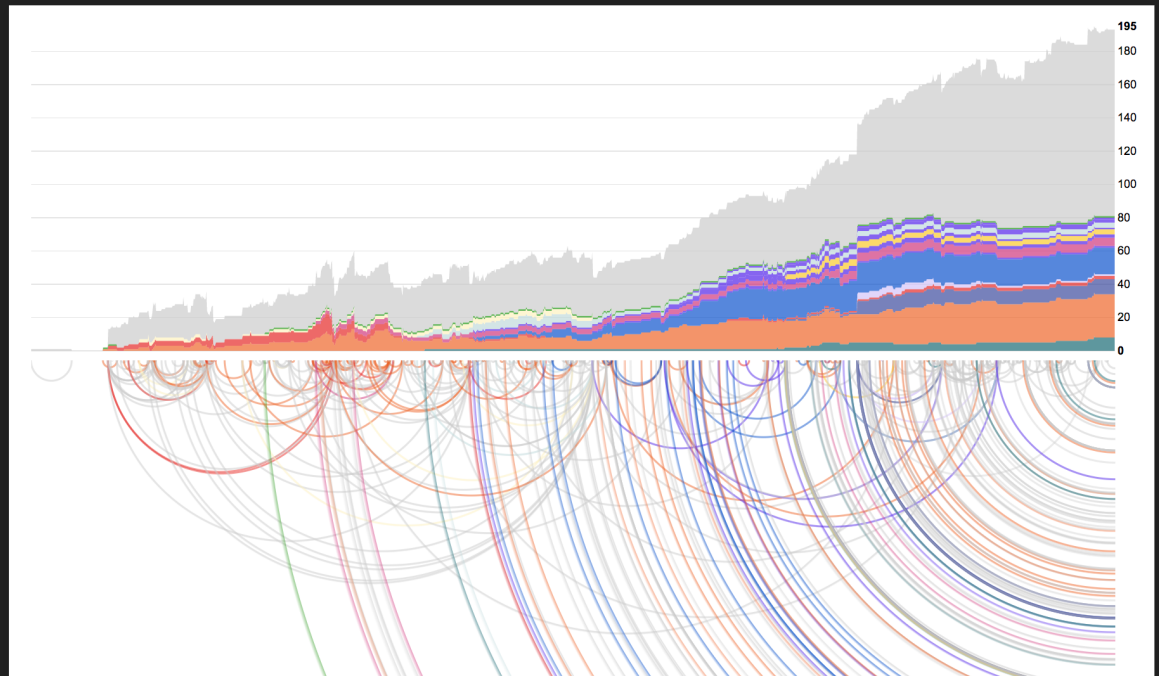
BUMBLEBEE UPDATE

ADSUG 2018

TIM HOSTETLER

SOME STATS FROM LAST YEAR

- 30 Releases
- 517 Commits
- 110 Issues Closed



PARITY WITH CLASSIC

- Author affiliation export tool
- Custom export formats
- Personalized user settings
- Tugboat support

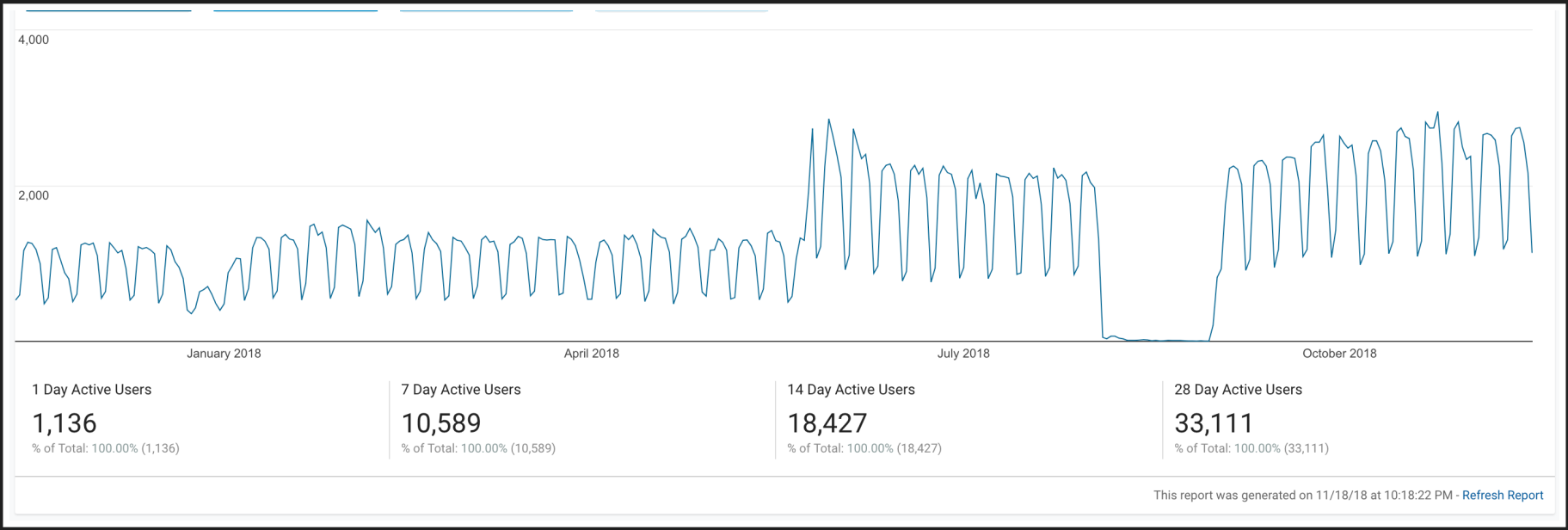
OTHER IMPROVEMENTS

- Overall speed increases
  - Reduce superfluous requests
  - Better caching/storage usage
  - Lazy loading highlights
- Better user feedback
  - Loading bar
  - Error messages

