Open Source Data Visualization Tools: Selection and Workflow Design

- Choose the right open source tool for a defined task, then design a repeatable workflow
- Audience: analysts, data journalists, educators, developers, and small data teams
- Open source means zero licence fees but real operational cost
- Six decision axes: data size, interactivity, channel, reproducibility, skills, maintenance
- Expect a two-tool reality: a charting library plus a self-hosted platform

What Open Source Actually Buys You

PERMISSIVE (MIT, Apache-2.0)

Focus on flexibility and adoption with minimal obligations.

Cost of ownership (self-hosted)

- Infrastructure
- Operations
- Maintenance
- Integration
- Total effort

COPYLEFT (AGPL-3.0)

Protects freedom of use and ensures improvements stay open.

Cost of ownership (self-hosted)

- Infrastructure
- Operations
- Maintenance
- Integration
- Total effort

SOURCE-AVAILABLE

Source is available for review, but use is governed by specific terms.

Cost of ownership (self-hosted)

- Infrastructure
- Operations
- Maintenance
- Integration
- Total effort

OPEN-CORE

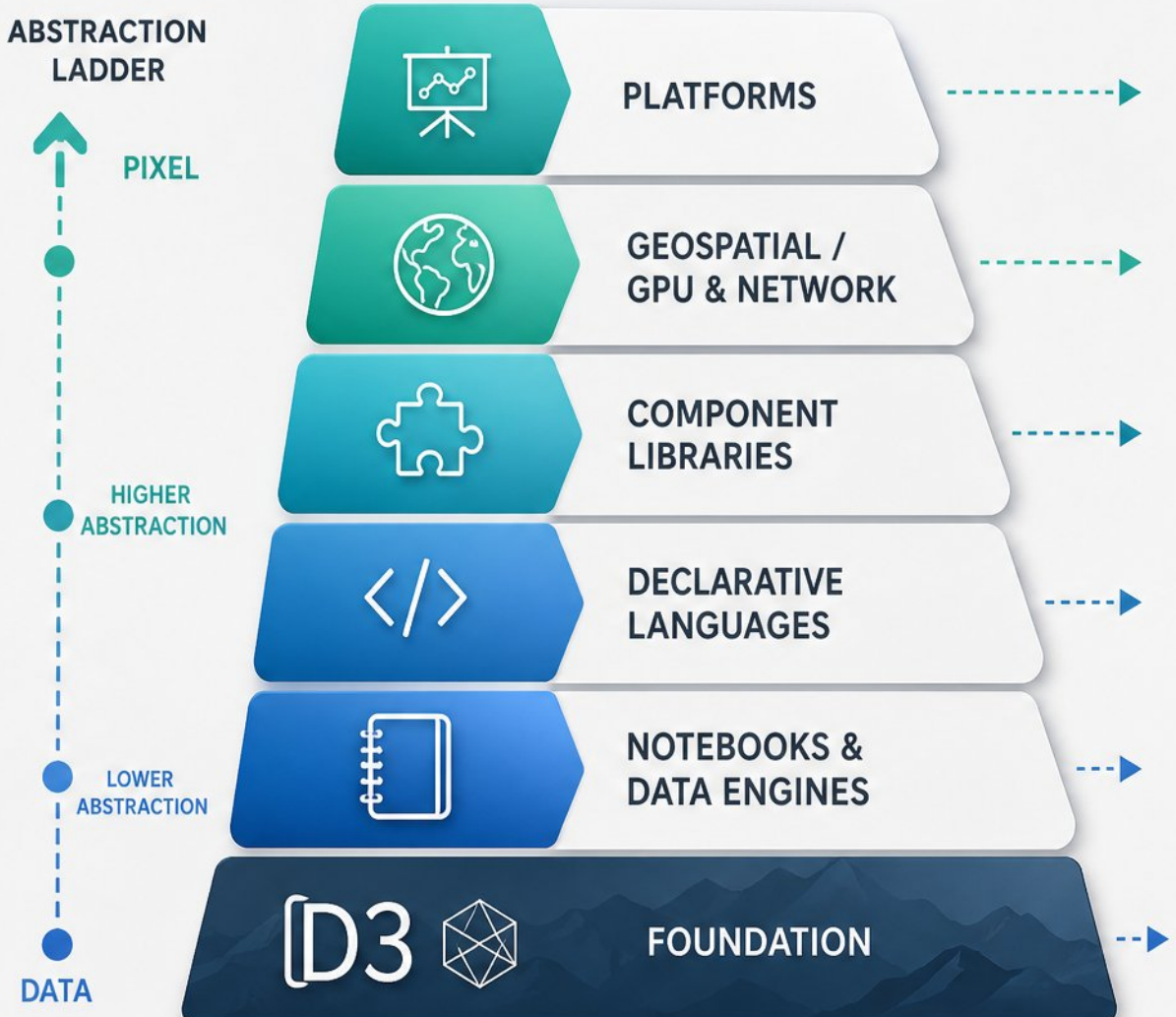
Core is open; advanced features or support offered commercially.

