Framework into the platform. To avoid this, zip the contents of your Framework rather than the initial zip folder that was downloaded. You will want to rename the .zip to match the name of the Framework originally downloaded (remember the .zip must have the same name as the root level .jpg and .xml).

You can then click on the green upload button on the Framework libraries page to upload the compressed Framework and save.

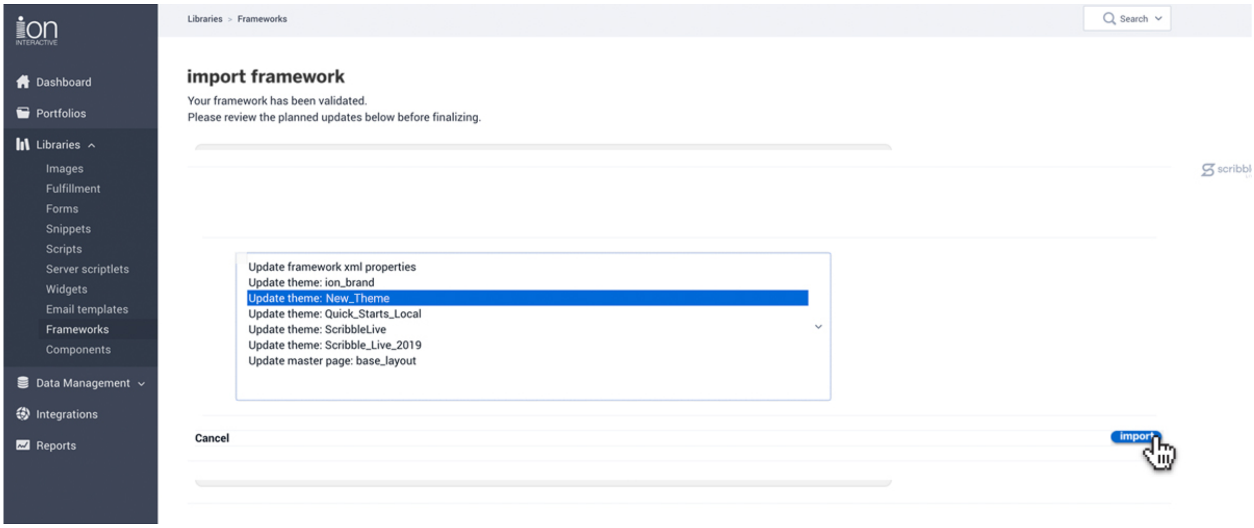


LIBRARIES
Frameworks

Framework	# Masters	# Themes	Version	Last Edited
ion Framework 4	1	7	4.0	10/17/2017

©2006-2019 Scribble Technologies, Inc - All rights reserved.

You will then see a list of items in the next step to review before finalizing the import. When you’re ready, click import and continue.



import framework

Your framework has been validated.
Please review the planned updates below before finalizing.

Update framework xml properties
Update theme: ion_brand
Update theme: New_Theme
Update theme: Quick_Starts_Local
Update theme: ScribbleLive
Update theme: Scribble_Live_2019
Update master page: base_layout

Your Framework is now uploaded and you’re ready to begin building interactive experiences!

In-console Framework management

Once a Framework has been uploaded into ion interactive, the ability to manage/edit the Framework files is available directly through your console.

Navigate to the Framework library and click the icon to the left of the Framework you would like to manage.

Editing the Framework: Within each Framework, you can edit the .css, .ascx, and any .js files directly within your console. Navigate to the file you would like to edit, click on it, make your edits and save.

NOTE: Any saved changes while editing the Framework code will affect **ALL** creatives using the Framework. Be careful!

Managing files: The ability to add/replace, preview, and delete files within a Framework.

To add/replace a file you will see a green pop-up saying ‘+ file’ when hovering your mouse over the Framework folders. Click it and you will get a pop-up window to browse and locate the files. You can also drag and drop multiple files (newer versions of Firefox and Chrome) to the 'add feature'.

To delete a file, click on the trash icon next to the item you would like to delete.

NOTES: If an image is in use you won't be able to delete it. Same applies for other files that are important to the Framework, you won't be able to delete them. Such as .xml, .css, .ascx, etc. You also can't add or replace an .xml file within a Framework.

Copy/Delete Masters and Themes: The ability to copy/create and delete masters and Themes.

To copy a master or Theme you will see a green pop-up saying ‘copy’ when hovering your mouse over the master and Theme folders. Click it and you will get a pop-up window to name it.