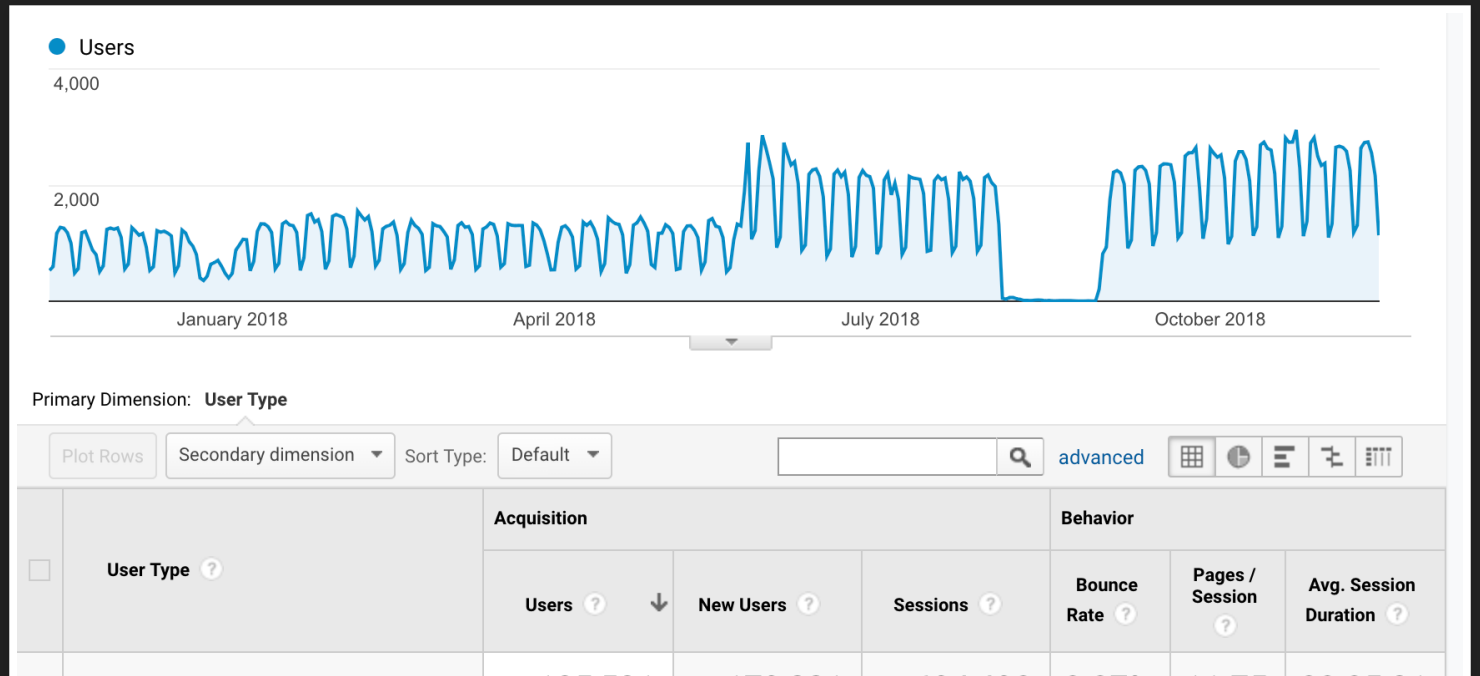
USABILITY STUDY RESULTS

- Interesting point 1
- Interesting point 2
- Interesting point 3

ACTIVE USERS



NEW VS RETURNING



IN THE WORKS

- Library set operations
- MyADS dashboard
- Expanded configuration

CHALLENGES

- Aging tech stack
- Difficult to maintain
- Only so fast, without major refactor
- Lack of front-end devs/specialists

CONSIDERATIONS

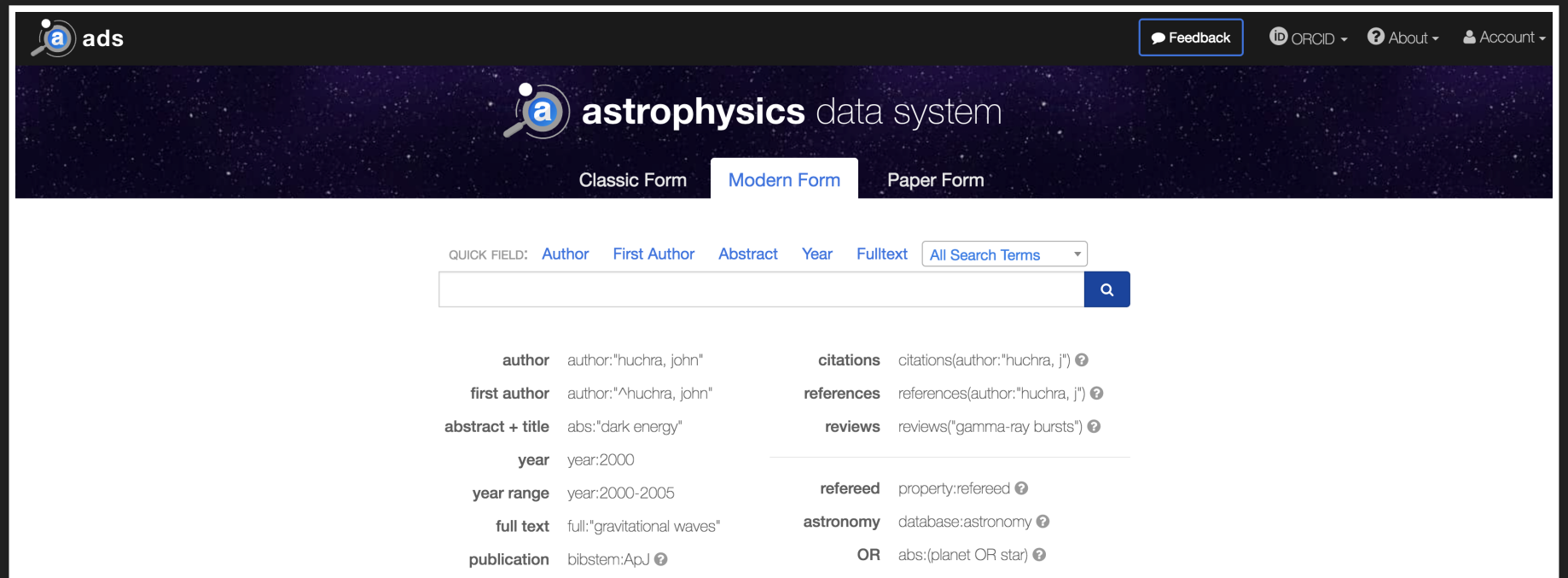
- Refactor/Update codebase
- Move to newer front-end frameworks/technologies
- How to transition properly
- Balance priorities

WHAT REFACTORING COULD FIX

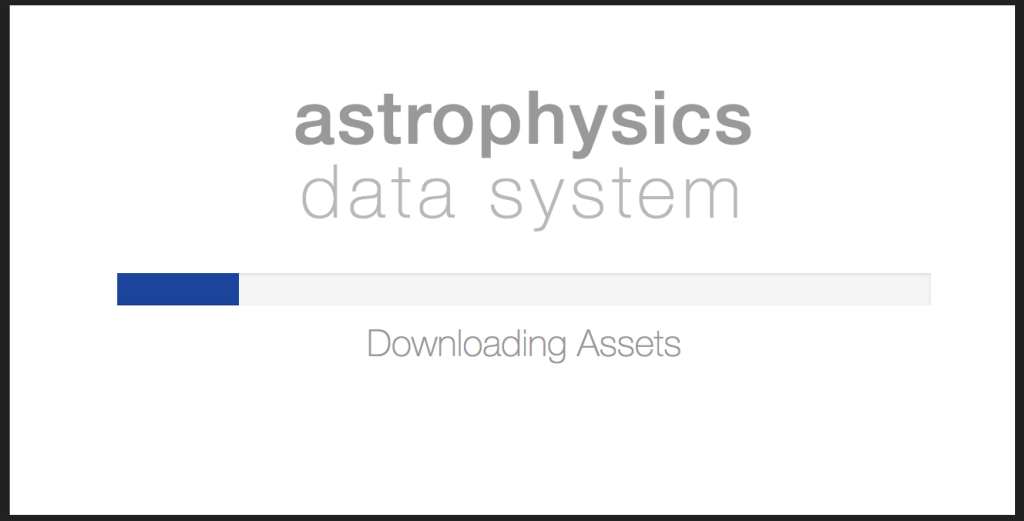
- Speed improvements (perceived & actual)
- Easier maintenance, less bugs
- Search engines and other non-js crawlers
- Better UX

