// FIELD GUIDE

v 1.0 · Apr 2026

Ruby Refresher

A field guide for returning Rubyists, built for the rusty.

FOR Steve Meisner

EVENT Blue Ridge Ruby · Asheville

PREP Two evenings, one ambition

```
$ ruby --version
ruby 3.4.0 (2024-12-25)
```

> "matz is nice and so we are nice"

— compiled with love

CONTENTS · 14 SECTIONS

What's inside

> Skim it tonight, dive deeper Friday morning, keep it open in a browser tab between talks. The Time Machine and Cheat Sheet are the highest-leverage sections — read those even if you read nothing else.

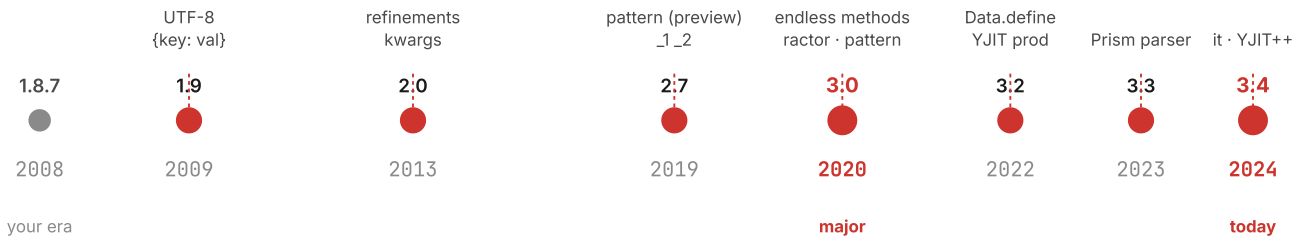
HOW TO READ THIS

Code blocks are runnable. Try them in `irb` as you read — muscle memory beats skimming. Anything in `red` means "this is new since you last touched Ruby seriously."

SECTION 01

01 Time Machine

> You started in the Ruby 1.8 / Rails 2 era. The world is now Ruby 3.4 and Rails 8. Here's the highlight reel of what landed while you were doing other things.



The five shifts that matter most

1. **Hash syntax flipped.** `{ :name => "Steve" }` is now `{ name: "Steve" }`. The rocket is legacy code smell, not your default.
2. **Keyword arguments became real.** Method signatures look totally different — `def greet(name:, greeting: "hi")` with *required* kwargs is the modern norm.
3. **Pattern matching landed (3.0).** `case/in` destructures hashes and arrays with type and shape constraints. It's the biggest language addition since blocks.
4. **YJIT made Ruby fast.** Free 15–40% speedup on real Rails apps with one flag. Shopify runs it at planet scale.
5. **Concurrency story matured.** Threads still exist. **Fibers** + Async became viable for I/O. **Ractors** exist for true parallelism (still rough).

> **BACKWARDS COMPAT HEADS-UP**
>
> Old code in tutorials uses `:foo => bar`, `attr_accessor` over `Data.define`, and bare `def foo(opts={})` instead of kwargs. It still *works* — but if you write it today, eyes will roll.

SECTION 02

02 Modern Syntax Cheat Sheet

> Old vs new, side by side. If you can read every block on these pages without flinching, you're 80% of the way to "looks fluent in a code review."

HASH LITERALS

```
# Then
user = { :name => "Steve", :role => :admin }

# Now (2.0+)
user = { name: "Steve", role: :admin }

# Now-now (3.1+) — shorthand pulls from local var
name = "Steve"
role = :admin
user = { name:, role: }
```

METHOD ARGUMENTS

```
# Then — positional with options hash
def greet(name, opts = {})
  greeting = opts[:greeting] || "hi"
  "#{greeting}, #{name}"
end

# Now — required and optional keyword args
def greet(name:, greeting: "hi")
  "#{greeting}, #{name}"
end

# Anonymous forwarding (3.0+ for blocks, 3.2+ for everything)
def log(...)
  puts caller.first
  inner(...)
end
```

SAFE NAVIGATION AND ONE-LINERS

```ruby
user&.address&.city            # nil if any link is nil

JSON.parse(str) rescue {}      # one-line rescue

def square(x) = x * x          # endless method (3.0+)

return :ok if all_good?        # trailing conditional, still cool
```

BLOCK PARAMETER SHORTHAND

```ruby
%w[hi there].map(&:upcase)     # symbol-to-proc, ages old

[1,2,3].map { _1 * 2 }         # numbered (2.7+)
[1,2,3].map { it * 2 }         # named (3.4+) — preferred now
```

> **STYLE NOTE**
>
> The Ruby community has settled: `it` is the future, `_1` still works, named params (`|x|`) win when the variable name adds clarity.