Cost of ownership (self-hosted)

- Infrastructure
- Operations
- Maintenance
- Integration
- Total effort

- Licence families: permissive (MIT, Apache-2.0), copyleft (AGPL-3.0), source-available, open-core
- Superset: Apache-2.0, no restricted directory; Grafana core AGPL-3.0; Metabase AGPL-3.0 plus commercial binary
- AGPL embedding: modified network-served code must be published; MIT and Apache-2.0 carry no such duty
- Zero licence cost is not zero cost: Superset needs web server, Postgres, Redis, Celery, websocket service
- Small-team self-hosting: 4-8 hrs/month Metabase or Grafana; 16-30 hrs/month production Superset
- Managed 2026 anchors: Metabase Cloud ~$85/mo (5 users), Preset ~$20/user/mo, Grafana Cloud free tier

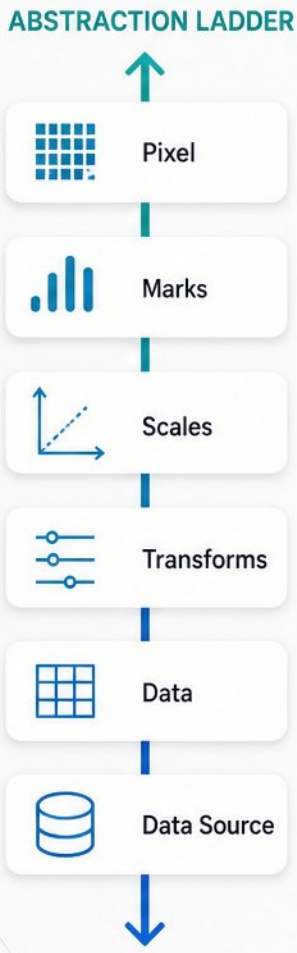
The 2026 Tool Landscape in One Page

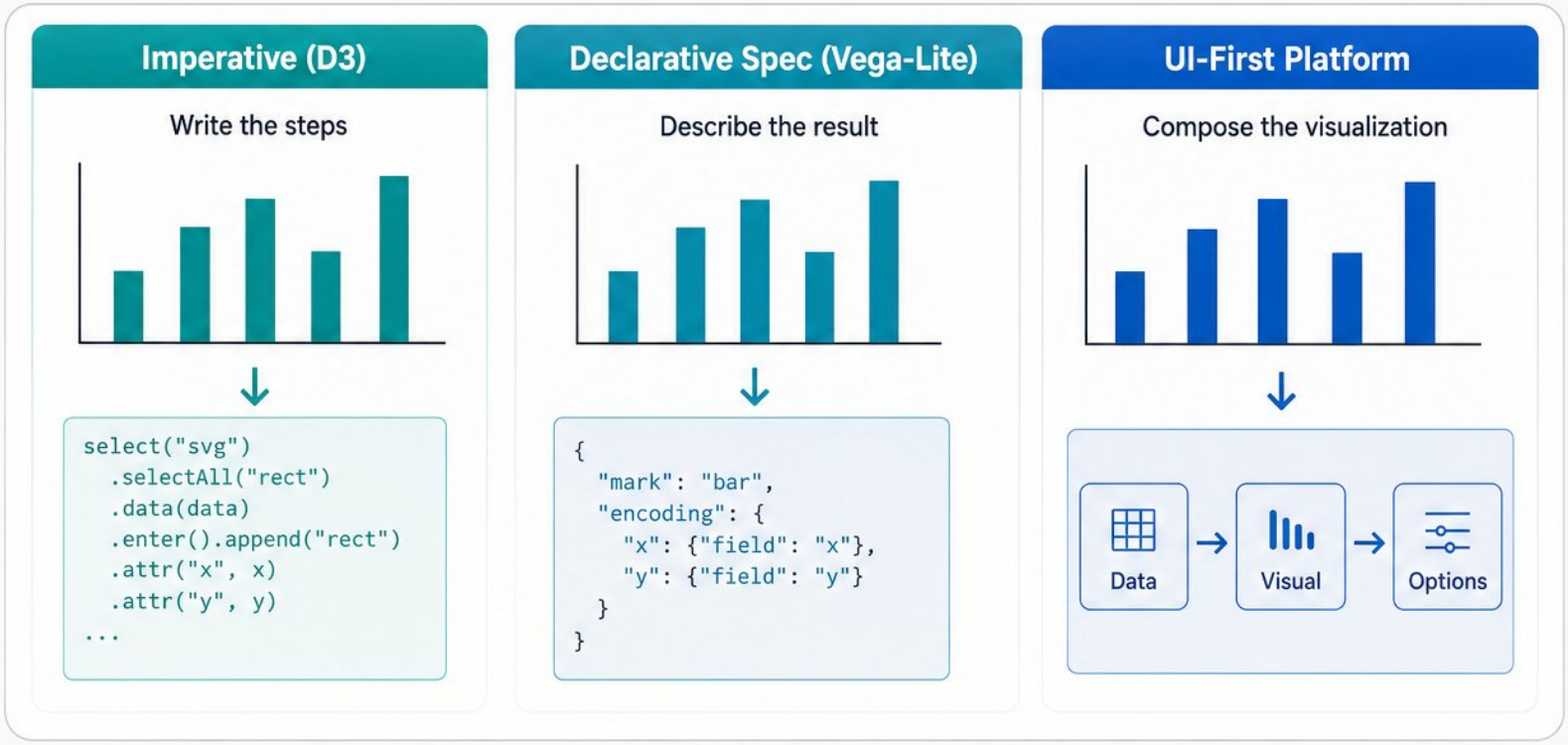


- Platforms: Superset 6.x, Metabase, Grafana, Lightdash, Evidence
- Geospatial/GPU: QGIS, Leaflet, deck.gl 9, regl 2; network: Gephi, Cytoscape
- **Components:** Recharts 2.15, ECharts 6 (Nov 2025, dual Canvas/SVG), Chart.js 4.5, Nivo, AntV
- **Declarative layer:** Vega-Lite 5.20, Observable Plot 0.7, ggplot2 4.x, Plotly.js 2.35
- **Notebooks:** Jupyter, Quarto 1.5+, Marimo, Observable Framework; Polars, DuckDB underneath
- **D3.js v7:** the base beneath most abstractions, ~8M weekly npm downloads, ISC licence

– **Abstraction ladder: if a gallery chart fits, start a rung or two above D3; reach for WebGL last**