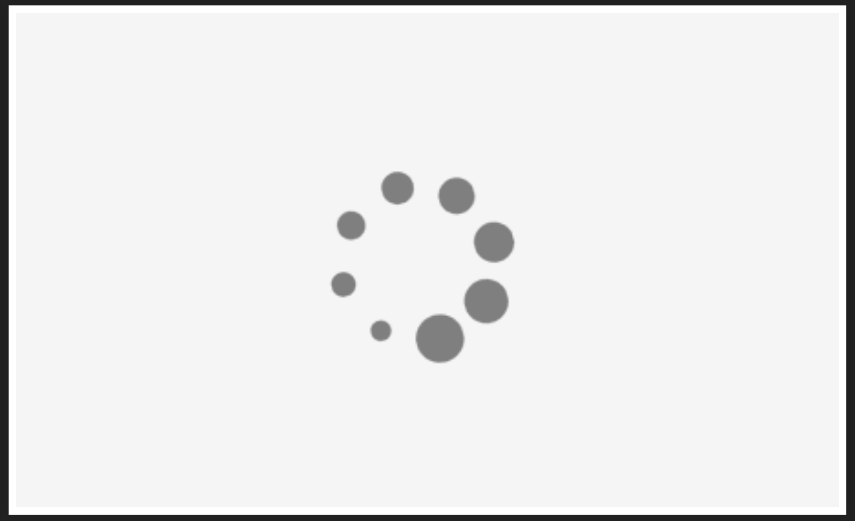
QUESTIONS?