Value objects without the boilerplate

```ruby
# Then — Struct, mutable by default
Money = Struct.new(:amount, :currency) do
  def +(other) = Money.new(amount + other.amount, currency)
end

# Now — Data is immutable, kwarg-constructed
Money = Data.define(:amount, :currency) do
  def +(other) = with(amount: amount + other.amount)
end

m = Money.new(amount: 10, currency: "USD")
m.amount        # 10
m.amount = 20   # NoMethodError — Data is frozen
m2 = m.with(amount: 99)  # creates a new one
```

Pattern matching, the headline feature

```ruby
case api_response
in { status: 200, body: { items: [first, *rest] } }
  process(first, rest)
in { status: (400..499) => code, body: { error: String => msg } }
  raise ClientError.new(code, msg)
in { status: 500.. }
  retry_later
end

# Rightward assignment — pull a value out of a hash
{ name: "Steve", role: :admin } => { name:, role: }
puts name   # "Steve"

# In-line check — returns true/false
{ a: 1, b: 2 } in { a: Integer }   # => true
```

Iteration helpers you should know

```
"hello world".chars.tally
# => {"h"=>1, "e"=>1, "l"=>3, "o"=>2, " "=>1, "w"=>1, "r"=>1, "d"=>1}

[1, 2, 3, 4].sum { |n| n * n }      # 30

%w[a b c d].each_cons(2).to_a       # => [["a","b"], ["b","c"], ["c","d"]]
%w[a b c d].each_slice(2).to_a      # => [["a","b"], ["c","d"]]

# Lazy for huge or infinite sequences
(1..Float::INFINITY).lazy
  .map  { it ** 2 }
  .select { it.even? }
  .first(5)
# => [4, 16, 36, 64, 100]

# Hash with default block — the "group by" pattern
groups = Hash.new { |h, k| h[k] = [] }
words.each { groups[it.length] << it }
```

Tap and then for fluent chains

```
User.new(params)
  .tap(&:save)                      # side effect, returns self
  .then { Mailer.welcome(it) }      # transform, returns new value
  .deliver_later
```

SECTION 03

03 Pattern Matching, Deep

> The single feature most likely to confuse you mid-talk. Worth 20 minutes of focused study because once it clicks, it changes how you write Ruby.

The shape of it

Three forms exist. Memorize them all.

```ruby
# 1. case/in — full branching
case value
in Integer => n if n > 0
  "positive int #{n}"
in [String, *]
  "array starting with a string"
in { status: 200 }
  "ok"
else
  "no match"
end

# 2. Rightward assignment — assert and destructure (raises NoMatchingPatternError)
{ name: "Steve", age: 47 } => { name:, age: }

# 3. Boolean test — returns true/false
{ status: 200 } in { status: Integer }   # => true
```

Patterns you can match

Pattern	Example	What it means
Literal	`in 200`	Exact value
Class	`in String`	`===` against the class
Range	`in 200..299`	Includes value
Array	`in [1, 2, *rest]`	Shape match, captures rest
Hash	`in { status:, body: }`	Has these keys, binds them
Find	`in [*, Integer => n, *]`	Has at least one Integer
Capture	`in Integer => n`	Bind matched value to `n`
Pin	`in ^expected`	Compare to outer var (don't bind)
Alternative	`in 1 | 2 | 3`	Any of these
Guard	`in Integer => n if n.even?`	Pattern AND condition

Custom deconstruction

Any object can opt into pattern matching by defining `deconstruct` (for arrays) or `deconstruct_keys` (for hashes).

```
class Point
  attr_reader :x, :y
  def initialize(x, y) = (@x, @y = x, y)
  def deconstruct       = [x, y]
  def deconstruct_keys(keys) = { x:, y: }
end

p = Point.new(3, 4)

case p
in [0, 0]            then "origin"
in { x: 0 }          then "on y-axis"
in { x:, y: } if x == y then "diagonal"
in [Integer, Integer] then "anywhere else"
end
```

WHEN TO REACH FOR IT

Pattern matching shines when you're *shaping* data: API responses, JSON parsing, AST walking, message dispatch. For simple `if/else` you don't need it. For deeply nested hash extraction, it's a superpower.

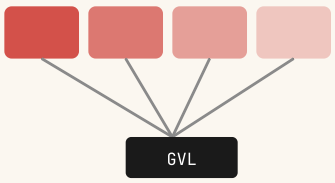
SECTION 04

04 Concurrency Models

> Ruby has *three* primitives for doing more than one thing at once. They're not interchangeable. Knowing which to reach for is a marker of fluency.

Thread

os-managed · GIL-bound



Use for

I/O-bound work — DB queries, HTTP calls

Watch out

CPU work won't go in parallel — GVL holds

Fiber

cooperative · scheduler-driven

single thread, many fibers

Use for

10K+ concurrent I/O connections (Async)

Watch out

Blocking calls block all fibers in thread

Ractor

isolated · truly parallel

message-passing only

Use for

CPU-bound parallel work that can be sharded

Watch out

Still experimental. Most gems aren't safe.