Reading the Landscape: The Comparison Dimensions That Matter



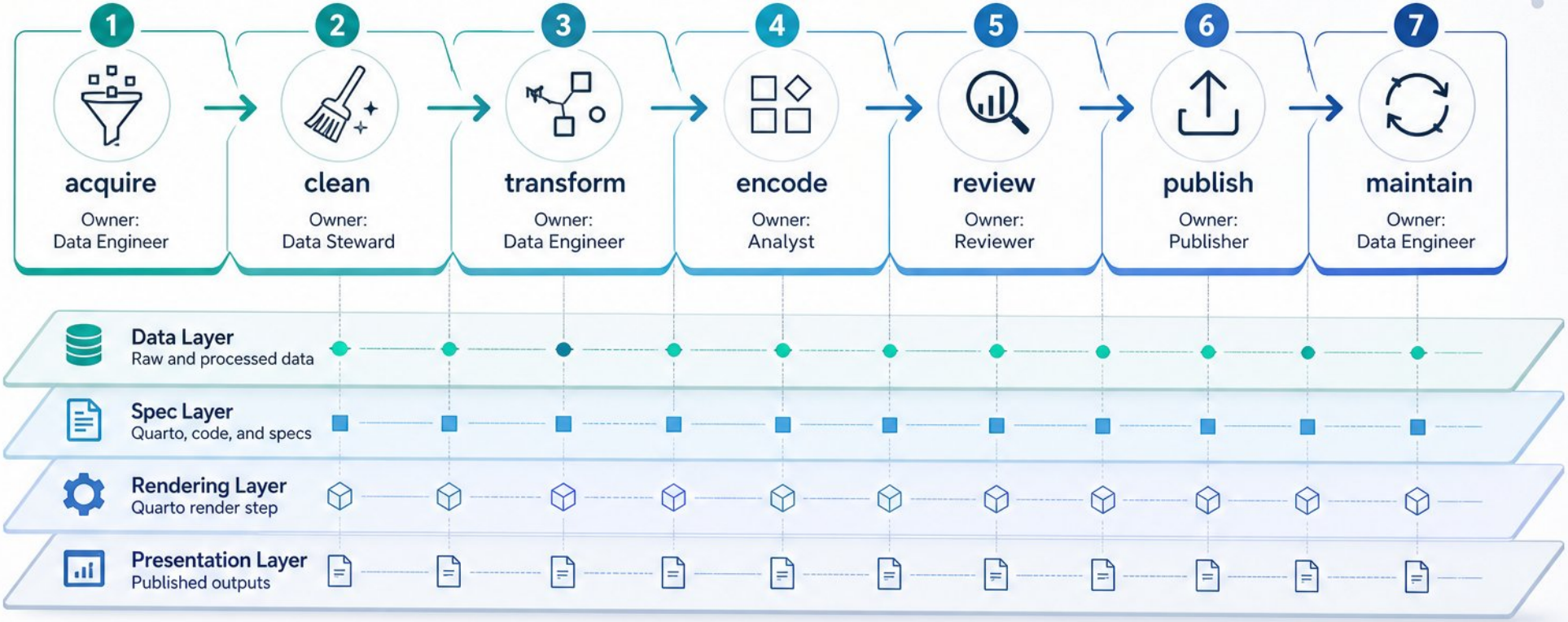


- **Programming model:** imperative D3 render loop vs declarative specs vs UI-first platforms
- **Control vs velocity:** bar chart in 60-150 D3 lines, 12-25 Vega-Lite JSON lines
- **Bundle cost:** D3 modular pay-as-you-go; Vega-Lite ships ~280-320 KB gzipped
- **Interactivity:** Vega-Lite gives a fixed menu free; D3 makes any interaction possible
- **Scale:** D3 wins at large mark counts; beyond 50k marks, aggregate or go GPU
- **Verify maturity yourself:** release cadence, contributor concentration, docs, backing, advisories
- **Audience mapping:** journalists to Plot/Quarto, analysts to Superset, developers to D3

Selection Criteria That Actually Matter

CRITERION	WHAT THIS MEANS	EVIDENCE SLOT
Task fit first	• Task fit first: exploratory, explanatory, monitoring, and teaching each reward different tools	
Scale and shape	• Scale and shape: row count, geospatial layers, streaming versus batch, update frequency	
Output channel	• Output channel: static image, web embed, dashboard, print, and slides impose different constraints	
Score skill honestly	• Score skill honestly: JavaScript, R or Python, SQL-only users, available designer time	
TCO includes	• TCO includes hosting, connectors, upgrades, patching, and training	
Evidence-backed scorecard	• Evidence-backed scorecard: must-have, nice-to-have, disqualifying criteria	

Workflow Design: From Raw Data to Published Visual



- Seven stages: acquire, clean, transform, encode, review, publish, maintain
