The Asset Pipeline

CSS, JS, images and other media in Rails.

CSS, JavaScript and images

Aside from the Ruby code for your routes and controllers and the HTML code in your views, a huge part of any Web application is **CSS**, **JS** and **images**.

CSS, JavaScript and images

Just like everything else, Rails has a place specifically for those kinds of files:

the `app/assets/` folder.

Structure of `app/assets/`

A folder for each type of file.

```
app/assets/
 ├── images/
 ├── javascripts/
 └── stylesheets/
```

Structure of `app/assets/`

Default contents:

```
app/assets/
├── images/
│   └── (empty)
├── javascripts/
│   └── application.js
└── stylesheets/
    └── application.css
```

Structure of `app/assets/`

CSS and JavaScript files for each controller:

```
app/assets/
├── images/
│   └── (empty)
├── javascripts/
│   ├── application.js
│   └── site.coffee
└── stylesheets/
    ├── application.css
    └── site.scss
```

All CSS and JS files in `app/assets/` are included **automatically** in **every page**.

CSS & JS on the Asset Pipeline

So you can write your CSS in any CSS file and your JavaScript in any JavaScript file.

CSS & JS on the Asset Pipeline

Since you can put it in any file, you have to decide how you want to organize your CSS and JavaScript.

CSS & JS on the Asset Pipeline

Generally speaking, CSS that affect the pages of a specific controller should go in that controller’s CSS file.

CSS & JS on the Asset Pipeline

For example, CSS for the pages on `SiteController` should go in the `site.scss`.

BONUS

SCSS files = Sass

Why do those CSS files end with `.scss`?

That's because they are really Sass files.

SCSS files = Sass

BONUS

Sass is like CSS but better. It’s got a lot of cool features that CSS doesn’t have.

Many front end developers use it.

BONUS

SCSS files = Sass

But browsers don’t understand Sass, they only understand CSS.

Rails automatically translates Sass to CSS.

BONUS

SCSS files = Sass

Even though they are Sass files you can put regular old CSS in there.

What about this `.coffee` file?

Images on the Asset Pipeline

As for images, you just drop them in the `app/assets/images/` folder.

Download this GIF into your project (here’s the link).

Name it something like `dramatic-glasses.gif`.

Now visit localhost:3000/assets/dramatic-glasses.gif.

Images you place inside `app/assets/images/` will be available on

[`localhost:3000/assets/NAME`](localhost:3000/assets/NAME).

Images on the Asset Pipeline

To use these images in your HTML:

```
<img src="/assets/dramatic-glasses.gif"
    alt="Kid removing his sunglasses epically">
```

app/views/site/home.html.erb

Images on the Asset Pipeline

The Rails way to do it is:

```erb
<%= image_tag "dramatic-glasses.gif" %>
```

app/views/site/home.html.erb

BONUS

The `image_tag` helper

The `image_tag` helper has a few benefits over using a plain `<img>` tag.

The `image_tag` helper

BONUS

It prevents browsers from saving old versions of the image (if you update it).

Browsers are able to **cache** images to prevent unnecessary downloads.

The `image_tag` helper

BONUS

For big projects in *production*, sometimes you store all your images on a different server.

The helper can adjust automatically to that.

The `image_tag` helper

BONUS

For example, Amazon doesn’t keep it’s images on `amazon.com`.

Images you put inside `app/assets/images/` shouldn’t be images uploaded by users.

Image uploads

Uploaded images should be handled in a different way.

You'll learn more about that while working on your project (if needed).

Other people's assets

BONUS

Finally, it's a good practice to place 3rd party assets in the `vendor/assets/` folder.

`app/assets/` is for your app's custom stuff.

IRON HACK

BONUS

Other people's assets

For popular libraries like Bootstrap it might be easier to include them using a gem.

See the `twitter-bootstrap-rails` gem.

RailsGuides

Always read the guide! There's a great one on the Asset Pipeline.

guides.rubyonrails.org/asset_pipeline.html