Drupal 8 Theming

Exercise 5:

Drupal asset libraries: adding css and javascript to your site

In Drupal 8, "Asset Libraries" are used as the system for loading stylesheets (CSS) and JavaScript (JS). The libraries system that you use in your theme is the same framework used by modules and core. Asset libraries can contain one or more CSS assets, one or more JS assets and one or more JS settings.

Drupal follows this high-level principle: CSS and javascript are only loaded if you tell Drupal it should load them. Drupal has stopped assuming that it should load all assets on all pages, because this is bad for front-end performance.

The biggest difference from Drupal 7 is when it comes to Javascript loaded on pages. By default Drupal doesn't need JavaScript on most pages that anonymous users can see. This means that jQuery is not automatically loaded on all pages anymore. If your theme does need it, you can declare it in one of your theme's libraries and it will be added to every page.

The general process for adding css and javascript

The basic process breaks down into 3 steps

1. Save the CSS or JS to a file.
2. Define a "library" in a `*.libraries.yml file`, which contains a reference to the CSS and/or JS files.
3. "Attach" the library to a render array in a hook using the #attached attribute, by including it in a template, or add the library as a dependency to the theme's `*.info.yml` file.

Create a library

In the following steps we will create a libraries.yml file, declare our library some css files, and create the css files in out theme.

1. Navigate to your theme's root directory

2. Create a file called **acme.libraries.yml** and open that file in your preferred code editor.

3. Add the following code to that file:

```
css-stuff:
  version: VERSION
  css:
    theme:
      css/css-stuff.css: {}
      css/css-stuff-print.css: { media: print }
      //fonts.googleapis.com/css?family=Abhaya+Libre: { type: external }
```

   We just declared our library called `css-stuff`. We can give it any name we want as long as another module or theme hasn't declared it. A best practice for theme libraries is to prefix with the name of the theme and some word(s) to describe what the library will do. A name like `acme-global` would be more accurate for this library, but I'm a free spirit and I like to run with scissors.

4. Navigate back to your theme's root directory, and create a folder called **"css"** (if one doesn't already exist).

5. Create two files in that directory. One called `css-stuff.css` and one called `css-stuff-print.css`

6. Open them both and **add any high level css you want**. For `css-stuff.css` you could add:

```
body {
  background-color: gray;
}

h2 {
  font-family: 'Abhaya Libre', serif;
}
```

   and for `css-stuff-print.css`, you could add:

```
body {
 font-family: sans-serif;
 color: #888;
}
```

   Go crazy if you want!

7. Navigate back to your theme root directory and open your *acme.info.yml* file

8. We need to make our theme aware of our newly declared library before anything will happen.
9. Add the following code to your *acme.info.yml* file

```
libraries:
  - acme/css-stuff
```

10. Clear your caches and you should now see your css styles in place.

If everything worked, you should see output like the following in the head section on all pages of your theme.

```
@import url("/themes/acme/css/css-stuff.css?of7sd1");
...
<style media="print">
    @import url("/themes/acme/css/css-stuff-print.css?of7sd1");
</style>
```

Attach Library to a template

In Drupal 8, it is recommended that libraries only be applied when needed. Unless a style is being used on every page we should use the `attach_library()` function in twig to add js or css.

1. Create a folder called `js` in your theme root and add a javascript file called **widget.js**
2. Create a `css` file in the **css** folder called **widget.css**.
3. Add some sample css and js to each file:

   widget.js:

   ```
   console.log('It Works')
   ```

   widget.css:

   ```
   article {
     background: white;
   }
   ```

4. Define your new library in **acme.libraries.yml**

```
widget:
  version: VERSION
  js:
     js/widget.js: {}
  css:
    theme:
      css/widget.css: {}
  dependencies:
     - core/drupal
     - core/jquery
```

5. Navigate to **MYDRUPAL/core/themes/classy/templates/content/** and look for `node.html.twig` .
6. Make a copy of node.html.twig and paste it in at **MYDRUPAL/themes/acme/templates/node** .
7. Add the following to **node.html.twig** near the top

```
{% if node.bundle == 'article' %}
  {{ attach_library('acme/widget') }}
{% endif %}
```

8. Verify that widget.js and widget.css have loaded on the article pages but NOT on basic pages.

Overriding and Removing libraries

Removing css or javascript

If we are using a base theme, it is very possible that some css or js is coming over from the base theme that we just do not want to use in our theme. Or sometimes there is some css coming from core that we just don't want. Even though Drupal tries to follow the rule of only "load stuff only if needed", You may find yourself in a situation where a contrib module or other library is just adding to much. We can remove stylessheets from output at the theme level by using the `stylesheets-remove` key in our *.info.yml file

This method is for removing individual css or javascript files

1. Take a look at the css that is being loaded on your site as a logged in user. Note that `/core/assets/vendor/jquery.ui/themes/base/dialog.css` and `/core/themes/classy/css/components/breadcrumb.css` are being loaded.
2. Navigate to your theme root and open the _acme.info.yml file

3. Place the following code in your file

```
stylesheets-remove:
  - core/assets/vendor/jquery.ui/themes/base/dialog.css
  - '@classy/css/components/breadcrumb.css'
```

In this example we are removing the extra css file that core tries to add with one of its jquery libraries. We are also removing a stylesheet from our parent (base) theme, classy. You probably notice the `@` symbol at the start of the third line. This is a placeholder token. @classy, or @node, or @whatever is the equivalent to `drupal_get_path()` in Drupal 7.

Overriding css or javascript

Sometime we don't want to remove a file or a library, but we do have a better file for it to use. We can override whole libraries or components of them. Below is an example of libraries-override from drupal.org. You would place this in your *.info.yml file.

This code is just for show

```
libraries-override:
    # Replace an entire library
    core/drupal.collapse: mytheme/collapsed

    # Replace an asset with another.
    subtheme/library:
        css:
            theme:
                css/layout.css: css/my-layout.css

    # Remove an asset.
    drupal/dialog:
        css:
            theme:
                dialog.theme.css: false

    # Remove an entire library.
    core/modernizr: false
```

Libraries Extend Example

```
# Extend drupal.user: add assets from classy's user libraries.
libraries-extend:
        core/drupal.user:
            - classy/user1
            - classy/user2
```

Questions you may have...

- Why is the word `theme` in the library definitiion?

Done ☺