- Separate data, spec, rendering, and presentation layers
- R: Quarto + targets, tar_load() in the report, tar_quarto() in the pipeline
- Python: uv lockfile, Quarto freeze: auto, Marimo .py notebooks for clean PR diffs
- Commit sources, treat PDF/HTML/PNG as build artifacts
- Keep experiments in a playground; promote only what survives review

Tool Path Deep Dive 1: Libraries and Grammar-Based Frameworks

Declarative (Grammar Path)
Vega-Lite / Altair Spec

```
{
 "data": {"name": "table"},

 "mark": {"type": "bar"},

 "encoding": {
   "x": {"field": "category",
         "type": "nominal"},
   "y": {"field": "value",
         "type": "quantitative"},
   "color": {"field": "group",
           "type": "nominal"}
 }
}
```

Concise, declarative spec
Describes **WHAT** to show
Framework handles **HOW**

Spec-to-D3 handoff for signature visual

Imperative (Library Path)
D3.js Rendering Code

```
// set up canvas and scales
const svg = d3.select('svg')
  .attr('width', width)
  .attr('height', height);

const x = d3.scaleBand()
  .domain(data.map(d => d.category))
  .range([0, width)])
  .padding(0.1);

const y = d3.scaleLinear()
  .domain([0, d3.max(data, d => d.value)])
  .nice()
  .range([height, 0]);

// draw bars
svg.selectAll('.bar')
  .data(data)
  .enter().append('rect')
  .attr('class', 'bar')
  .attr('x', d => x(d.category))
  .attr('y', d => y(d.value))
  .attr('width', x.bandwidth())
  .attr('height', d => height - y(d.value))
  .attr('fill', d => color(d.group));

// axes, labels, tooltips, interactions...
```