The pragmatic guidance

For 99% of Rails code, you'll never write a thread, fiber, or ractor by hand — Puma uses threads, Falcon uses fibers, Sidekiq has a worker pool. **You just need to know which model your tools use, so you don't fight them.**

```ruby
# Async — the modern way to do concurrent I/O
require 'async'
require 'async/http/internet'

Async do
  internet = Async::HTTP::Internet.new
  urls = %w[https://a.com https://b.com https://c.com]

  results = urls.map do |url|
    Async { internet.get(url).read }   # fires off concurrently
  end.map(&:wait)
end
```

SECTION 05

05 YJIT & The Performance Era

> If a 2014-era Rubyist time-traveled to today, the most shocking thing wouldn't be the syntax — it'd be how *fast* Ruby got.

What YJIT is

YJIT (Yet-another-just-in-time-compiler) is a JIT compiler written in Rust, built into Ruby itself since 3.1. It watches your code as it runs and compiles hot paths to native machine code. It was developed at Shopify and runs in production on every Shopify storefront.

Workload	Typical YJIT speedup	Notes
Rails app, real traffic	15–40%	Shopify reports ~15% latency drop
CPU-heavy benchmarks	2–4×	Optcarrot, railsbench
Tiny scripts	often slower	Compile cost not amortized

Turning it on

```
# CLI flag
ruby --yjit script.rb

# Env var
RUBY_YJIT_ENABLE=1 bundle exec rails server

# In code
RubyVM::YJIT.enable

# In Rails (config/environments/production.rb)
config.yjit = true   # default in Rails 7.2+
```

Other speed wins worth knowing

FROZEN STRING LITERALS

```
# frozen_string_literal: true   # magic comment, top of file

# Every string literal is frozen — no allocation on repeat use
def status_label = "active"   # one String object, forever
```

BOOTSNAP

Shopify gem (bundled with Rails) that caches compiled code, autoload paths, and YAML. Cuts Rails boot time by 50%+ on cold starts.

FAST JSON, FAST EVERYTHING

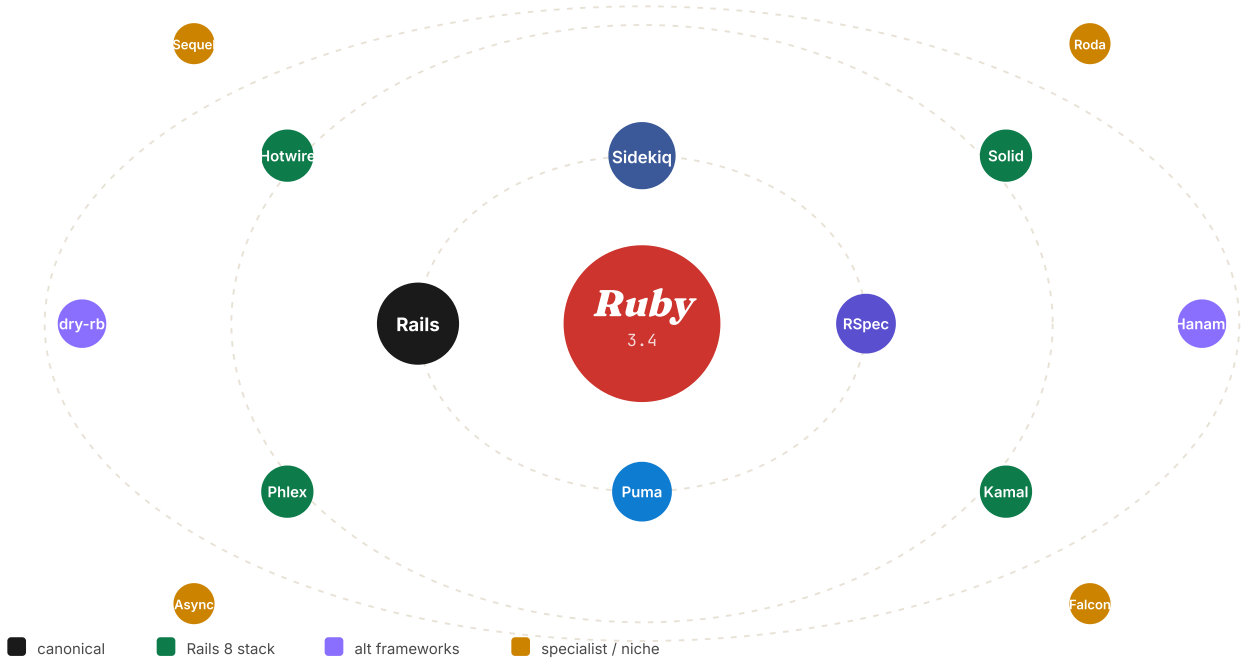
`oj` (Optimized JSON) is 4–8× faster than stdlib JSON. `brotli` for compression. `nokogiri` for XML/HTML. The `Ankane` gems are reliably fast.