BUMBLEBEE IS SHOWING IT'S AGE



MOST COMPLAINTS

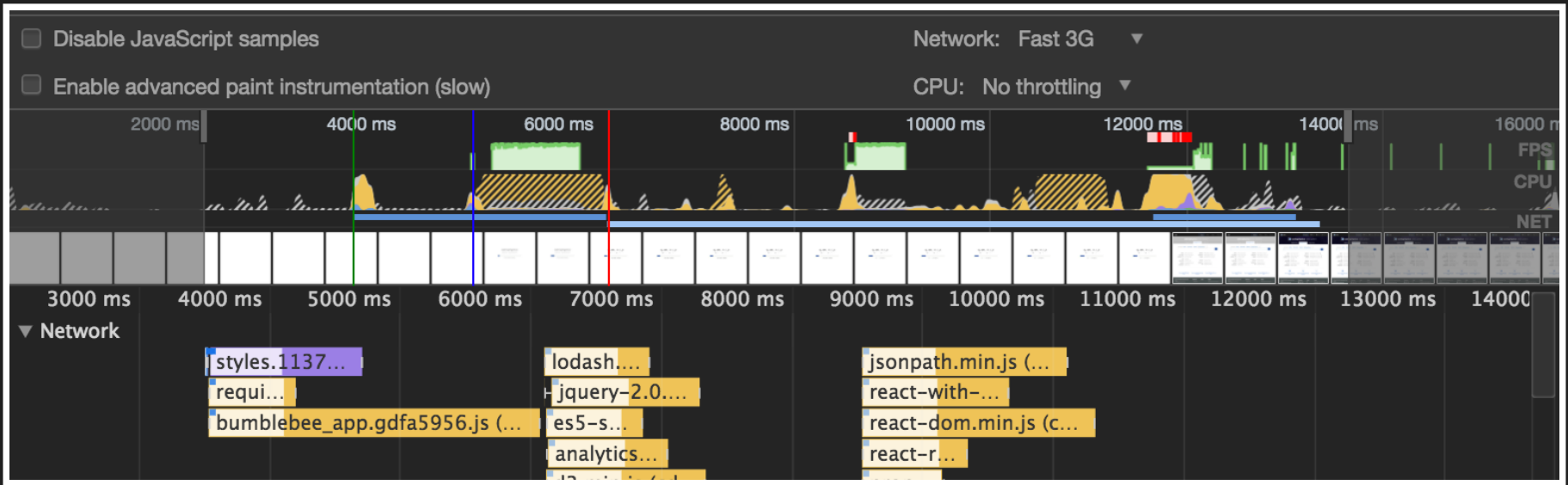




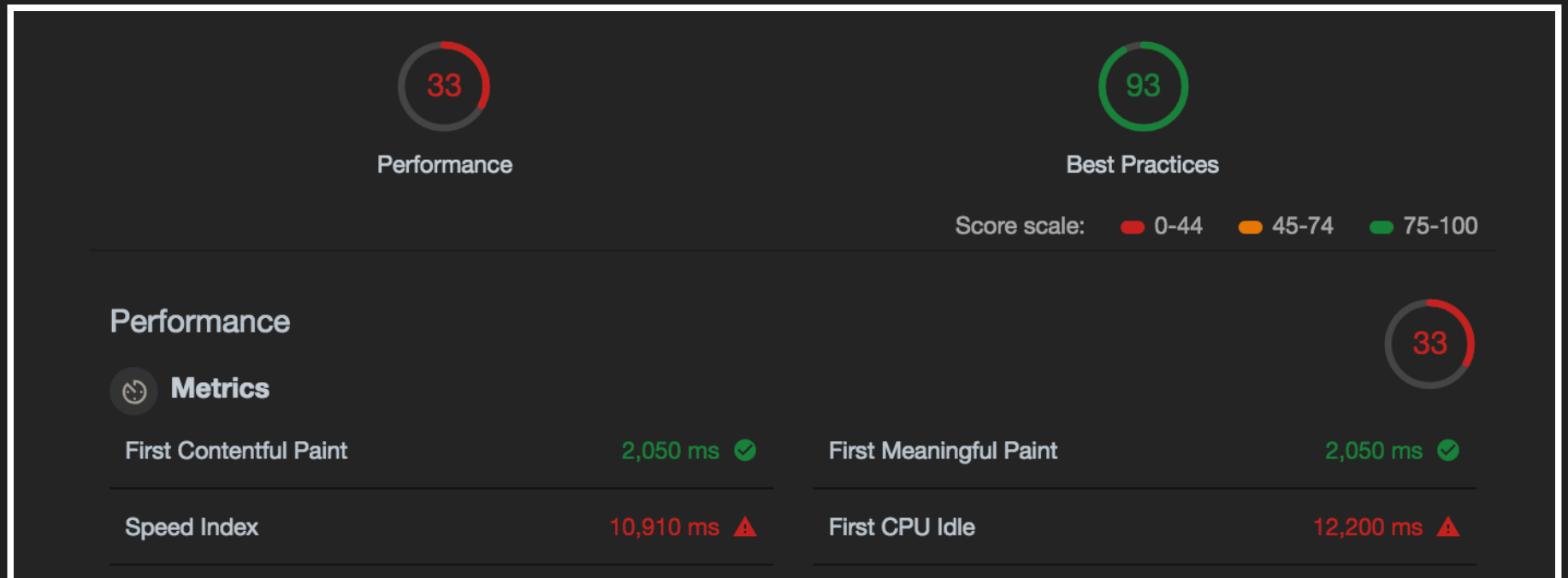
LOAD TIME

Average application load time: **5.79s**

Many unnecessary page renders/reflows



LIGHTHOUSE

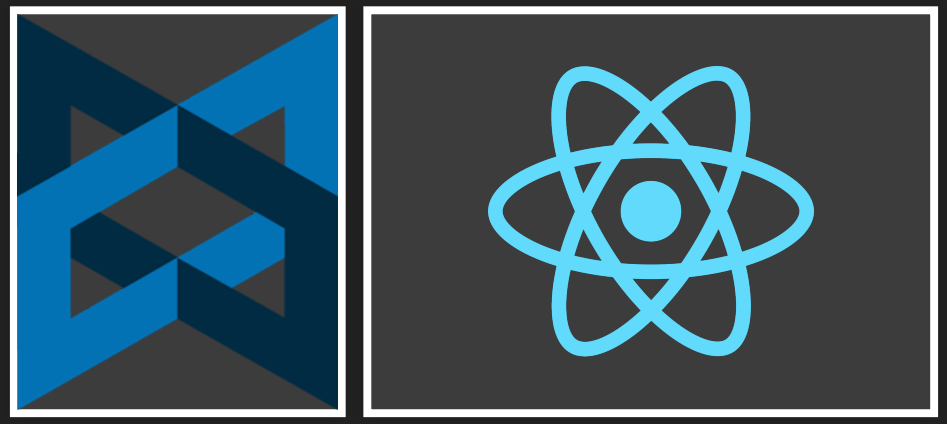

33
Performance
93
Best Practices
Score scale: 0-44 45-74 75-100
Performance
33
Metrics
First Contentful Paint 2,050 ms
First Meaningful Paint 2,050 ms
Speed Index 10,910 ms
First CPU Idle 12,200 ms

WHY?

- Based on older (+4 years) tech stack
- Hard to maintain
- Hard to integrate with user applications
- Bad mobile performance

TECH STACK

BackboneJs & React



MAINTENANCE

- Separation of concerns (DRY)
- Complicated builds
- Too many dependencies (100+ 3rd party libs)
- Testing...

INTEGRATIONS & SEO

- Zotero, Bookends, etc.
- Unfurl links
- Scrapers, non-js users
- Google, Bing, others

UNFURL LINKS

10:37 AM **tim** https://stackoverflow.com/questions/45418820/how-does-webpack-actually-works-what-does-it-does-behind-the-loaders

> stackoverflow.com
>
> **How does Webpack actually works? What does it does behind the loaders?**
>
> Every video about what Webpack is just tell you about general concepts like "it's a module bundler, it's take a module and bundle it", but I always wanted to know, how it's engine works?

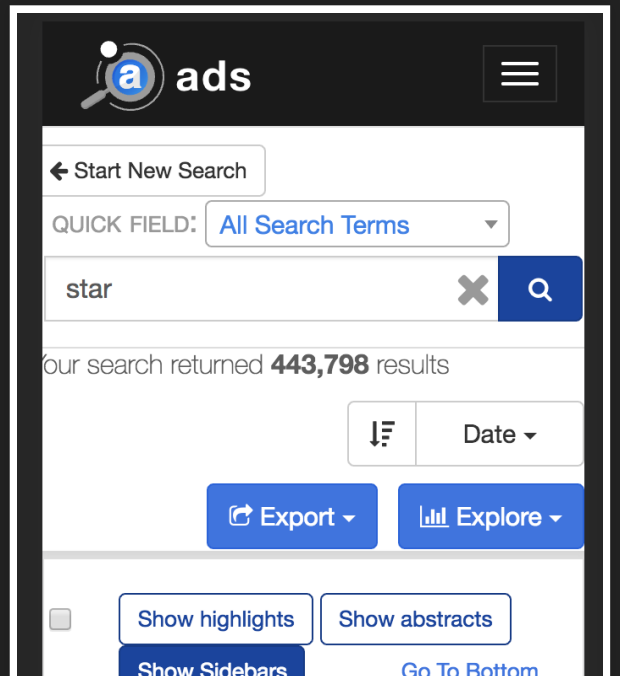
5:02 PM **tim** https://ui.adsabs.harvard.edu/#abs/2018EPJWC.18608001A/abstract

> ui.adsabs.harvard.edu

USER EXPERIENCE

- Not completely responsive
- Bad user feedback
- Confusing or hidden features

MOBILE RESPONSIVENESS



ACCESSIBILITY

- Facets not useable by screen readers
- Page regions can get readers into loops
- Some sections inaccessible

IT GETS BETTER

TYPES OF WEB APPLICATIONS

- Server-rendered (traditional)
- Client-rendered (single page application)
- Hybrid (universal)

HYBRID APPROACH

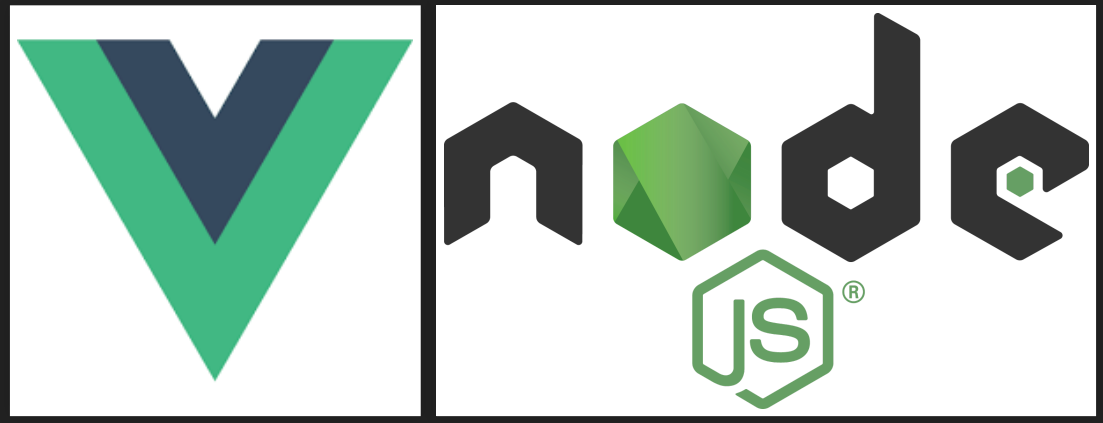
- Lives on both client and server
- Search engine optimization is easier
- Less load on server

PROPOSAL

- Hybrid web app
- New tech stack
- Updated UI framework

NEW TECH STACK

- Vue.js
- Node.js server



HOW WILL IT BE BETTER?

- Better state management
- Easier maintenance
- search engine optimization, non-js crawler support
- Responsive
- FASTER!

SPEED IMPROVEMENTS

- Consistent load times < 1s
- Fewer page renders, utilizing virtual DOM
- Pre-rendering on server, hydrate on client

* tested on prototype (wip)

SEO, NON-JS

- Metadata rendered server-side
- Non-js users possible
- Google, Bing, unfurl crawlers should just work

USER EXPERIENCE

- Consolidate and simplify interface
- Easier to implement user requested features
- Better mobile usability
- Prioritize based on user feedback

EASIER MAINTENANCE

- Support newer language features
- Better tooling
- Batteries included (fewer dependencies)
- Simplified builds
- Easier testing...

