STYLING THE PLATFORM

STYLING WITH PANACHE

- Once we've built our basic HTML structures through templates, we're ready to set the overall style of Liferay.
- Remember that themes control the basic look and feel across Liferay.
- That means all of our CSS styles will set the tone for our applications and content:
  - Typeface
  - Margins
  - Buttons and links
  - Background colors
  - Foreground colors
  - Highlight and accent colors
- Though we're moving through structure, styling, and configuration one at a time, you'll probably revisit each of these areas many times while putting the finishing touches on your theme.

DEFINING OUR BRAND

- The S.P.A.C.E. brand is defined by a number of items:
  - Typeface
  - Background color
  - Foreground color
  - Secondary and accent colors
  - UI element styles
  - Application border styles
- Many of these branding elements will be contained in our theme's CSS.
- For instance, our custom typeface will be a new font added to the theme.
- Others we can do using custom images.
- Some advanced customizations we'll use other modules or templates for.

GETTING SASSY

- All of Liferay's CSS is built on Sass.
- Sass (for **S**yntactically **A**wesome **S**tyle **S**heets) adds extra features to CSS:
  - Variables
  - Mixins
- You may already be familiar with Sass, or similar technologies like Less.js.
- Some key syntax to look for:
  - *Variables* are written like this: `$variable`
  - *Mixins* are written like this: `@include mixin-name(...)`
    - You may recall *Bourbon* — it has lots of mixins for us to use.
- So you may see stuff like this in our CSS:

```
@include linear-gradient(to top, gray, $brand-color);
```

EXERCISE: ADDING CUSTOM FONTS

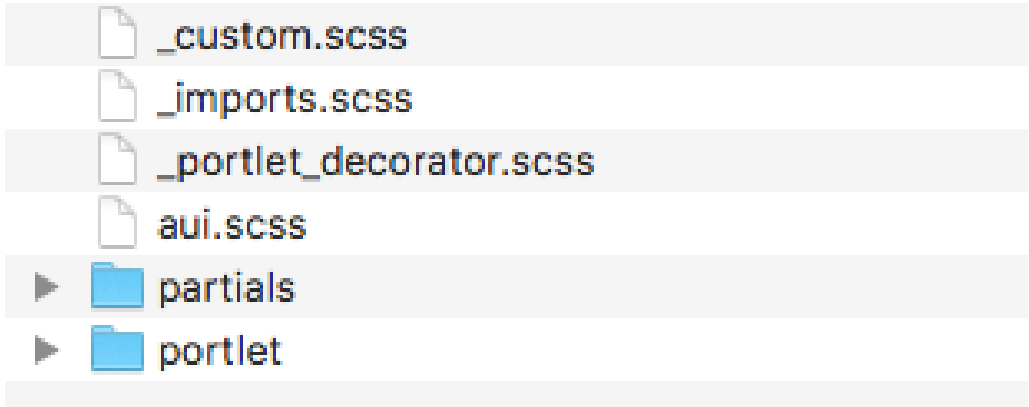
- We want to use the *Open Sans* font in our theme.
- Adding fonts into the theme is as simple as adding the font files and then referencing the fonts in an SCSS file.
- Let's start by adding the font files to our theme.

1. **Go to** the *exercises/front-end-developer-exercises/03-theme-development/01-generating-a-theme/exercise-src* folder.
2. **Copy** the files from the `fonts` folder.
3. **Paste** the files into our new Space Program Theme `src/fonts`.

✓ Now we have fonts we can import!

THE CSS FOLDER STRUCTURE

- You can customize every styled aspect of the platform in a theme.
- The default `src/css` in `build` includes every `.scss` and `.css` file.
- Our theme includes customizations to some main SCSS files and breaks up other customizations into a `partials` and `portlet` folder.
- The `partials` folder includes custom styles for components while the `portlet` folder includes custom styles for our applications.
- We'll walk through adding the styling for some of these sections.



EXERCISE: MODIFYING WITH STYLE

- Next, let's add our CSS structure and walk through our styling.
- First, we'll add our CSS files to modify, and then we'll break down our file structure.

1. **Go to** the *exercises/front-end-developer-exercises/03-theme-development/01-generating-a-theme/exercise-src* folder.
2. **Copy** the files from the `css` folder.
3. **Paste** the files into our new Space Program Theme `src/css`.
    - This will have you replace the default `_custom.scss` in the theme.

EXERCISE: IMPORTING OUR FONTS

- Next, let's add our CSS structure and walk through our styling.
- First, we'll import the fonts we added earlier. This will allow us to use the fonts in our SCSS files and see changes on-the-fly with gulp watch.