> **CONF-FLOOR TALKING POINT**
> "Are you on YJIT?" is a perfectly normal question to ask anyone running Rails in production today. If they say no, the follow-up is "any blocker?" — usually it's just "haven't gotten to it."

SECTION 06

06 The Library Galaxy

> A field map of gems by orbit: the inner planets you'll see in every app, the outer-belt specialists, and the comets — weird and wonderful.



The canon — every Rails dev knows these

Gem	What it does	Notes
Rails	Full-stack web framework	v8 is current; "no PaaS required" defaults
Sidekiq	Background jobs (Redis)	Mike Perham; Pro/Ent funds the open core
Solid Queue	Background jobs (Postgres/SQLite)	Rails 8 default — eating Sidekiq's lunch
Devise	Authentication	Old guard, still everywhere
Rodauth	Authentication	Modern, secure, gaining ground over Devise
Pundit	Authorization (policies)	Tiny, idiomatic
RSpec	Testing DSL	Most common in apps; `describe/context/it`
Minitest	Testing (Rails default)	Faster, simpler, fewer abstractions
FactoryBot	Test data factories	Was `factory_girl`
Rubocop	Linter / formatter	Often paired with `standard` for style
Puma	App server (threaded)	Default in Rails
Nokogiri	HTML/XML parsing	Foundational; ships with Rails
Pagy	Pagination	Replaced Kaminari for many; very fast
Bullet	N+1 query detection	Dev-only must-have

The "modern Rails" power-ups

Gem	What it does
Hotwire	Turbo + Stimulus — HTML-over-the-wire, "no SPA needed"
ViewComponent	GitHub's component framework — server-rendered Ruby objects
Phlex	HTML *in* Ruby. `def view_template; h1 { "Hi" }; end`
Lookbook	Visual component browser for ViewComponent / Phlex
Kamal	DHH's deploy tool — Docker + SSH, no Kubernetes
Propshaft	New asset pipeline — Sprockets is on the way out
importmap-rails	Ship JS without a bundler. Rails 7+ default.
Litestack	SQLite-as-everything (queue, cache, pub/sub) — buzzy

The dry-rb & functional Ruby orbit

A whole parallel ecosystem favoring small, composable objects over Rails-y conventions. Common at Stripe, Shopify, and shops that came out of the Trailblazer school.

Gem	What it does
dry-validation	Schema validation — declarative, composable
dry-monads	`Success` / `Failure`, `Maybe`, do-notation
dry-types	Type system for Ruby objects
ROM (Ruby Object Mapper)	Alt to ActiveRecord — repository pattern
Sequel	The *other* great ORM. Often better than AR for complex SQL.
Roda	Routing-tree microframework. Beautifully designed.
Hanami	Full alternative to Rails — dry-rb-flavored, slice-based

Specialists, niche gems, and curiosities

Async

`github.com/socketry/async · Samuel Williams`

Structured concurrency for Ruby. The future of high-concurrency I/O. Powers Falcon webserver.

Falcon

`github.com/socketry/falcon`

Async-based webserver. Tens of thousands of concurrent connections per process.

Polars-rb

`github.com/ankane/polars-ruby`

DataFrames in Ruby, Rust-backed. Very fast.

pgvector-ruby

`github.com/pgvector/pgvector-ruby`

Vector embeddings in Postgres — the "I have an LLM in my Rails app" gem.

langchainrb

`github.com/patterns-ai-core/langchainrb`

LLM agents and RAG in Ruby. Small but real ecosystem.

Sorbet

`sorbet.org · Stripe`

Gradual type checker. Used at Shopify, Stripe, GitHub. `RBS` + `Steep` is the official alt.

Avo

`avohq.io`

Modern admin panel framework. Beautiful out of the box.

GoodJob

`github.com/bensheldon/good_job`

Postgres-backed background jobs. Pre-cursor to Solid Queue, still excellent.

Bridgetown

`bridgetownrb.com`

Modern Jekyll. Static sites, Ruby-flavored.