HOW WE GET THERE

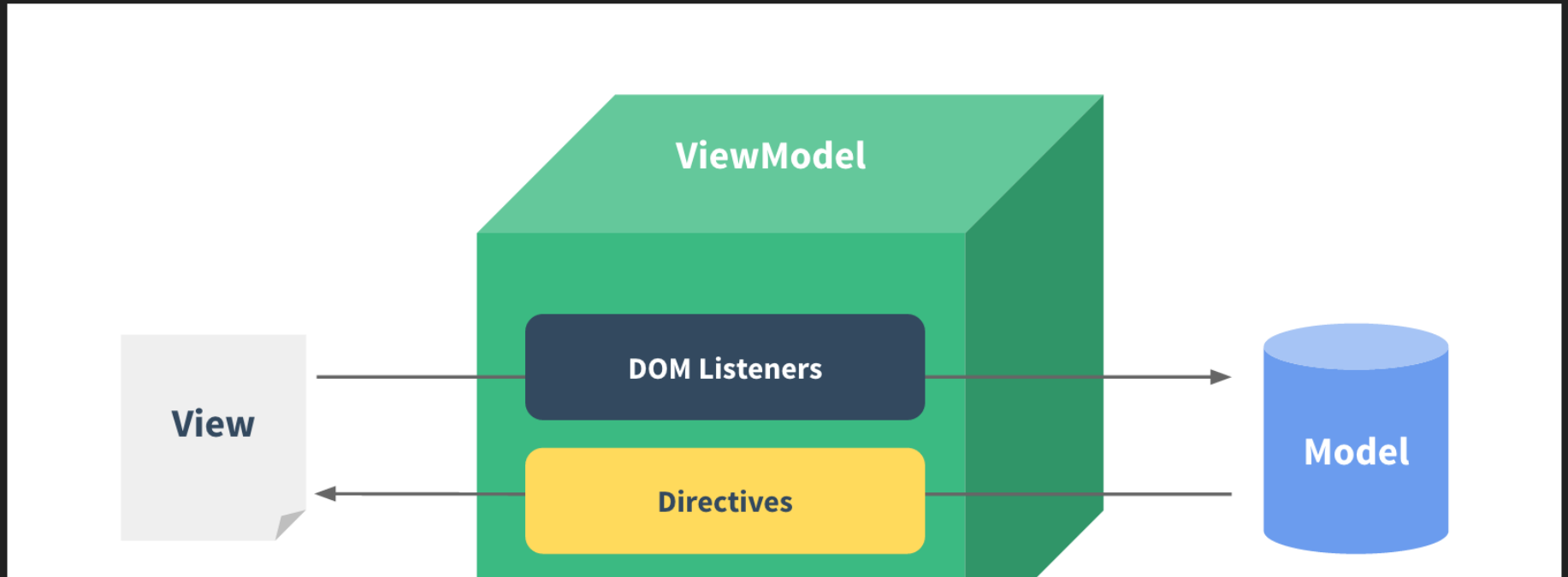
- Visual components need to be refactored
- User-facing part should stay mostly untouched
  - Improvements to responsiveness
  - Restructure of certain areas to provide better UX
- Gradual transition to new system

QUESTIONS?