1. **Drop** the `_fonts.scss` file from your theme's `src/css/partials` into the Brackets editor.
2. **Open** the *css* section under *snippets*.
3. **Click** on the `01-fonts-scss` snippet.
4. **Copy** the contents of the snippet.
5. **Paste** the snippet contents over the `// Insert snippet 01-font-scss here` comment in the `_fonts.scss` file.
6. **Save** the file.

ADDING FONTS EASILY

- Our font SCSS file uses *mixins* to simplify adding fonts:

```
@include font-face(...);
```

- The `font-face` mixin gives us a quick way to set a bunch of font values in one method.
- For instance, including a new font can be this easy:

```
@include font-face("open_sansregular", "../fonts/opensans-bolditalic-webfont",
```

- And will generate full CSS, something like:

```
@font-face {
    font-family: "open_sansregular";
    font-style: italic;
    font-weight: bold;

    src: url("../fonts/opensans-bolditalic-webfont");
  }
```

- Other mixins are used in similar ways — keep an eye out for them!

TAKING ADVANTAGE OF ATLAS STYLES

- As mentioned earlier, generated themes default to using the *Lexicon Base* theme.
- To include additional styles that are used in the classic theme, we can modify our `aui.scss` file to include the *Atlas theme*.
- The *Atlas theme* provides more styling than is included in the `_styled` theme.
- *Atlas* is Liferay's custom Bootstrap theme that is used in the Classic Theme.
- It overwrites and manipulates Bootstrap and *Lexicon Base* to create the classic Liferay look and feel.
- It is equivalent to installing a Bootstrap third party theme.

EXERCISE: UPDATING THE AUI.CSS FILE

- Let's modify the `aui.scss` to include the *Atlas theme*.

1. **Drop** the `aui.scss` file from your theme's `src/css` into the Brackets editor.
2. **Click** on the `02-aui-scss` snippet.
3. **Copy** the contents of the snippet.
4. **Paste** the snippet contents over the `// Insert snippet 02-aui-scss here` comment in the `aui.scss` file.
5. **Save** the file.

CUSTOMIZING ALL THE SCSS

- The main file for adding any custom style changes is the `_custom.scss` file.
- This is where we can add our global styling for things like:
  - Background color
  - Accent colors
  - Applications Styling
  - Etc.
- The majority of the changes that S.P.A.C.E. requires will go in this file.
- Let's start by adding some basic customization to the file.

EXERCISE: ADDING CUSTOM STYLES

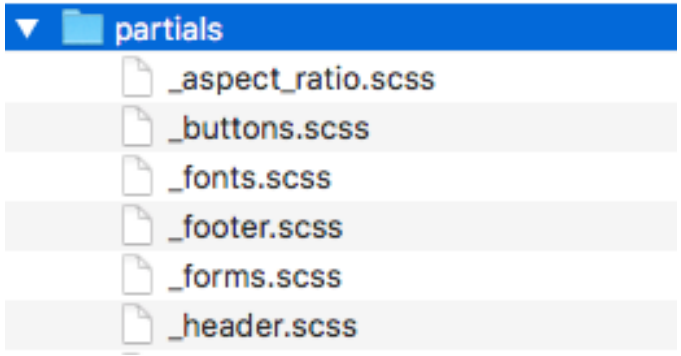
1. **Drop** the `_custom.scss` file from your theme's `src/css` into the Brackets editor.
2. **Click** on the `03-custom-scss` snippet.
3. **Copy** the contents of the snippet.
4. **Paste** the snippet contents over the `// Insert snippet 03-custom-scss here` comment in the `_custom.scss` file.
5. **Save** the file.

KEEPING THINGS ORGANIZED

- Now we have some general style changes we would like to make to the background and our applications.
- Based on the requirements, we still need to customize a number of other things on the platform.
- We could add all of the style changes we need to make here, but it could potentially get a bit unwieldy with hundreds of lines of code.
- What would be a good approach for keeping things organized and clean?
- This is where a more modular approach comes in to play.

MODULARITY WITH STYLE

- We've added the *partials* folder to organize each CSS component we want to customize.
- Each aspect of the platform we want to change will be it's own file.
- For example, we want to change the button styles in our theme, so we have a *_buttons.scss* file in the partials folder.
- All we need to do from there is import the files into our *_custom.scss* file.
- This will make maintanence and organization much better for any theme scss in the future.



EXERCISE: ADDING THE PARTIALS IMPORTS

- Let's go ahead and import the partials styling into our *_custom.scss*.

1. **Click** on the `04-imports` snippet.
2. **Copy** the contents of the snippet.
3. **Paste** the snippet contents over the `// Insert snippet 04-imports here` comment at the top of the `_custom.scss` file.
4. **Save** the file.