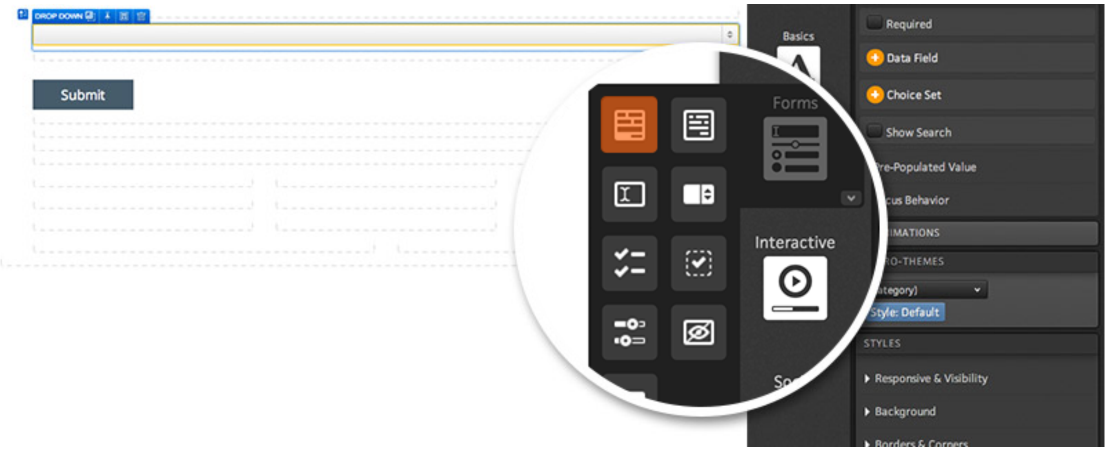
To delete a master or Theme, click on the trash icon next to the item you would like to delete. **NOTE:** If a master or Theme is in use you won't be able to delete it.

A Special Note About Forms

For Freestyle Forms

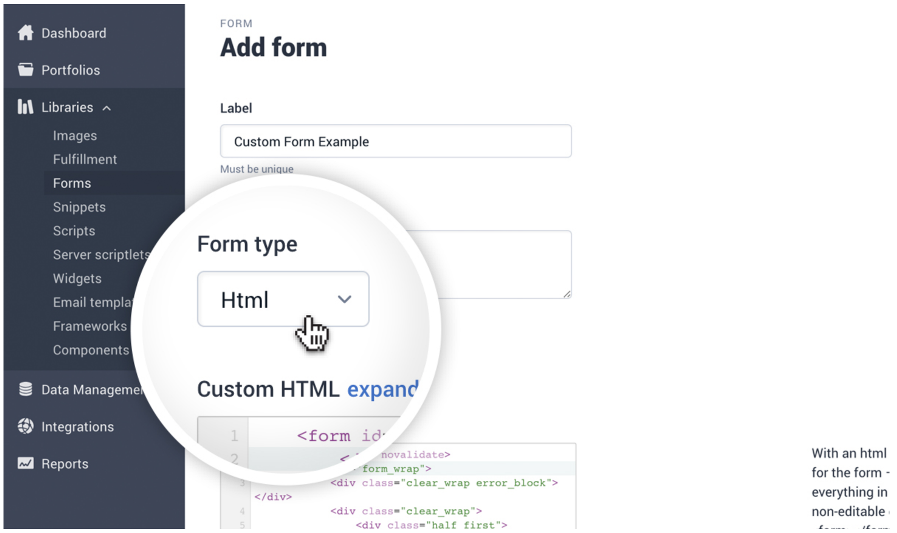
Forms are created and managed by users to give dynamic control of what type of data they're collecting and what they'd like to do with it.

As noted above, you can control the look and feel of your form within the `brand-form.css` file located in your Theme directory. All of the classes within the `brand-form.css` seen within the 'Quick Start 2' and the sample Theme are required. The sample Theme includes comments that help explain what each of the classes are for.



For Library Forms

As noted above, you can control the look and feel of your form within the `Themeform.css` file located in your Theme directory. Again, each `Themeform.css` file must have the same classes for each Theme in your Framework. The sample Theme includes comments that help explain what each of the classes are for.



Label: Custom Form Example

Form type: Html

```
<form id
novalidate>
form_wrap">
<div class="clear_wrap error_block">
</div>
<div class="clear_wrap">
<div class="half first">
```

With an html
for the form -
everything in
non-editable

Image Libraries

Optionally, you can include Framework specific image libraries within your Framework files. These images would then be available to the user while publishing pages dynamically through ion interactive.

Within the root of the FrameworkID directory, include a directory for "imagelib" `(/FrameworkID/imagelib/)`. Within this directory, you will need to include at least one subdirectory.

NOTES: All of your images must be within a subdirectory in the main "imagelib" folder. Only .jpg, .gif and .png files are accepted. These image libraries are Framework specific; they will only be available for use within dynamic image spots using the Framework. The user can always add images to the Image Library themselves.

Additional Resources

The Power of Themes

How to Edit & Create Pages and Creatives

Themes & Micro-Themes

Containers & Grids in Responsive Design

Freestyle Forms Introduction