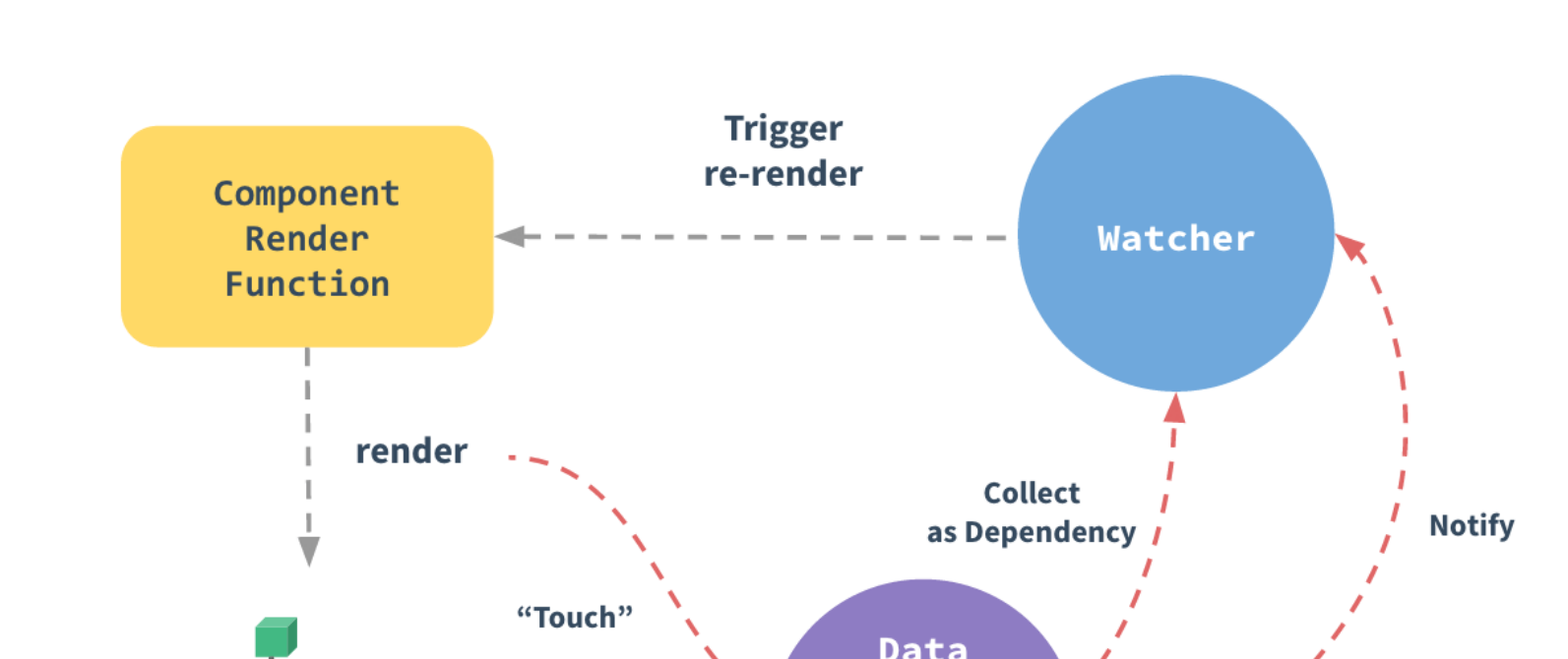
VUE INTERNALS

- MVVM pattern
- Action -> Model -> Diff -> Render
- Virtual DOM

VUE MVVM

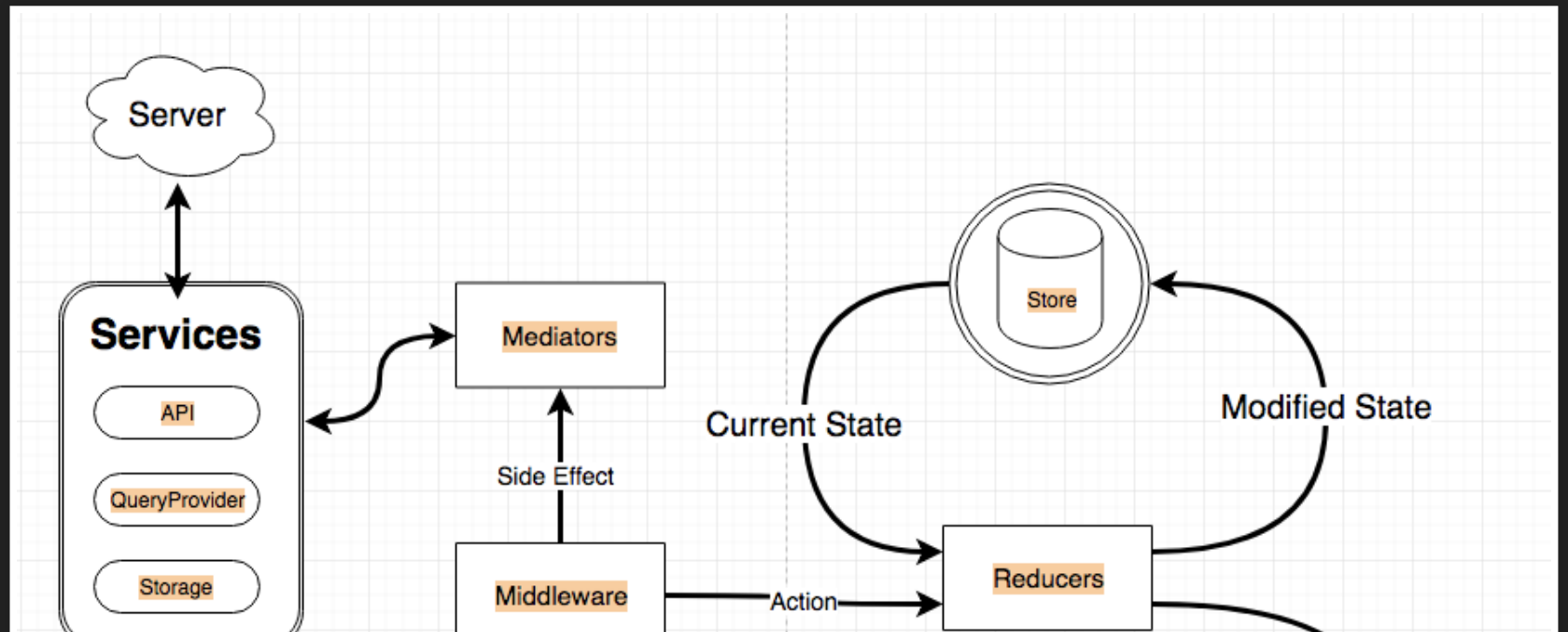


VUE DATABINDING

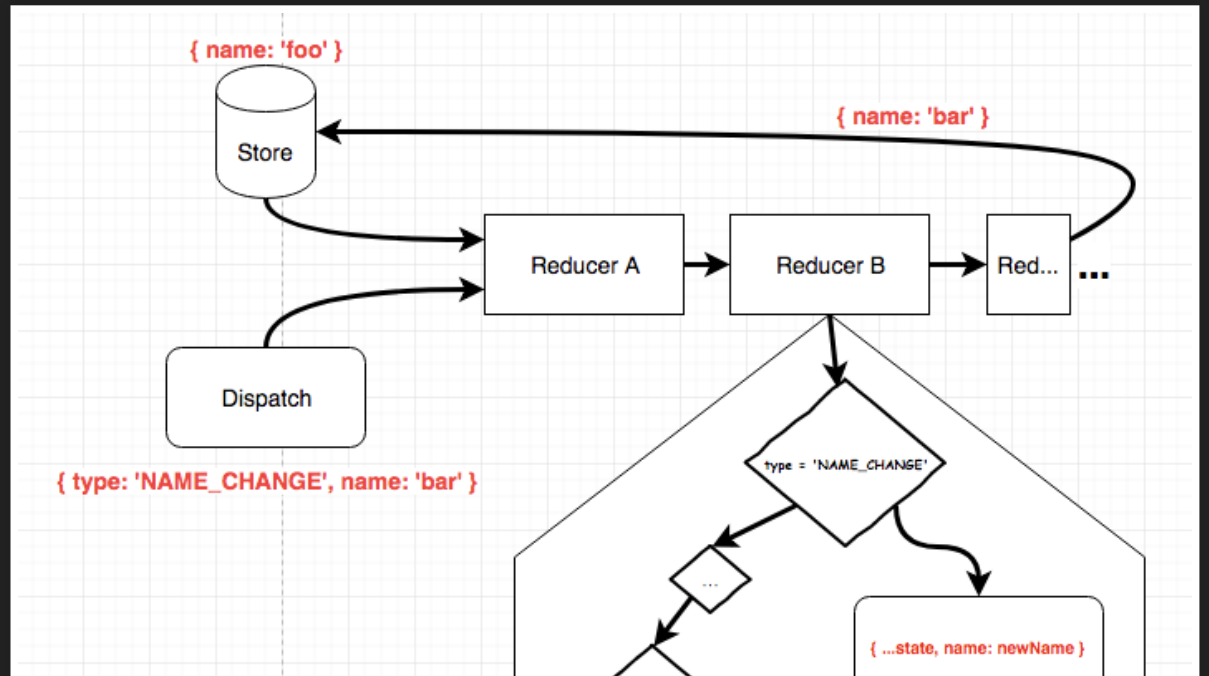


FLUX

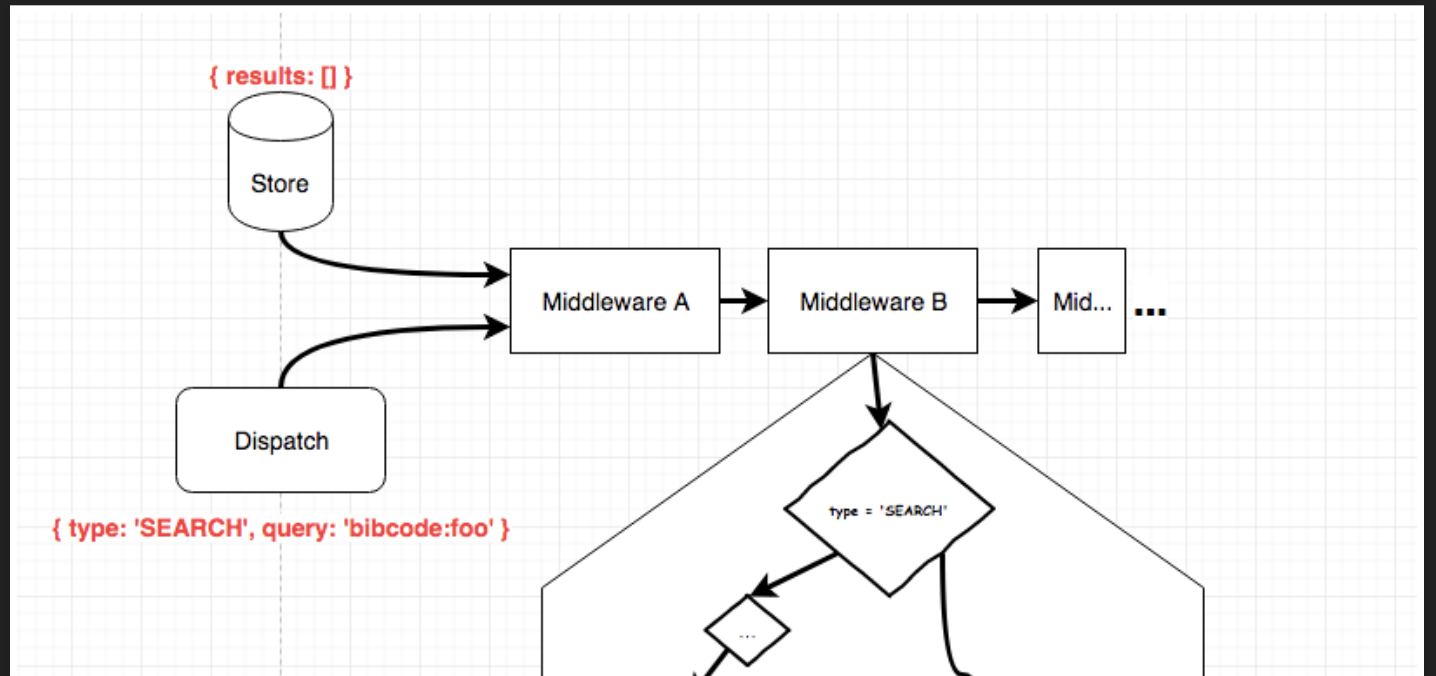
- Technique for state management
- Reducers and single store
- Non-immutable data structures
- Pure functions


Server
Services
API
QueryProvider
Storage
Mediators
Side Effect
Middleware
Action
Store
Current State
Modified State
Reducers

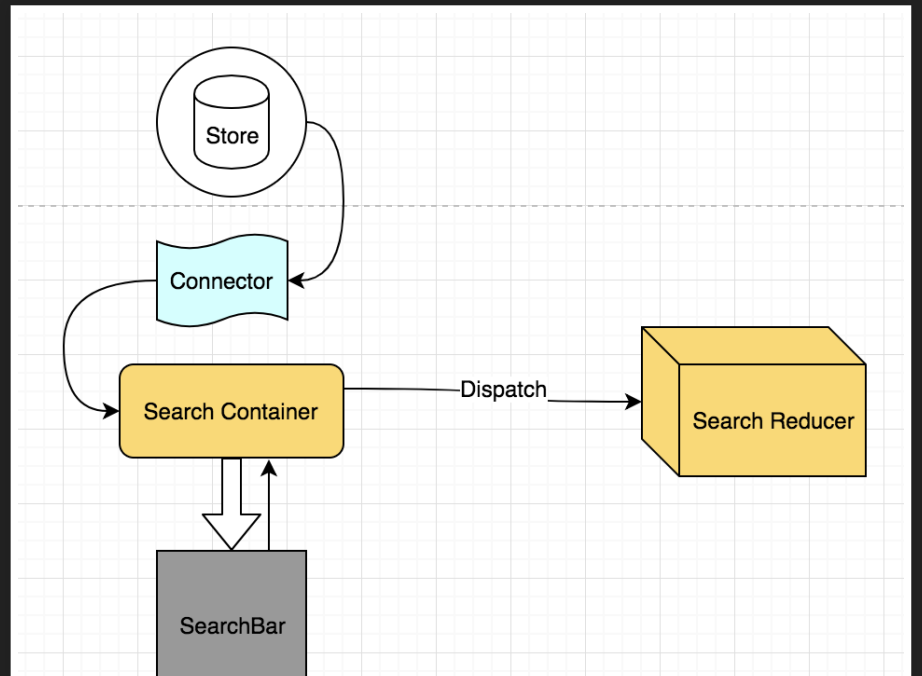
REDUCERS


{ name: 'foo' }
Store
{ name: 'bar' }
Reducer A
Reducer B
Red...
...
Dispatch
{ type: 'NAME_CHANGE', name: 'bar' }
type = 'NAME_CHANGE'
...
{ ...state, name: newName }

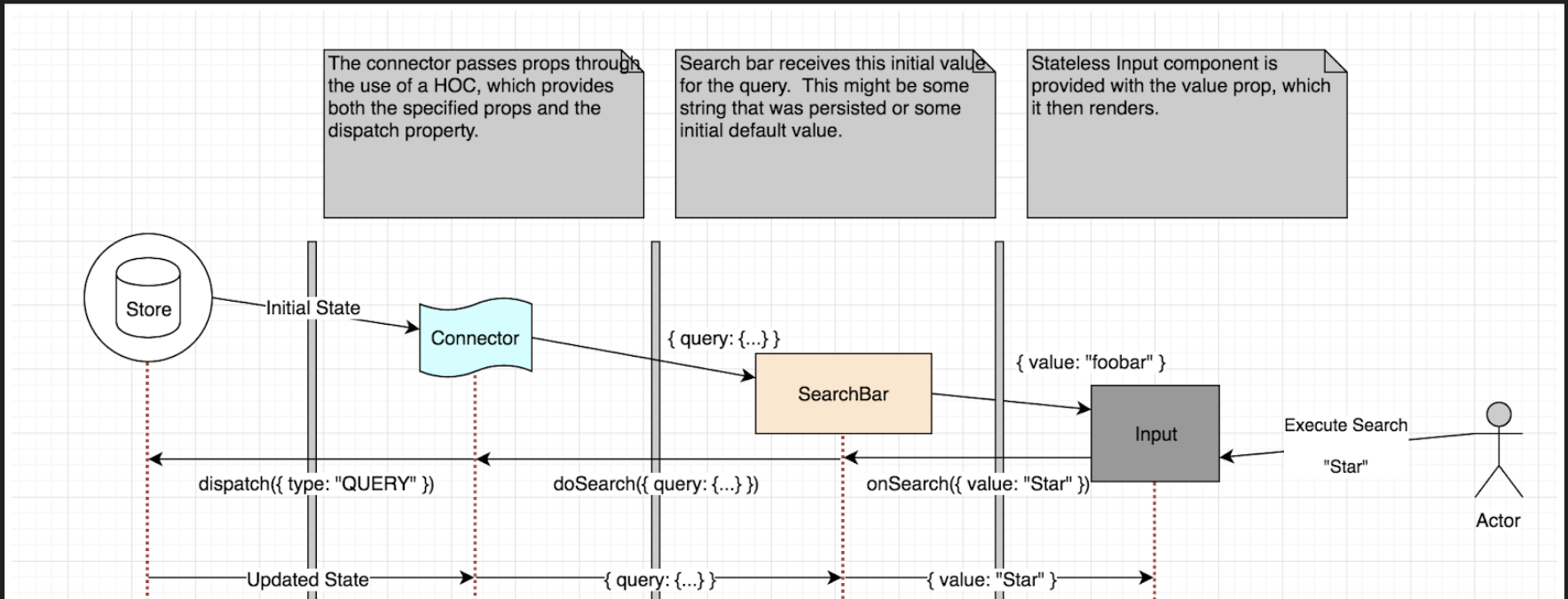
MIDDLEWARE


{ results: [] }
Store
Middleware A
Middleware B
Mid...
...
Dispatch
{ type: 'SEARCH', query: 'bibcode:foo' }
type = 'SEARCH'
...

COMPONENT COMMUNICATION



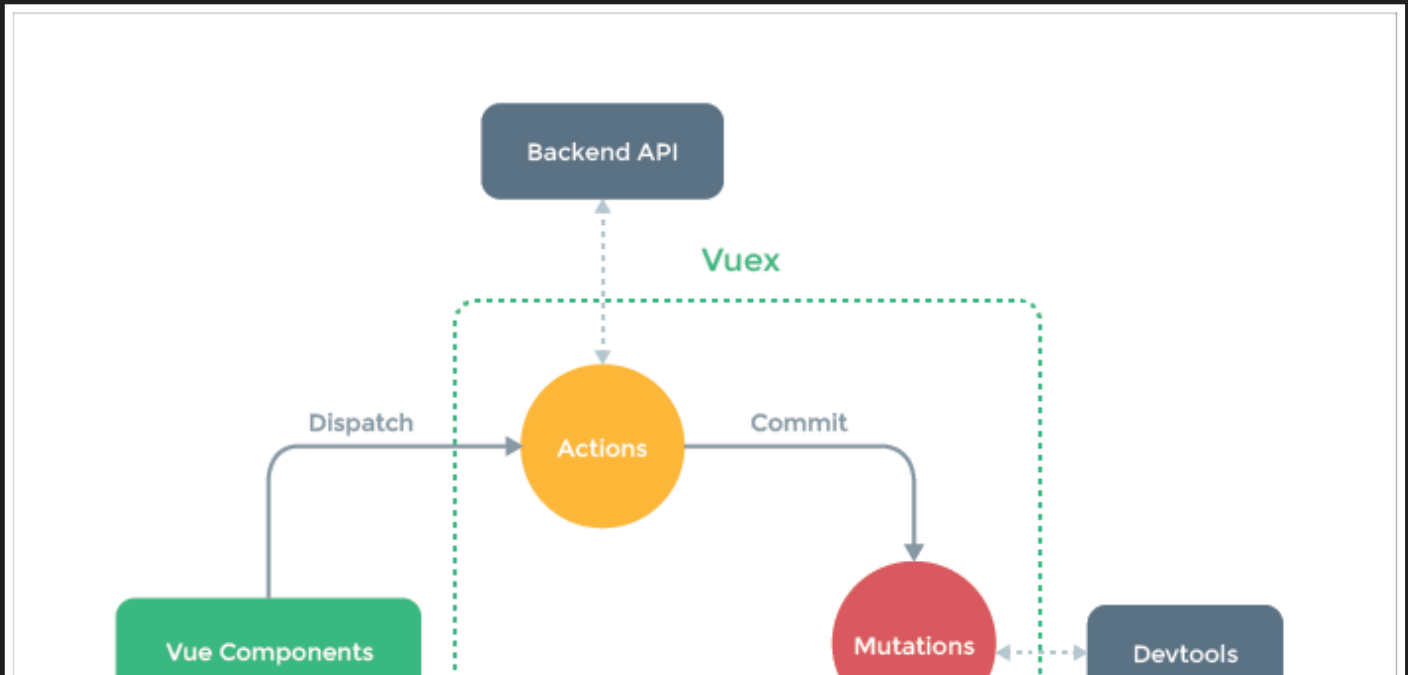
CONNECTED COMPONENTS



VUEX

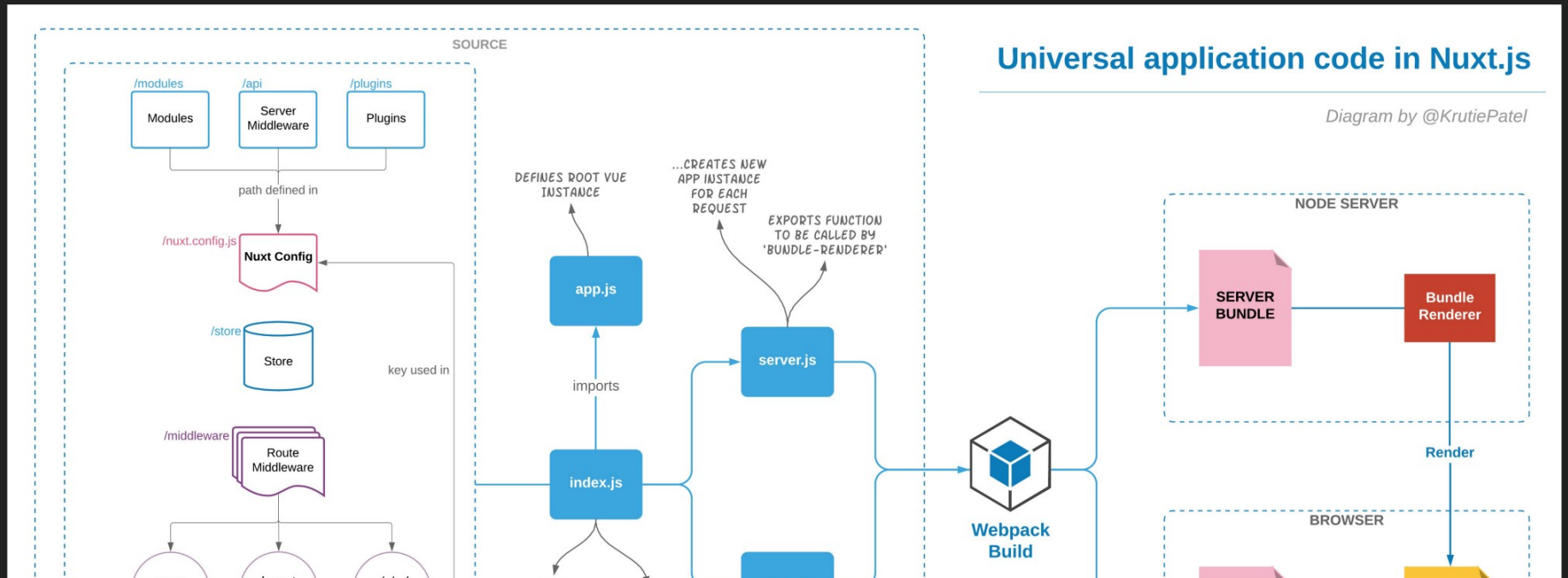
- Vue's implementation of Flux
- Baked into the core framework

VUEX


Backend API
Vuex
Dispatch
Actions
Commit
Vue Components
Mutations
Devtools

NUXT

- SSR skeleton/framework for Vue
- Provides scripts, hooks, middleware, etc.
- Based on Node.js server
- Selective rendering
- Bundling of client/server automatically


Universal application code in Nuxt.js
Diagram by @KrutiePatel
SOURCE
/modules
/api
/plugins
Modules
Server Middleware
Plugins
path defined in
/nuxt.config.js
Nuxt Config
/store
Store
key used in
/middleware
Route Middleware
DEFINES ROOT VUE INSTANCE
...CREATES NEW APP INSTANCE FOR EACH REQUEST
EXPORTS FUNCTION TO BE CALLED BY 'BUNDLE-RENDERER'
app.js
imports
index.js
server.js
Webpack Build
NODE SERVER
SERVER BUNDLE
Bundle Renderer
Render
BROWSER