StimulusReflex

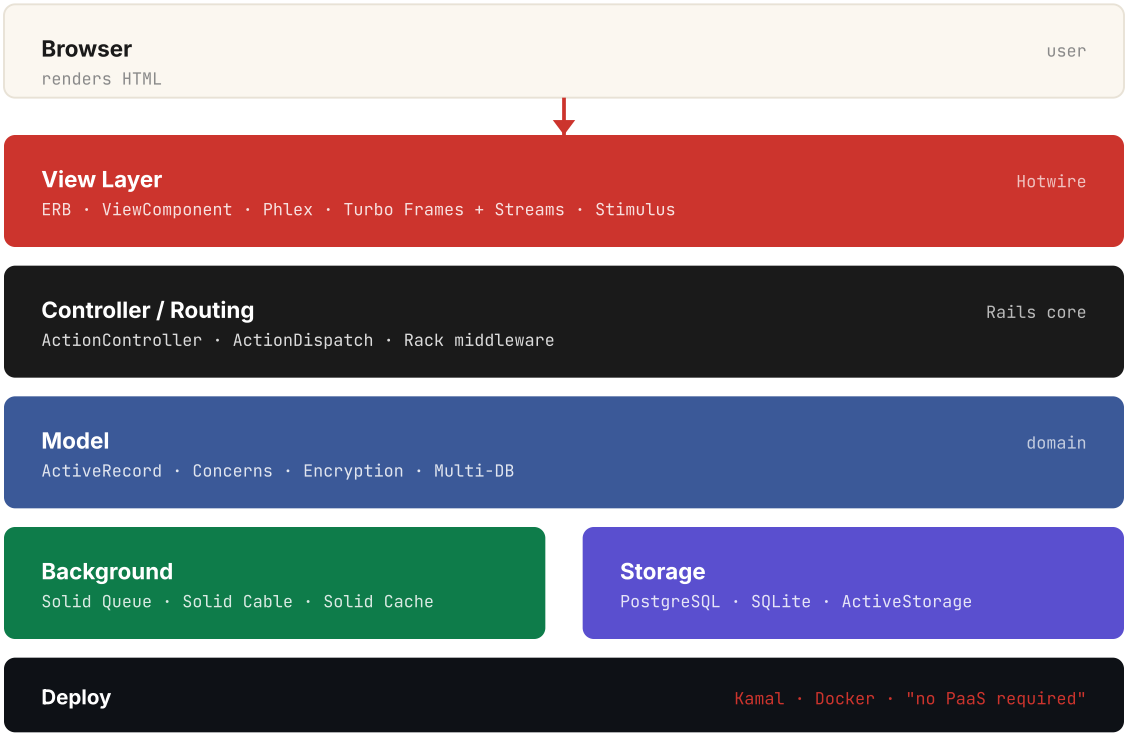
`stimulusreflex.com`

Reactive Rails before Hotwire. Niche but loved.

SECTION 07

07 The Modern Rails Stack

> Rails 8 is opinionated about the whole stack now — front to back, deploy included. Here's what a "fresh `rails new`" gives you in 2026.



Hotwire decoded

Piece	What it actually is
Turbo Drive	Intercepts links/forms, swaps the `<body>` without a full reload. Like Rails' old Turbolinks but better.
Turbo Frames	Lazy-load & replace a chunk of HTML. `<turbo-frame id="cart">` updates independently.
Turbo Streams	Server pushes HTML over WebSocket / SSE. `<turbo-stream action="append" target="messages">`
Stimulus	Tiny JS framework. Controllers attach via `data-controller="..."`. Sprinkles, not SPA.
Native	iOS/Android adapters that wrap your Hotwire app in a thin native shell. (How HEY ships.)

The Solid trio

Rails 8 ships **Solid Queue**, **Solid Cache**, and **Solid Cable** — Postgres-or-SQLite-backed implementations of background jobs, caching, and websockets. The narrative: *you don't need Redis to run a Rails app anymore*. (You can still use Redis. Many do.)

> **WHY THIS MATTERS**
> The Solid trio + Kamal + SQLite is DHH's "you can self-host on a single VPS" pitch. It's a real shift — small Rails apps no longer need Heroku-style PaaS or 5 services. Big ones still benefit from Redis & managed infra.

SECTION 08

08 Phlex, ViewComponent, ERB

> The view layer is where Ruby has the most active aesthetic debate right now. You'll hear all three names. Here's what each one looks like.