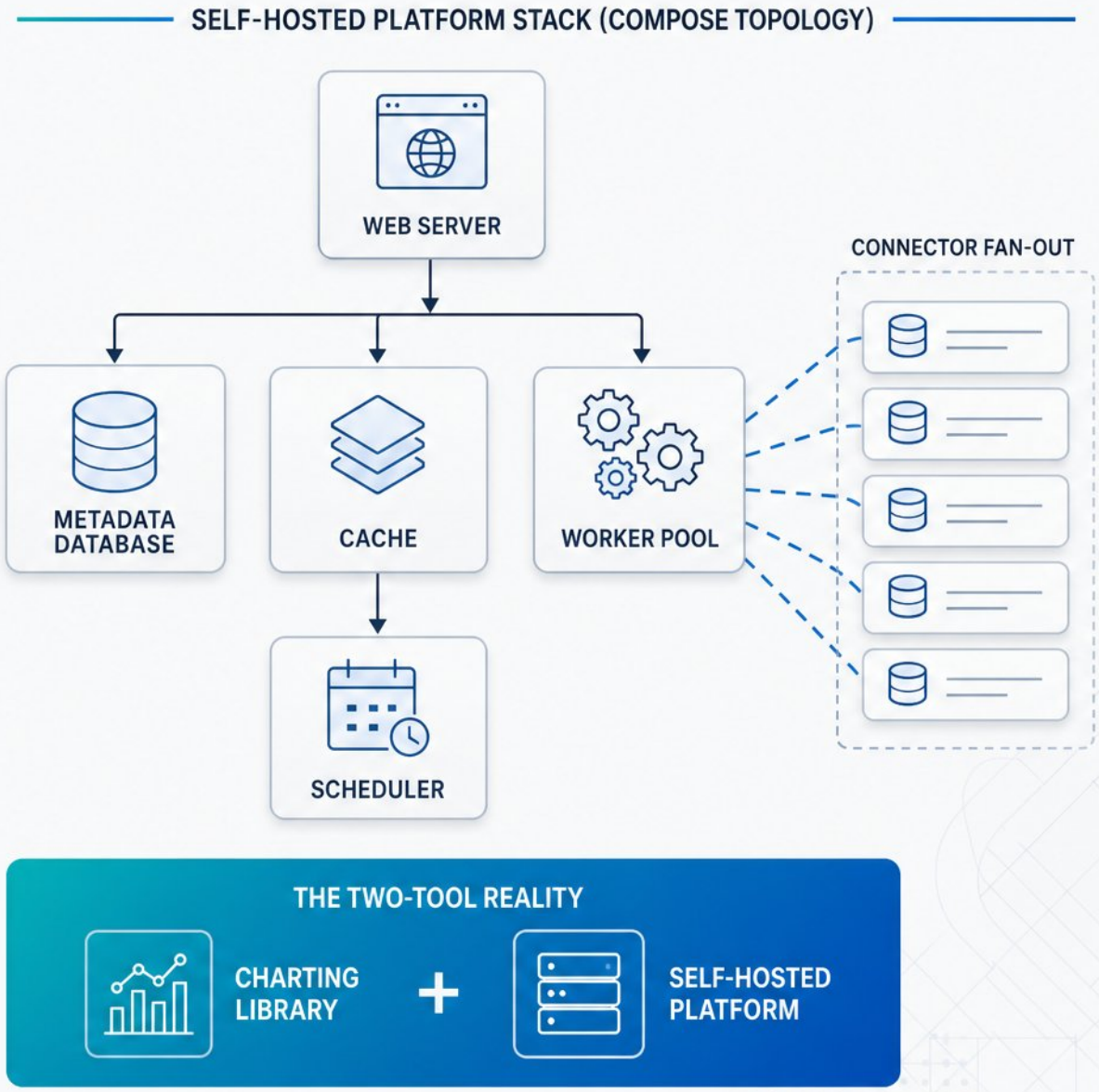
Maximum control, bespoke logic
Defines **HOW** to draw
More code, developer-leaning

Abstraction Ladder: From Data to Pixel

- Library path (D3, Plotly.js, Matplotlib, deck.gl): maximum control, more code, developer-leaning
- Grammar path (Vega-Lite, Observable Plot, ggplot2, Altair): concise declarative specs for standard charts
- Altair compiles a tidy DataFrame into a portable Vega-Lite spec
- Strongest pattern: deliberate coexistence — Vega-Lite default, D3 for signature visuals
- Rendering engine is an accessibility choice: SVG free, Canvas opaque, WebGL needs aggregates
- Decision shortcut: volume of standard charts to grammar; bespoke design to libraries

