Ruby on Rails Importmaps Project Structure

Native ES modules with Importmaps. No bundler, no node_modules, just browser imports.

#rails #ruby #importmaps #javascript #minimal

PNG PDF Copy Prompt

Project Directory

```
myapp/
    Gemfile
    Gemfile.lock
    Rakefile
    config.ru
  app/
    controllers/
      application_controller.rb
      posts_controller.rb
    models/
    views/
      layouts/
        application.html.erb javascript_impo…
      posts/
    javascript/ ES modules
      application.js Entry point
      controllers/ Stimulus contro…
        index.js
        application.js
        hello_controller.js
      lib/ Custom JS modul…
        utils.js
    assets/
    helpers/
  config/
    application.rb
    routes.rb
    database.yml
    importmap.rb Pin JS dependen…
    environments/
    initializers/
  vendor/
    javascript/ Vendored JS libs
      .keep
  db/
  test/
  lib/
  public/
  bin/
```

Why This Structure?

Importmaps leverage native browser ES module support. No Webpack, no esbuild, no node_modules. Pin dependencies in `importmap.rb`, browser fetches from CDN or vendored files. Simpler toolchain, faster development.

Key Directories

`config/importmap.rb` - Pin dependencies to URLs or local files

`app/javascript/` - ES modules, auto-imported via pins

`vendor/javascript/` - Vendored JS for offline/versioning

`app/javascript/controllers/` - Stimulus controllers with importmap

Importmap Configuration

```
# config/importmap.rb
pin "application", preload: true
pin "@hotwired/turbo-rails", to: "turbo.min.js", preload:
pin "@hotwired/stimulus", to: "stimulus.min.js", preload:
pin_all_from "app/javascript/controllers", under: "control

# Add external lib
pin "lodash", to: "https://ga.jspm.io/npm:lodash@4.17.21/l
```

Getting Started

1. `rails new myapp` (Importmaps is default)
2. `bin/importmap pin lodash` (add dependency)
3. Import in `app/javascript/application.js`
4. `rails server`
5. Check Network tab—no bundled JS file

When To Use This

- Want simplest possible JS setup
- Hate configuring Webpack/esbuild
- Mainly using Stimulus and Turbo
- Lightweight JS requirements
- Team unfamiliar with Node tooling

Trade-offs

`No bundling` - Many HTTP requests for large dependency trees

`No transpilation` - JSX, TypeScript not supported natively

`CDN dependency` - jspm.io or self-vendor for production

`Limited ecosystem` - Some npm packages need bundler

Best Practices

- Vendor critical dependencies for production
- Use `preload: true` for essential imports
- Keep JavaScript minimal—embrace Hotwire
- Use `pin_all_from` for Stimulus controllers
- Check browser support for import maps