ConFoo.CA
WEB TECHNO CONFERENCE

Rails Asset Pipeline Production Techniques

Leonard Teo, Ballistiq
@leonardteo
leonard@ballistiq.com

www.ballistiq.com
www.leonardteo.com

ballistiq»

Why should I care about the Asset Pipeline?

- Use preprocessors easily: SCSS/SASS, Less, Coffeescript, Slim, Haml
- Performance improvements. Automatic concatenation and compression of asset files. Minification and Gzip where appropriate.
- Sanity saving during development. All javascripts and CSS are laid out nicely.
- Cache busting.
- Easy reuse and library integrations using Rubygems. E.g. Bootstrap gem

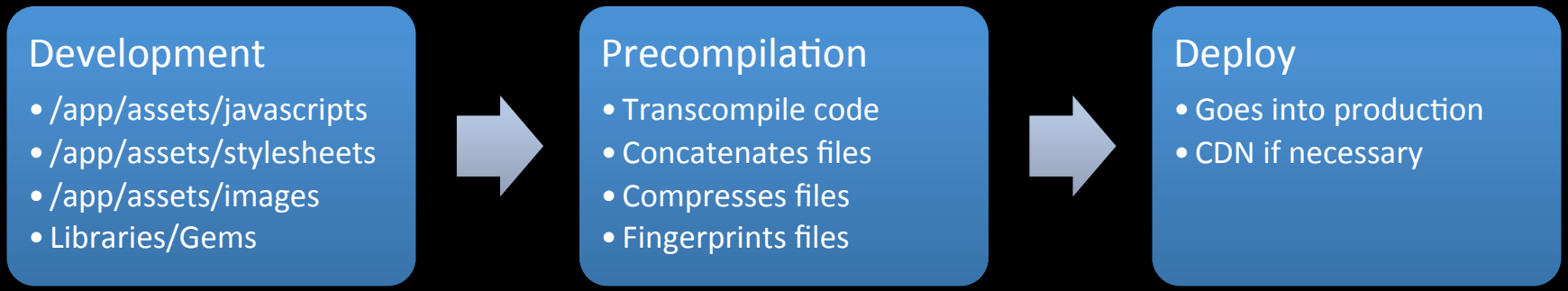
What's the problem?

- It's complex until you 'get it'. Unfortunately it's more complex than it needs to be.
- Newcomers to Rails are weirded out by it.
- Front-end designers/devs who aren't L33T just freak out with it (JS+Coffescript+SCSS+LESS in the same project!?!?!?!?).
- Hardcore backend devs don't appreciate it as much as they should.

What it's for / not for

- Use it for:
  - Anything UI/UX related
  - UI images, icons
  - Javascripts
  - CSS
- Don't use it for:
  - Large stuff
    - Videos
  - User content
    - Avatars
    - Images and files uploaded by users

The Asset Pipeline



Rails Asset Pipeline

IN DEVELOPMENT

Asset Pipeline Bare Basics

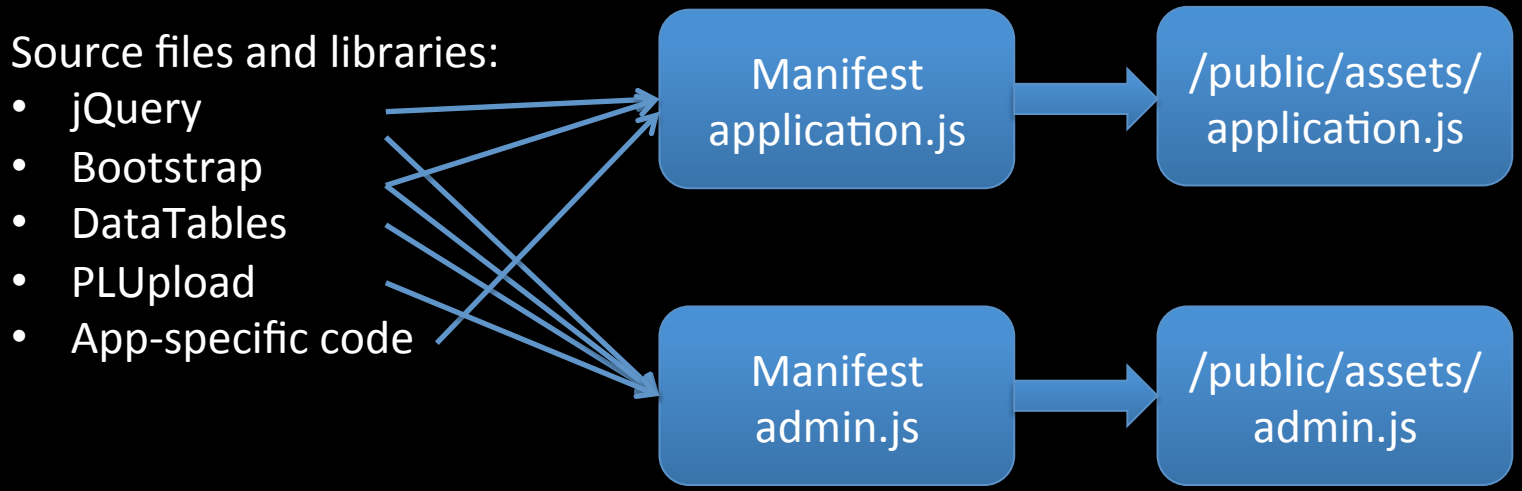
- When you start a Rails project, the asset pipeline is enabled by default.
- The module is called Sprockets (gem). Sprockets == Rails asset pipeline.
- Assets are in /app/assets/
  - /app/assets/javascripts
  - /app/assets/stylesheets
  - /app/assets/images

Demo

- Rails new project
- Scaffold some controllers
- Show each individual JS/CSS file in development
- Precompile assets
- Show how JS and CSS are now a single file
- Show SCSS quickly
- Show Coffeescript quickly

Manifests

A manifest is a single CSS or JS file that will take all the assets that you define in it and compile it into a single asset file.



Manifest Gotcha

- When you create a new manifest, you MUST declare it in production.rb.
- If not, it will NOT precompile and you WILL get a 500 server error when you deploy.

Preprocessors/Transcompilation

- CSS:
  - SCSS (default)
  - SASS
  - Less
  - ERB
- JS:
  - Coffeescript
  - ERB
- You are not forced to use these. You can use vanilla CSS and JS. Different preprocessors and templating engines can coexist.

Manually Integrating a Library Bootstrap

- Demo:
  - Show how to manually integrate a 3rd party library (Bootstrap):
  - Copy files to respective directories in:
    - /vendor/assets/javascripts
    - /vendor/assets/stylesheets
    - /vendor/assets/images
  - Change paths of images in CSS to /assets/
  - Show how to use <%= asset_path %> in CSS with ERB extension

The FAST way - Rubygems

- Rails enables you to store assets as part of its plugin/gem architecture.
- There are MANY gems available for common libraries such as:
  - Bootstrap
  - Zurb
  - Datatables
  - Etc. Just google it and you'll likely find it
  - Be careful that some Gems are NOT maintained. Check what version of the library the Gem is at.

Rubygems Bootstrap Demo

- Gem “twitter-bootstrap-rails”
- Rails generate bootstrap:install static
- Rails g bootstrap:layout application fixed
- Rails g bootstrap:themed [Resource name]

Controller/Action specific javascripts

- [Demo]
- Issue: Not all javascript should be run on specific controllers. Some libraries don't check if an element exists before executing and will throw an error.
- Options:
  - Check if the element exists using $('#..').length (I don't like this)
  - Methods for checking controller/action/namespace:
    1. Pass the controller and action into the layout in the <body> tag as data attributes. You can also pass the namespace if you wish.
    2. Global functions to check what the controller/action/namespace is.
    3. If… blocks to only execute the JS if the user is in the correct namespace/controller/action.

Rails Asset Pipeline

PRECOMPILATION

Precompilation

- Precompilation command:
  - bundle exec rake assets:precompile
- Creates /public/assets directory
- To undo, just `rm –rf /public/assets` or `bundle exec rake assets:clean`
- In development, when you precompile assets, it creates a directory on /public which is served (unless you configure it to serve from a different path). This means that any CSS/Javascripts are run TWICE.

Rails Asset Pipeline

DEPLOYMENT OPTIONS

1. Precompile assets on server on deployment (this is bad)

- In Capistrano, you uncomment `load ‘deploy/assets’` in Capfile.
- Basically it runs the assets precompilation on the server when you deploy.
- It’s VERY computationally intensive. Can knock out a micro EC2 instance.
- Assets might not be changed. Why precompile again?
- It can take down your site while it is precompiling (Passenger)
- You have to install all dependencies on the server – therubyracer, libv8. When you deploy a new rails version sometimes it upgrades these and it can add 30+ minutes to your deploy time.

2. Precompile on server only if you have to (also bad)

- There is a capistrano recipe to only precompile if there are actually changes to the files.
- http://stackoverflow.com/questions/9016002/speed-up-assetsprecompile-with-rails-3-1-3-2-capistrano-deployment

- Can still bog the server down as you are compiling server-side.

3. Precompile assets locally and rsync them to the server

- Capistrano recipe will precompile the assets locally then upload them:
- http://keighl.com/post/fast-rails-assets-precompile-capistrano

- Pretty good solution if you control asset precompilation manually

4. Deploy on another branch

- Create a ‘production’ branch of your code.
- When deploying, merge master into production, precompile assets, commit and push, then `cap deploy`.
- Fastest deploy time as all compilation is done locally. It just has to check out the files and restart the app.
- Don’t need rubyracer/libv8 on server. `bundle install –without test assets development`
- Can be painful as you now have an extra branch. Be careful not to commit stuff to the production branch, then merge production into master and have /public/assets/ in your master branch.
- I use this method because it means that I’m in full control of precompilation and what goes into production.

5. Configuration

- Deploy from Master
- Change development.rb
  - config.assets.prefix = "/dev-assets”
- Change application.rb
  - config.assets.initialize_on_precompile = false
- Enables you to develop without the conflict of /public/assets
- Precompile assets and commit code as needed.

CDN to serve assets

- Using Cloudfront, set up a new distribution and configure it to get assets from your app. If your app is serving at “www.leonardteo.com” make that the origin.
- In production.rb ,set: config.action_controller.asset_host = “http://xxx.cloudfront.net”
- See this guide: http://blog.codeship.io/2012/05/18/Assets-Sprites-CDN.html

Rails Asset Pipeline

REUSE

Creating your own reusable library

- [Demo]
- Create a rails plugin using:
  - `rails plugin new [name]`
- In your plugin root, create the folder structures to store your assets:
  - ./vendor/assets/javascripts/[name]/
  - ./vendor/assets/images/[name]/
  - ./vendor/assets/stylesheets/[name]/
- Use manifest files for stylesheets and javascripts.
- Remember to add the Engine class that inherits from Rails::Engine!
- Always namespace so that you don't conflict with your app stylesheets.
- In your application, add the gem and configure it to read from github repository.
- In your application manifest files, you can now require the stylesheets and javascripts.

Rails Asset Pipeline

QUESTIONS?

If we have time…

BONUS FRONT END STUFF

Slim templating

- Take a good look at Slim (slim-lang.com) as an alternative to ERB.
- Very light syntax.
- Html2Slim to convert existing ERB templates.

```
doctype html
html
  head
    title Slim Examples
    meta name="keywords" content="template language"
    meta name="author" content=author
    javascript:
      alert('Slim supports embedded javascript!')

  body
    h1 Markup examples

    #content
      p This example shows you how a basic Slim file looks like.

      == yield

      - unless items.empty?
        table
          - for item in items do
            tr
              td.name = item.name
              td.price = item.price
      - else
        p
          | No items found.  Please add some inventory.
            Thank you!

    div id="footer"
      = render 'footer'
      | Copyright © #{year} #{author}
```

Static Websites - Middleman

- If you need to create a static website or mini-site (e.g. marketing site for your app), don't roll a Rails app! Use Middleman!
- middlemanapp.com
- Take advantage of asset pipeline, preprocessors, layouts, ERB/slim/whatever.
- Single command-line build will compile everything into a static HTML site (even with .php files if you need) for deployment.

We're hiring!

- Looking for a front-end designer ninja
- Email us!
  - hr@ballistiq.com

Credits / Contact

- Thanks to the Rails team for an awesome framework.
- To the dedicated people on Stack Overflow answering questions on the Asset Pipeline.
- Peer reviewers: Marc-Andre Lafortune, Martin Provencher.

Please give feedback!
https://joind.in/7970

Contact:

leonard@ballistiq.com
@leonardteo