Tool Path Deep Dive 2: Choosing and Running a Platform

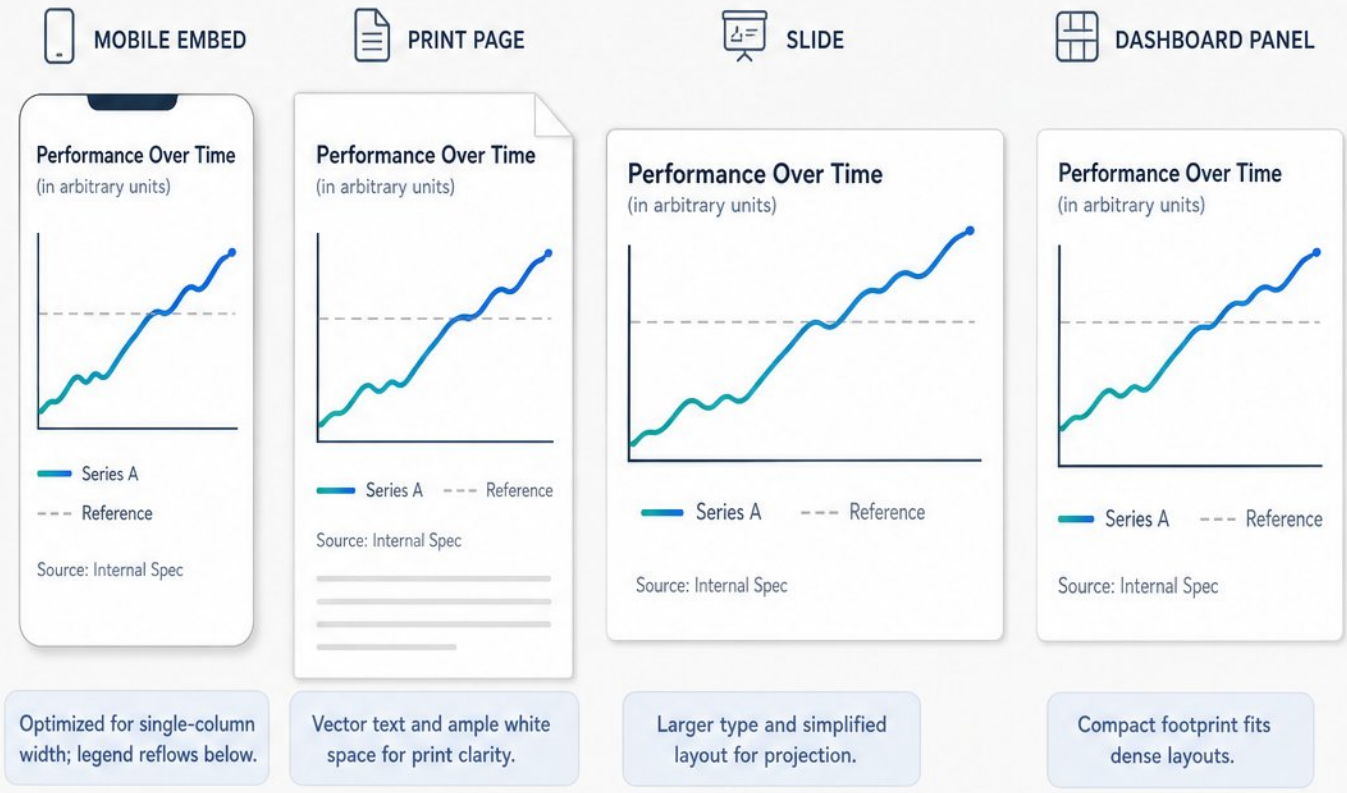
- Superset for SQL-heavy teams; Metabase for non-technical self-service; Grafana for time series
- Connector breadth varies: Superset 80+, Metabase 20+, Lightdash nine warehouses only
- Setup: Metabase ~10 minutes; Superset 30-60 minutes across seven services
- Superset scales furthest but has the slowest, most fragile upgrade path
- Superset embeds free under Apache-2.0; Metabase Pro gates embedding near $500/month
- Avoid lock-in: check dashboards-as-code before committing