ERB — THE CLASSIC

```erb
<!-- app/views/users/show.html.erb -->
<h1><%= @user.name %></h1>
<ul>
  <% @user.posts.each do |post| %>
    <li><%= link_to post.title, post %></li>
  <% end %>
</ul>
```

VIEWCOMPONENT — SERVER-RENDERED COMPONENTS

```ruby
# app/components/user_card_component.rb
class UserCardComponent < ViewComponent::Base
  def initialize(user:)
    @user = user
  end

  def admin? = @user.role == :admin
end

# app/components/user_card_component.html.erb
<div class="card <%= 'admin' if admin? %>">
  <h3><%= @user.name %></h3>
</div>

# Use it
<%= render UserCardComponent.new(user: @user) %>
```

PHLEX — HTML IN PURE RUBY

```ruby
class UserCard < Phlex::HTML
  def initialize(user:)
    @user = user
  end

  def view_template
    div(class: card_class) do
      h3 { @user.name }
      p { "Joined #{@user.created_at.to_date}" }
      a(href: user_path(@user)) { "View profile →" }
    end
  end

  private

  def card_class
    @user.admin? ? "card admin" : "card"
  end
end

# In a controller / view
render UserCard.new(user: @user)
```

Choosing between them

	Best for	Watch out
ERB	Most apps, most pages. The default for a reason.	Logic-in-template creep over time.
ViewComponent	Reusable UI in a design system. Big team apps.	Boilerplate per component (two files).
Phlex	Devs who think in Ruby and dislike template languages. Highly composable UI.	Smaller community. ERB tutorials don't translate.

CONF WEATHER REPORT

Phlex is hot — talks about it appear at every Ruby conf in 2025–26. ViewComponent is mature and dominant in big shops. ERB is still 90%+ of all Rails templates. None of these are going away.

SECTION 09

09 Open Source to Read

> Reading great Ruby is the fastest way to absorb modern style. Here's a curated tour, ordered easiest-to-hardest.

Read for code quality & idioms

Pagy `small`

`github.com/ddnexus/pagy`

~1k lines. Read the whole thing. Masterclass in keeping a gem tiny while doing one thing well.

Sidekiq `medium`

`github.com/sidekiq/sidekiq`

Mike Perham. Production-grade Ruby. Notice the lack of meta-magic and the comments.

Sequel `deep`

`github.com/jeremyevans/sequel`

Jeremy Evans. A masterclass in Ruby library design. Plugins-as-modules pattern.

Roda `deep`

`github.com/jeremyevans/roda`

Routing-tree microframework. Every line considered. Read alongside Rails for contrast.

dry-rb gems `style`

`github.com/dry-rb`

Exemplars of small-object, functional-flavored Ruby. Different aesthetic from Rails.

Phlex `small`

`github.com/phlex-ruby/phlex`

Joel Drapper's HTML-in-Ruby. See how a modern DSL is built today.

Read to see real apps

Lobsters `beginner`

`github.com/lobsters/lobsters`

Tiny HN-style site. Small enough to grok in an afternoon. Classic Rails patterns.

Discourse `huge`

`github.com/discourse/discourse`

Forum software. Modern Rails, ambitious frontend, heavily commented. Sam Saffron writes great Ruby.

Mastodon `huge`

`github.com/mastodon/mastodon`

Federated social. Rails + ActivityPub. Real-world distributed systems in Ruby.

Forem `huge`

`github.com/forem/forem`

The dev.to engine. Classic Rails monolith, well-tended.

Chatwoot `medium`

`github.com/chatwoot/chatwoot`

Customer support platform. Good showcase of WebSockets / ActionCable in production.

Solidus `deep`

`github.com/solidusio/solidus`

E-commerce. Complex domain modeling — pricing, inventory, fulfillment.

HOW TO READ CODE WELL

Don't read top-to-bottom. Pick a single feature ("how does Lobsters do voting?"), find its controller, then trace down: model → service object → tests. Forty minutes of one feature beats four hours of skimming.

SECTION 10

10 People to Follow

> The Ruby community is small, friendly, and remarkably accessible. These are the names you'll hear referenced in talks and hallway chats.

Core & legends

Name	Known for	Where
Yukihiro Matsumoto (Matz)	Created Ruby. Still active, still nice.	@yukihiro_matz
DHH (David Heinemeier Hansson)	Created Rails. 37signals. Opinionated.	@dhh · world.hey.com/dhh
Aaron Patterson (tenderlove)	Rails core. Performance. Hilarious.	@tenderlove · tenderlovemaking.com
Eileen Uchitelle	Rails core. Multi-database. GitHub.	@eileencodes
Xavier Noria (fxn)	Zeitwerk autoloader. Deep Ruby internals.	@fxn
Koichi Sasada (ko1)	YARV, Ractor, GC.	atdot.net/~ko1

Library authors worth following

Name	Built
Mike Perham	Sidekiq · mikeperham.com
Jeremy Evans	Sequel, Roda, Rodauth
Andrew Kane (ankane)	Searchkick, PgHero, Lockbox, dozens of gems
Samuel Williams (ioquatix)	Async, Falcon — the future of Ruby I/O
Sam Saffron	Discourse co-founder. Performance writer.
Joel Drapper	Phlex creator
Stephen Margheim	SQLite-on-Rails evangelist
Bozhidar Batsov (bbatsov)	Rubocop, The Ruby Style Guide
Noah Gibbs	"Rebuilding Rails" book, benchmarking

Newsletters & podcasts

- **Ruby Weekly** — Peter Cooper. The canonical Ruby newsletter. Subscribe.