✓ Now that they're all imported into the *_custom.scss*, we can add some styling.

INHERITANCE

- Our `_custom.scss` is a good place to some other useful SASS features.
- Sometimes we need to apply styles to areas of the DOM that are logically related.
- For instance, a span inside a div, all wrapped in another div.
- It'd be nice to show this hierarchy.
- Sass lets you do this:

```
.portlet-layout {
    &.row {
        margin: 0;
        .col-md-12 {
            padding: 0;
        }
    }
}
```

- Instead of `.portlet-layout.row`, we can use `&` to show the relationship.

EXERCISE: STYLING THE FOOTER

- As an example of the more modular styles, let's add some styling to our `_footer.scss`.
- This file is being imported in our `_custom.scss` file and will include CSS for the footer.ftl we added earlier.

1. **Drop** the `_footer.scss` file from your theme's `src/css/partials` into the Brackets editor.
2. **Click** on the `05-footer-scss` snippet.
3. **Copy** the contents of the snippet.
4. **Paste** the snippet contents over the `// Insert snippet 05-footer-scss here` comment in the `_footer.scss` file.
5. **Save** the file.

EXERCISE: COLORFUL VARIABLES

- In some of our other `.scss` files, we have been referencing some color variables.
- Having a color scheme represented as `scss` variables can simplify color styling throughout the theme.
- Let's go ahead and set up our color variables.

1. **Drop** the `_color.scss` file from your theme's `src/css/partials/variables` into the Brackets editor.
2. **Click** on the `06-colors-scss` snippet.
3. **Copy** the contents of the snippet.
4. **Paste** the snippet contents over the `// Insert snippet 06-colors-scss here` comment in the `_colors.scss` file.
5. **Save** the file.

EXERCISE: MAKING COLORS MODULAR

- Although it is possible to place any custom variables in the `_variables.scss` file, we have instead added a `variables` folder to keep our variables separate.
- Since our color variables are the only variables added, we'll just import them into our `_variables.scss` file.

1. **Drop** the `_variables.scss` file from your theme's `src/css/partials` into the Brackets editor.
2. **Click** on the `07-variables-scss` snippet.
3. **Copy** the contents of the snippet.
4. **Paste** the snippet contents over the `// Insert snippet 07-variables-scss here` comment in the `_variables.scss` file.
5. **Save** the file.

MODIFYING DEFAULT STYLES

- The base `build` folder includes a number of default styles that can be modified.
- These sections include things like application, navigation, layout styles, and more.
- This folder also includes the `portlet` folder with all the default styles.
- To modify application styles in a theme, you can simply copy that folder and the relevant `.scss` files into your theme's `src` folder.
- We already have the folder, so we can add some variable modifications for our applications.

EXERCISE: MODIFYING APPLICATION CSS

1. **Drop** the `_variables_custom.scss` file from your theme's `src/css/portlet` into the Brackets editor.
2. **Click** on the `08-portlet-variables-custom-scss` snippet.
3. **Copy** the contents of the snippet.
4. **Paste** the snippet contents over the `// Insert snippet 08-portlet-variables-custom-scss here` comment in the `_variables_custom.scss` file.
5. **Save** the file.

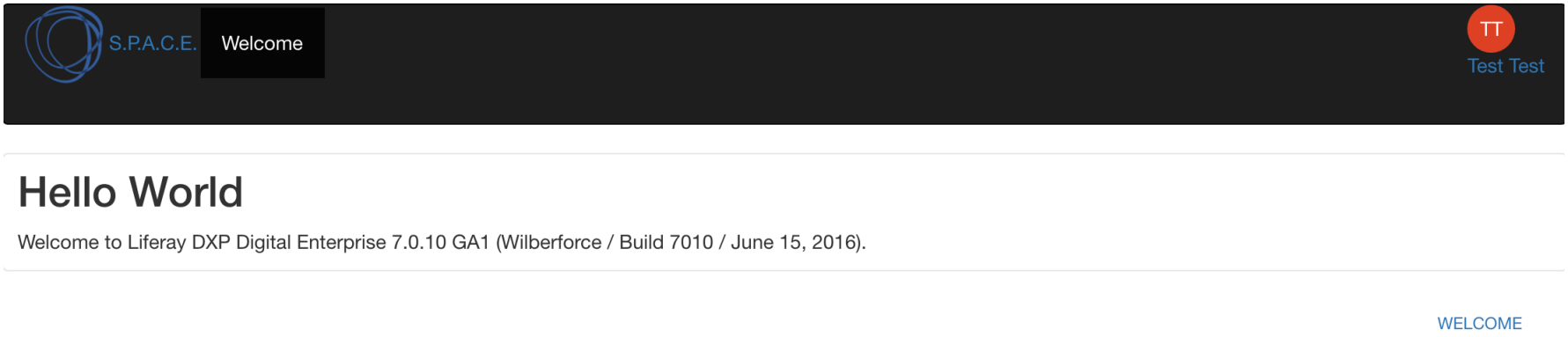
EXERCISE: PROVIDING APPLICATION DECORATOR STYLES

- Application decorators provide borders for all the applications on a page.
- We'll provide styling for the default application decorators in our theme.
- We'll discuss application decorators in more detail in a later section.