Designing for the Audience and the Medium

- Name the decision the visual supports before choosing a chart
- Match chart type to data relationship; avoid dual and truncated axes
- Medium drives constraints: mobile legend reflow, print vector text, slide type size
- Annotation lives in the spec: takeaway titles, units, reference lines, callouts
- Centralize style tokens; publish a one-page chart contract per team

One Chart, Four Mediums

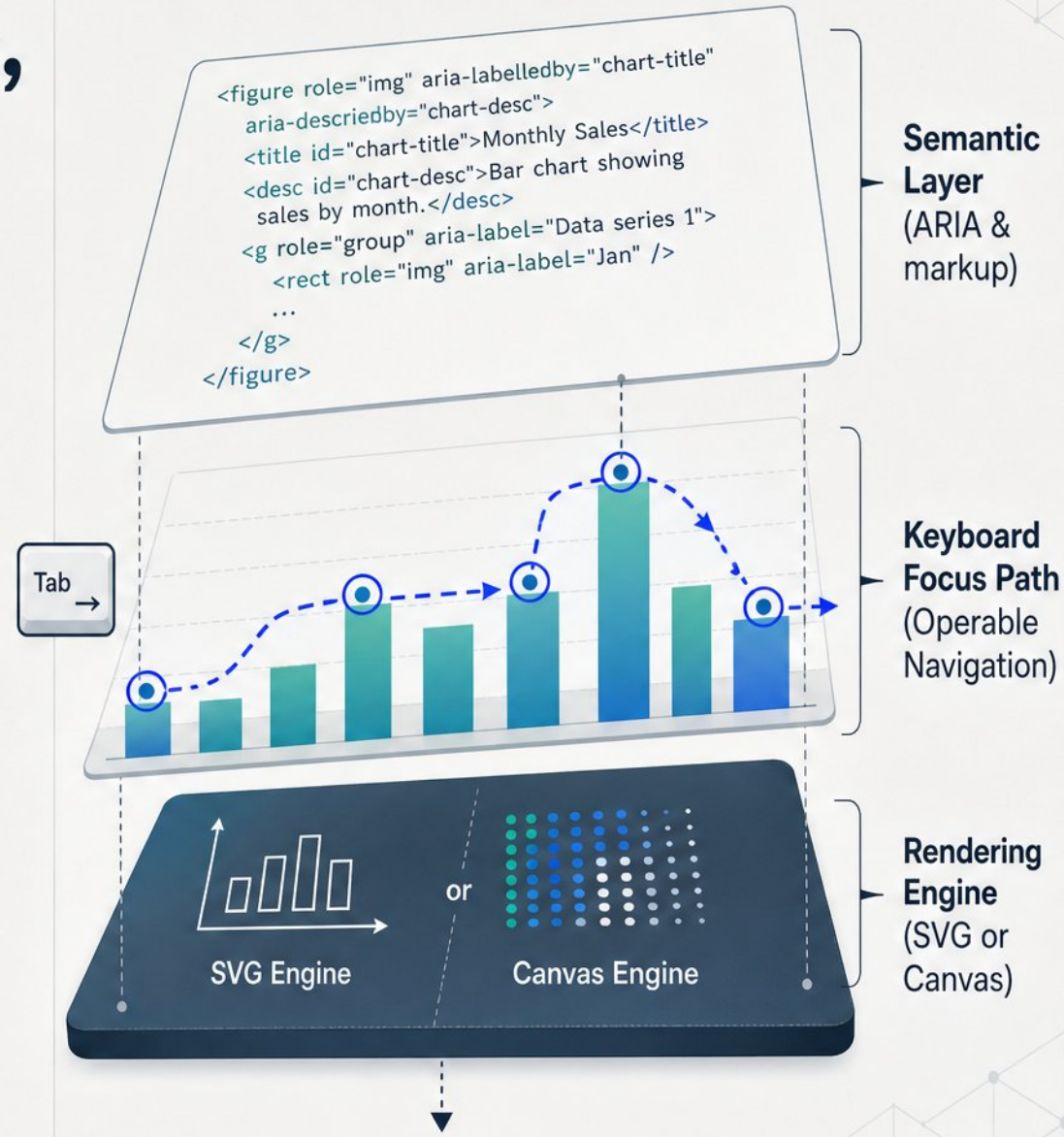


Same story. Same units. Same source.

Consistency across mediums builds trust and speeds comprehension.