- **Short Ruby News** — Lucian Ghinda. Excellent and visual.
- **Remote Ruby** — Chris Oliver, Jason Charnes, Andrew Mason. Friendly weekly podcast.
- **The Bike Shed** — thoughtbot. Long-running, deep on practice.
- **Ruby on Rails Podcast** — currently hosted by Brittany Martin.

SECTION 11

11 Conference Survival Kit

> Ruby people are some of the friendliest in tech. The community has a strong "Matz is nice and so we are nice" (MINASWAN) ethos. Lean into it.

Hallway track tactics

- **"I started in 1.8 and I'm rusty"** is a great icebreaker. Old-timers will light up; younger devs will be curious about the journey.
- **Ask "what does X buy you over Y?"** — works for any tool comparison. Signals you're thinking, not lost.
- **Find the conference Slack/Discord before day one.** Most regional confs have one. Hallway track really starts there.
- **Lunch is the best track.** Don't sit alone. Sit with strangers. Ruby people are unusually open to "mind if I join you?"
- **Remember names by the gem they wrote.** "Oh you're the Pagy person!" makes someone's day.

Conversation starters that work

TRY THESE

- "What's something you've shipped in Ruby recently you're proud of?"
- "What's the wildest gem you've used in production?"
- "How are you feeling about Rails 8's *no Redis* story?"
- "Are you running YJIT yet?"
- "Phlex, ViewComponent, or just ERB?"
- "What's the last technical talk that genuinely changed how you write code?"

If a talk goes over your head

1. Don't panic. Half the room is faking it too.
2. Write down the *vocabulary*, not the content. Look it up after.
3. Find the speaker after — speakers *love* "I didn't fully follow X but it sounded important; can you point me at a starter resource?"
4. Check the talk's slides/repo on the conference website afterward — most Ruby speakers post code/slides publicly.

The vocabulary survival list

If someone uses one of these and you blank, here's the 6-second mental cache:

If they say...	You think...
"We Kamal'd it"	Deployed via DHH's Docker+SSH tool
"Just throw it on a Solid Queue job"	Background job, Postgres-backed, Rails 8 default
"It's a Phlex component"	HTML written in Ruby methods, no template file
"We're on YJIT"	Ruby's JIT compiler, free perf
"That's a Turbo Stream"	Server pushing HTML over WebSocket/SSE
"Use a Sorbet sig"	Stripe's gradual type checker
"It's a Ractor"	Isolated parallel actor (Ruby 3+)
"Bundler issue"	Gemfile dependency problem
"Zeitwerk's complaining"	Rails autoloader; usually file/class name mismatch

SECTION 12

12 Two-Evening Study Plan

> Designed for tonight + Friday morning before the conf opens. ~2 hours total. Don't aim for fluency — aim for *recognition*.

Tonight (~90 min)

1. **Read sections 01–03 of this guide** (Time Machine, Cheat Sheet, Pattern Matching). ~25 min
2. **Open `irb` and type every code block from Section 02.** Don't paste — type. Muscle memory beats reading. ~30 min
3. **Solve one pattern-matching exercise.** Open a scratch file:

```
# Given an API response shape like:
#   { status: 200|400|500, body: { ... } | { error: "..." } }
# write a method that returns:
#   :ok | :client_error | :server_error
# using ONE case/in statement.
```

   ~15 min
4. **Skim sections 06–08 (libraries, Rails stack, view layer wars).** Don't memorize — just absorb the shape. ~20 min

Friday morning (~30 min)

1. **Re-skim Section 11 (Conference Survival Kit)** on the way to the venue. ~10 min
2. **Read one Hotwire example** — the Turbo handbook intro on hotwired.dev. ~10 min
3. **Pick the talk that scares you most** from the schedule. Read its abstract *carefully* and Google any unfamiliar word. ~10 min

During the conf

- Keep this guide open in a browser tab — Section 13 (Glossary) is your panic button.
- Note unfamiliar terms during talks; look them up at lunch.
- One real conversation > ten passive talks. Prioritize the hallway.

"You don't need to understand everything. You need to recognize enough to ask the next question."

SECTION 13

13 Glossary

> Every term you might hear, with a one-sentence definition. Bookmark this page; refer often.

A – H

ActionCable — Rails' WebSocket framework.

ActionMailer — Rails' email framework.

ActionText — Rich-text editing in Rails (uses Trix).

ActiveJob — Rails' background-job abstraction; backends include Sidekiq, Solid Queue, GoodJob.

ActiveRecord — Rails' ORM.

ActiveSupport — Ruby stdlib extensions Rails ships (e.g. `blank?`, `presence`).