1. **Drop** the `_portlet_decorator.scss` file from your theme's `src/css` into the Brackets editor.
2. **Click** on the `09-portlet-decorators-scss` snippet.
3. **Copy** the contents of the snippet.
4. **Paste** the snippet contents over the `// Insert snippet 09-portlet-decorators-scss here` comment in the `_portlet_decorator.scss` file.
5. **Save** the file.
   - ✓ If `gulp watch` is running, the theme will automatically update, if not, run the `gulp deploy` command.

SASS COMPLETE

- Our theme contains a number of other SCSS files for different aspects of the page.
- We've provided the rest of the styles so we can focus on adding the rest of our theme content.
- Now that we have our base html structure and styles down, let's add our theme images, JavaScript, and configurations.



EVEN MORE SASS

- There's a lot you can do with Sass and Bourbon.
- For more information on Sass, check out their site:
  - *http://sass-lang.com*
- Additional features, like a full-featured mixin library, are provided by Bourbon.
- To learn how to use all the advanced tools available through Bourbon, check out:
  - *http://bourbon.io*
- S.P.A.C.E. only uses a small portion of these features in the theme.

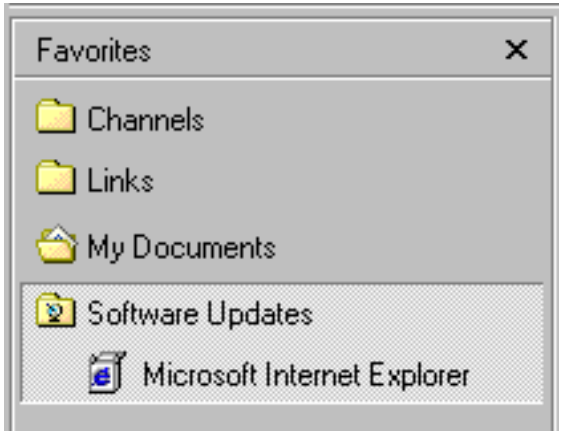
EXERCISE: ADDING IMAGES

- Let's add images first.
- In general, we're going to keep any major images out of our theme.
- Any images needed for a banner or footer content can be added later by a content team.
- Our theme only needs a couple of images to provide a screenshot and thumbnail for use on the platform.

1. **Go to** the *exercises/front-end-developer-exercises/03-theme-development/01-generating-a-theme/exercise-src* folder.
2. **Copy** the files from the `images`.
3. **Paste** the files into our new Space Program Theme `src/images`.

THE REAL FAVICON

- We can also easily add or modify the *favicon* of the theme.
- *Favicons* (or **fav**orite **icons**) are commonly used to make your website's brand recognizable as a small icon. Usually seen on bookmarks, desktop shortcuts, browser tabs, and home screen icons on a mobile device.



- *Favicons* can be stored as *ICO*, *JPG*, or *PNG* files.
- Sizes range from squares of **16x16** and **32x32** up to **310x310** pixels.

EXERCISE: ADDING FAVICONS TO YOUR THEME

- You can modify your favicon very easily by including a `favicon.ico` file in the `src/images` folder of the theme.
- Favicons can be generated on a number of websites, such as *http://realfavicongenerator.net/*
- Let's add a S.P.A.C.E. favicon to our theme.

1. **Copy** the `favicon.ico` from the *exercises/front-end-developer-exercises/03-theme-development/01-generating-a-theme* directory.
2. **Paste** the `favicon.ico` in the space theme's `src/images` folder.

✓ Now we will see our favicon logo!

PAINTING THE PICTURE

- With our styling set using CSS, we have the overall look and feel of the S.P.A.C.E. platform defined.
- Custom images help enhance our brand identity, and give a place to put custom design work.
- Additional little details like Favicons can really tighten up the overall presentation.
- Even though we've tackled one of the biggest visual impacts of the theme, we'll still need look at:
  - Implementing custom JavaScript
  - Finish the theme's configuration

Notes: