Sourcegraph

How Frontend Platform Teams can build a data-driven relationship with your codebase

Frontend platform teams are a vital part of many software orgs: they accelerate company velocity; own frontend tech stacks, tools, and libraries; and drive key cross-functional products like accessibility standards and design systems.

But because their projects are so often wide-reaching, it's difficult to answer key questions like:

1. How quickly are other teams adopting the internal component library we invested in building?
2. When will a migration be complete?
3. Which areas of the code still need to be updated after our accessibility audit?

Without a system to provide easy answers to these questions, you either spend hours and hours trying to track them manually, or you give up.

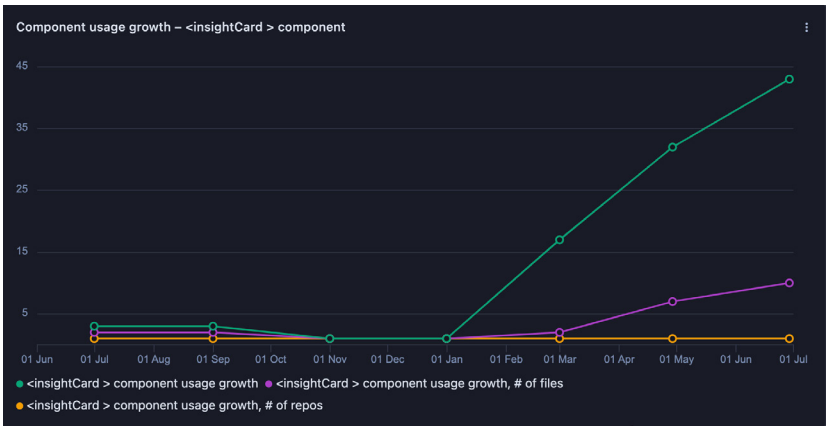
Instead, frontend platforms should build the same data-driven relationship to their work that every user-facing feature team builds out of the gate. Every platform project should have KPIs like, How many other teams use our work? and How quickly was our work adopted?

In the data-driven world, platform work is faster and more effective.

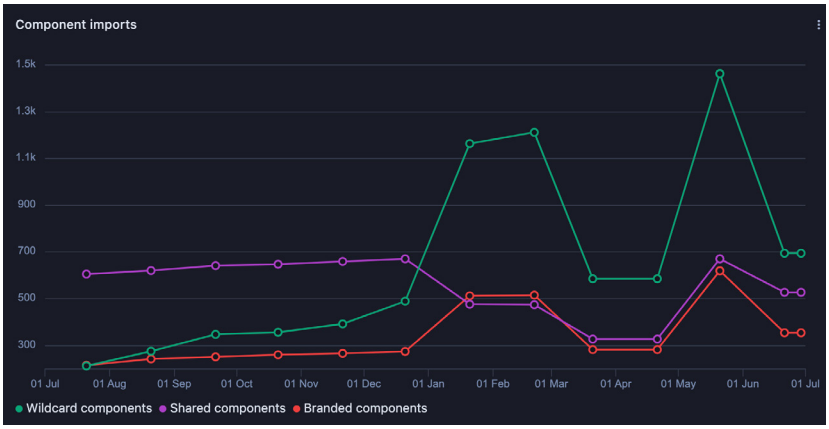
Data-driven development for internal component libraries

Tracking the adoption of individual components and entire libraries can demonstrate impact and offer key insights.

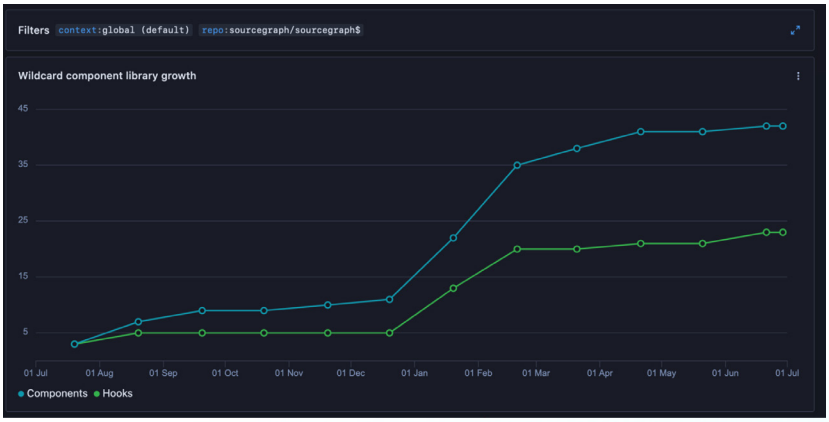
Individual component usage tracking:



Overall library usage tracking:



Metrics about the size of your library:

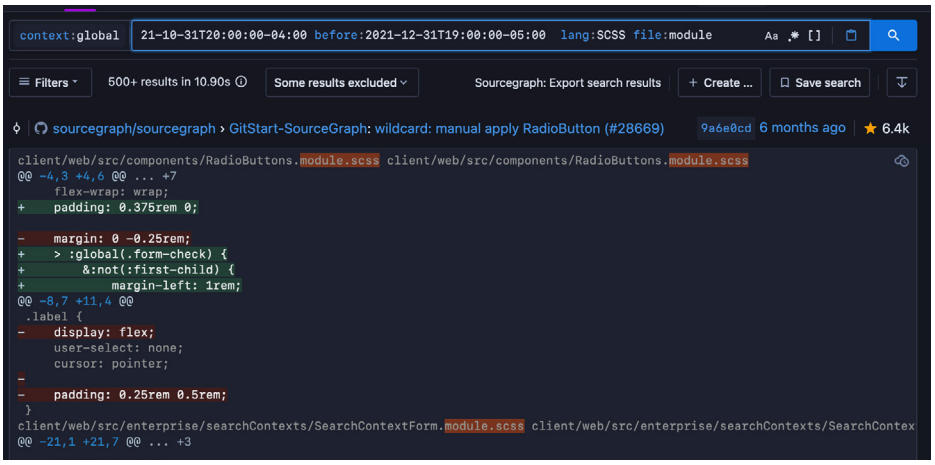


When you can track the reuse of your component libraries across your entire codebase, you can:

1. **Plan and prioritize future component additions to the library.** For example, perhaps you discover a button component was adopted across the codebase 10x faster than a new form component. What other components are often used along with the button that should be added to the library next? What other form components might be worth deprioritizing?
2. **Discover teams and projects that aren't using your component library, and understand why.** By tracking the use of non-library components, you may discover that a whole team hasn't been using the new components at all. Was there a communication gap? Are the new components not conducive to the team's needs? What could be improved?
3. **Showcase your impact with the rest of the organization (and your manager).** Platform team features are notoriously difficult to justify to folks less familiar with engineering. Being able to point to the number of teams using and referencing the tools you've built lets you celebrate and share your success, as well as ensure future projects get executive support.

Data-driven acceleration for large migration projects

Large frontend migration projects – like redesigns of modularization initiatives – can be tracked automatically to show progress and inform planning.



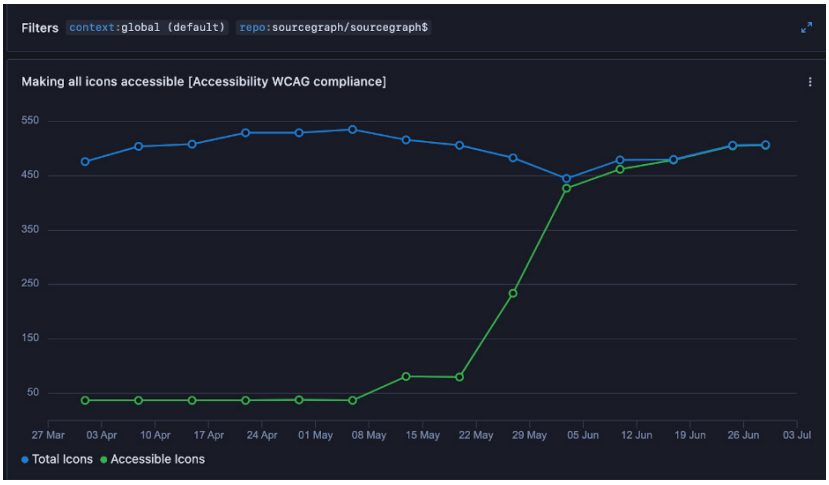
Having this tracking will **accelerate** the progress of your migration:

1. **Immediately discover and fix slowdowns.** If the project starts to stall out, you can filter the data by repository or team area and understand what's going on, then fix it. Without this visibility, you might lose weeks to thinking things are going well when they aren't.
2. **Ensure the migration gets done completely.** Often, migrations stall out at 70-80% completion because people aren't able to track the total progress.
3. **Surface what teams and code gets migrated fastest, and learn from what works.** The inverse of finding slowdowns is that you can also learn from the progress individual teams make.
4. **Motivate teammates and teams across the organization.** It's difficult to get buy-in on cross-team projects, but having a chart to burn up provides a central point of focus and excitement as it progresses.

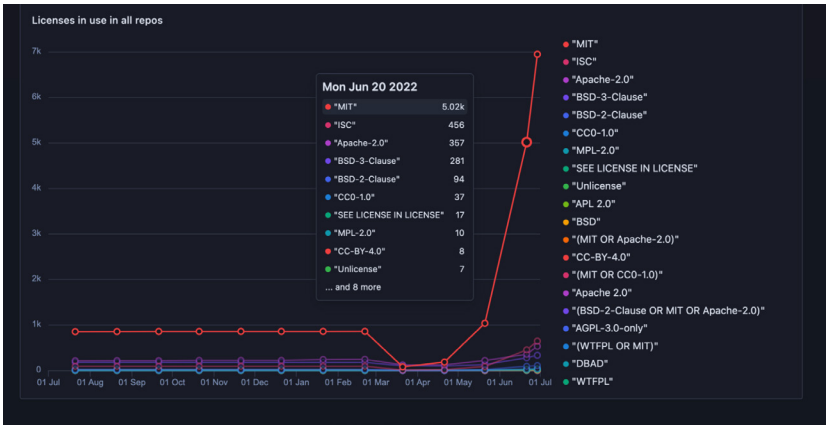
Data-driven compliance and maintenance

Compliance initiatives (like accessibility and licensing) and maintenance are critical functions of a platform team and deserve the same data-driven vigor as other work.

You can track accessibility initiatives across your user interface:



Licenses in third-party javascript libraries to ensure your platforms are legally compliant:



And internal compliance standards, like using versioned docs links:

Using data from your codebase to track compliance progress makes your work faster and more accurate, which is crucial when contractual obligations are at stake:

- **Be confident in the source of truth.** Rather than hoping everyone remembered to update a Jira epic or spreadsheet manually, just look in the code itself.
- **Save time on follow-up.** Instead of spending days following up to make sure every team is compliant with a new standard, pull the answer directly from your full codebase.
- **Notice and fix when code falls out of compliance.** Since you now have quantitative data, you can set automatic alerts to be notified if your compliance levels change.

Being a data-driven frontend platform team takes you from guessing to knowing, and from being reactive to being proactive

Every frontend platform team has dealt with a migration, built an internal tooling system, or owned a compliance initiative.

Few of them have access to data that would make these projects faster, more effective, and complete. But if we can build a world where that's the standard, teams can spend more time on proactive consolidation, building internal tools and libraries that will actually get used, and fixing problems, and less time trying to figure out the impact of their work, what they should do next, and whether or not a project is complete.