Accessibility as a Selection Criterion, Not a Retrofit

- Score candidates on five axes: SVG+ARIA, palettes, keyboard, screen reader, table
- Defaults vary sharply: Plotly ships keyboard nav, ECharts needs aria on
- All 10 libraries passed axe-core; only 4 shipped real accessible structure
- Test WCAG 2.2: 1.1.1, 1.4.1, 1.4.3 (4.5:1), 1.4.11 (3:1), 2.1.1
- Colour alone never works: add labels, shapes, patterns, or a linked table
- SVG is hours of a11y work; Canvas DOM proxy is days — budget the engine choice
- Audit with POUR-CAF or Chartability; contrast fails 88% of audits
- Toolkit: MAIDR and py-maidr add braille, text, sonification to charts



Accessibility is built in the stack, not added on after the fact.

Hands-On Evaluation: Scorecard and Pilot

- Pick one realistic pilot task: weekly dashboard, single graphic, or teaching artifact.
- Score two or three tools on the same dataset: task fit, setup, data handling, design, accessibility, maintenance.
- Time-box the pilot; log setup time, spec lines, render performance, accessibility gaps, review effort.
- Expect friction in SSO, connectors, reverse proxy, and Celery tuning on self-hosted platforms.
- Decide with explicit trade-offs, a documented fallback, then roll out templates, training, and owners.

One Dataset
Use the same dataset for all tools.

Two or Three Candidate Tools
Evaluate the same task.

Time-Boxed Pilot
Measure and log evidence.

Evaluation Criteria	Tool Option 1	Tool Option 2	Tool Option 3
Task Fit *Evidence / Notes*			
Setup *Evidence / Notes*			
Data Handling *Evidence / Notes*			
Design *Evidence / Notes*			
Accessibility *Evidence / Notes*			
Maintenance *Evidence / Notes*			

Governance, Sustainability, and Risk

- **AGPL vs permissive:** embedding obligations differ, check before you ship
- **Health check:** commit concentration, bus factor, security response cadence
- **Single-maintainer risk** is a security control problem, not a footnote
- **xz-utils** (CVE-2024-3094): two years of trust-building, then a backdoor
- **Countermeasures:** SBOM, pinned versions, modest sponsorship, Scorecard signals
- **Exit plan:** portable specs, no paid-tier lock-in, decide pin, migrate, or fork

PROJECT HEALTH BOARD

Charting Library
Dependency

- Commit concentration
- Bus factor
- Security response cadence

Exit strategy: Migrate

Self-Hosted Platform
Dependency

- Commit concentration
- Bus factor
- Security response cadence

Exit strategy: Pin

CASE STUDY

xz-utils
Dependency

- Commit concentration
- Bus factor
- Security response cadence

Exit strategy: Fork

Utility Library
Dependency

- Commit concentration
- Bus factor
- Security response cadence

Exit strategy: Pin

THE TWO-TOOL REALITY

Charting library beside self-hosted platform

Practical Playbook: Cheat Sheet, Templates, and a 30-60-90 Day Plan

Days 1-30

Days 1-30: inventory charts, pick pilot task and two tools

Days 31-60

Days 31-60: run time-boxed pilot, score, document trade-offs

Days 61-90

Days 61-90: train, publish accessibility contract, set review cadence

Selection Cheat Sheet

React dashboard <5k points → Recharts; blog diagrams → Observable Plot	Recharts Observable Plot
Non-standard shapes → D3; million-point scatter → regl + D3	D3 regl + D3
BI widgets, wide chart variety → ECharts 6; JSON-spec auto-charts → Vega-Lite	ECharts 6 Vega-Lite
Workflow template: numbered scripts, lockfile, data/ specs/ docs folders, CI publish	Template Workflow Path

Playbook Checklist

- Cheat sheet: React dashboard <5k points → Recharts; blog diagrams → Observable Plot
- Non-standard shapes → D3; million-point scatter → regl + D3
- BI widgets, wide chart variety → ECharts 6; JSON-spec auto-charts → Vega-Lite
- Workflow template: numbered scripts, lockfile, data/ specs/ docs folders, CI publish
- **Days 1-30:** inventory charts, pick pilot task and two tools
- **Days 31-60:** run time-boxed pilot, score, document trade-offs
- **Days 61-90:** train, publish accessibility contract, set review cadence
- **Self-check:** dominant axis, accessibility budget, migration trigger, upgrade owner

Pipeline Stages and Owners

- **Inventory** — *Owner: Analyst*
- **Pilot & Build** — *Owner: Engineer*
- **Evaluate** — *Owner: Reviewer*
- **Document** — *Owner: Writer*
- **Accessibility** — *Owner: A11y Lead*
- **Review Cadence** — *Owner: Lead*

PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/ff0b7dcf/public