Async — Samuel Williams' concurrency gem; structured fibers.

Bootsnap — Caches compiled Ruby + autoload paths for fast boot.

Bundler — Dependency manager. `Gemfile` + `Gemfile.lock`.

Concern — Module mixin pattern; `extend ActiveSupport::Concern`.

Crystal — Compiled language with Ruby-like syntax. Not Ruby.

Data.define — Immutable value object class (Ruby 3.2+).

DSL — Domain-specific language. Ruby's superpower (RSpec, routes).

dry-rb — A suite of small functional gems (validation, monads, types).

Endless method — `def foo = bar` (Ruby 3.0+).

I – Z

importmap — Ship JS without a bundler. Rails 7+ default.

Kamal — DHH's Docker-based deploy tool.

kwargs — Keyword arguments. `def f(name:)`.

Minitest — Rails' default test framework.

mruby — Embedded Ruby (IoT, robotics).

Pattern matching — `case/in` destructuring (Ruby 3+).

Phlex — HTML in Ruby instead of ERB.

Prism — New Ruby parser, written in C, default in 3.4.

Propshaft — New asset pipeline replacing Sprockets.

Puma — Default Rails app server, threaded.

Rack — Foundational web server interface (every Ruby web framework is Rack-based).

Ractor — Actor-based parallel execution unit (Ruby 3+, experimental).

RBS — Official Ruby type signature format.

Roda — Routing-tree microframework (Jeremy Evans).

Sidekiq — Redis-backed background job system.

Solid Cache / Cable / Queue — Rails 8's Postgres/SQLite-backed cache, websockets, jobs.

Falcon — Async-based webserver.

Fiber — Cooperative lightweight thread.

Frozen string literals — Magic comment that freezes all string literals. Faster.

Gemfile — Bundler dependency declaration.

GVL — Global VM Lock. One thread runs Ruby code at a time.

Hanami — Alternative web framework, dry-rb-flavored.

Hotwire — Turbo + Stimulus, the "no SPA" Rails frontend.

Sorbet — Stripe's gradual type checker.

Sprockets — Old asset pipeline, being replaced by Propshaft.

Stimulus — Tiny JS sprinkles framework (part of Hotwire).

Turbo Drive — Intercepts navigation, swaps body without reload.

Turbo Frame — Independently-updatable HTML chunk.

Turbo Stream — Server-pushed HTML over WebSocket/SSE.

ViewComponent — GitHub's component framework.

YARV — Yet Another Ruby VM. The bytecode interpreter.

YJIT — Ruby's JIT compiler. Written in Rust.

Zeitwerk — Rails' autoloader. Filename → class name.

SECTION 14 · TEAR-OUT

14 Quick Reference Card

> The minimum-viable cheat sheet. Print this page if nothing else; it covers ~90% of what you'll see in a modern Rails codebase.

HASH & KWARGS

```ruby
{ name: "x", role: :admin }
{ name:, role: }              # 3.1+
def f(name:, age: 18); end
def f(...); inner(...); end
```

BLOCKS

```ruby
arr.map(&:upcase)
arr.map { it * 2 }            # 3.4+
arr.map { _1 * 2 }            # 2.7+
arr.each_with_object({}) { |x, h| h[x] = x.size }
```

SAFE NAV & RESCUE

```ruby
user&.address&.city
JSON.parse(s) rescue {}
def square(x) = x * x
```

PATTERN MATCHING

```ruby
case x
in Integer => n if n > 0
in [a, *rest]
in { status: 200, body: }
end

x in { ok: true }             # boolean
{a:1} => { a: }               # destructure
```

DATA & STRUCT

```ruby
Money = Data.define(:amt, :cur)
m = Money.new(amt: 5, cur: "USD")
m.with(amt: 10)               # new copy
```

ENUMERABLE HITS

```ruby
arr.tally
arr.sum { it * 2 }
arr.each_cons(2)
arr.each_slice(3)
arr.partition(&:even?)
hash.transform_values { it * 2 }
hash.filter_map { |k, v| k if v > 0 }
```

LAZY

```ruby
(1..).lazy.select(&:prime?).first(10)
```

CHAINS

```ruby
obj.tap(&:save)               # side effect
   .then { wrap(it) }         # transform
```

MODERN RAILS COMMAND FLAGS

```ruby
rails new app --css=tailwind \
              --javascript=importmap \
              --database=sqlite3
bundle exec rails db:migrate
RUBY_YJIT_ENABLE=1 bundle exec puma
```

TOP 10 CONF VOCABULARY

Hotwire · Turbo Frame/Stream · Stimulus · Solid Queue · Solid Cache · Solid Cable · Kamal · Phlex · ViewComponent · YJIT

END OF GUIDE

Have a great conference, Steve. Bring back gossip.