Comparison of Modern JS Frameworks

The Process

Ground Rules

Goal: Build the same app using Vue, React, Angular & Ember to compare and contrast approaches.

- Latest version only
- No CSS-in-JS
- No external libraries*
- Same application / functionality
- Generate app via CLI
- No unit / integration / e2e tests
- Mobile design only
- No outside help - docs / google only

Disclaimer!

ONE DOES NOT SIMPLY
RN 3 NEW JS FRAMEWORKS IN 4 D

The App

GitHub Repos

- **Ember:** /johnstonbl01/game-shop-ember
- **React:** /johnstonbl01/game-shop-react
- **Vue:** /johnstonbl01/game-shop-vue
- **Angular:** /johnstonbl01/game-shop-angular

[redacted]

Deployed Apps

- **Ember:** game-shop-ember.netlify.app
- **React:** game-shop-react.netlify.app
- **Vue:** game-shop-vue.netlify.app
- **Angular:** game-shop-angular.netlify.app

Build Details

Order	Framework	Version	Build Time
1	React	16.13.x	~2d
2	Vue	2.6.x	~1d
3	Angular	9.1.x	<1d
4	Ember	3.18.x	~2d

Objective Comparisons

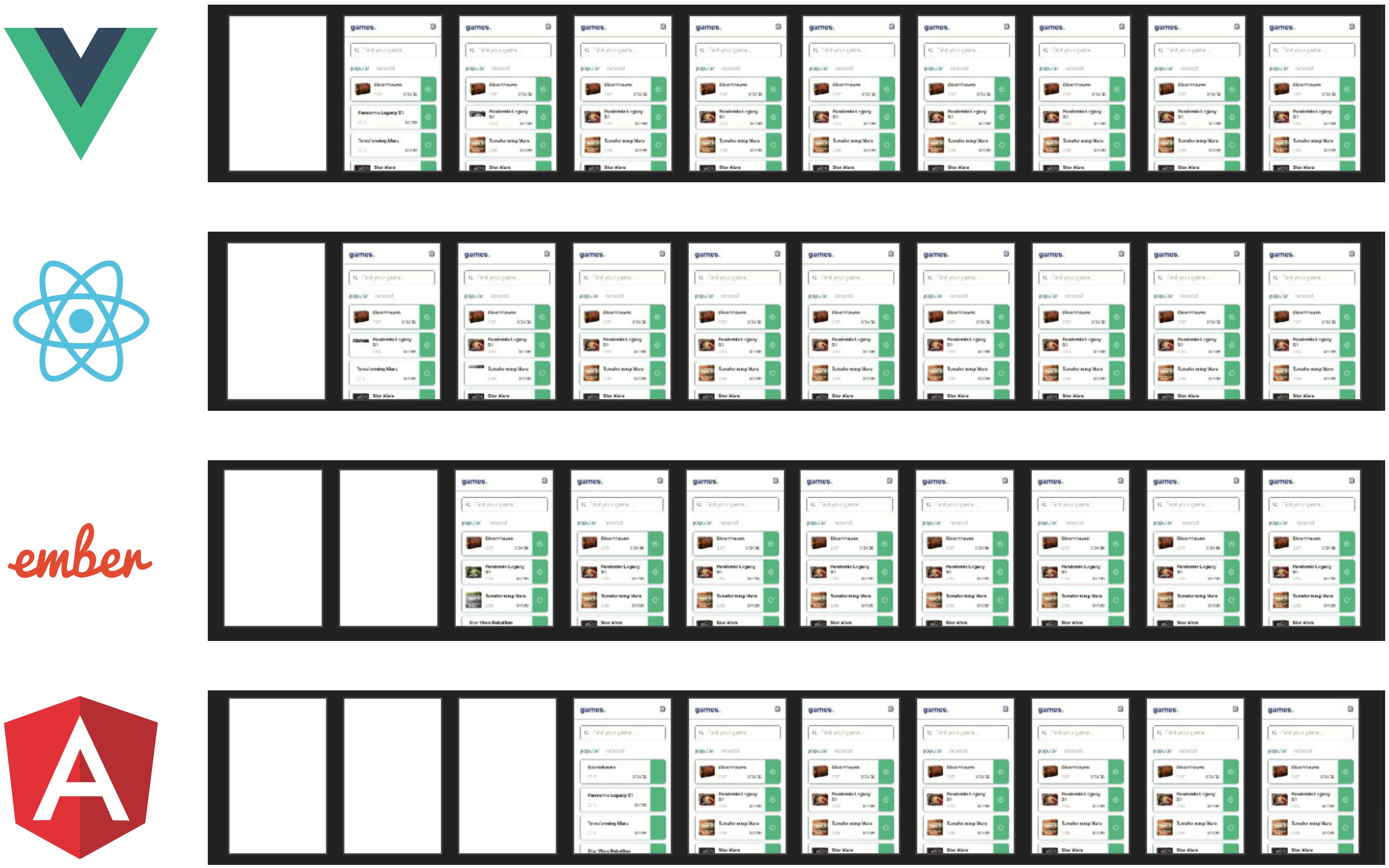
Project Metrics

Framework	LOC	Folders	Files
React	304	10	9
Ember	313	8	14
Vue	361	1	10
Angular	425	10	23

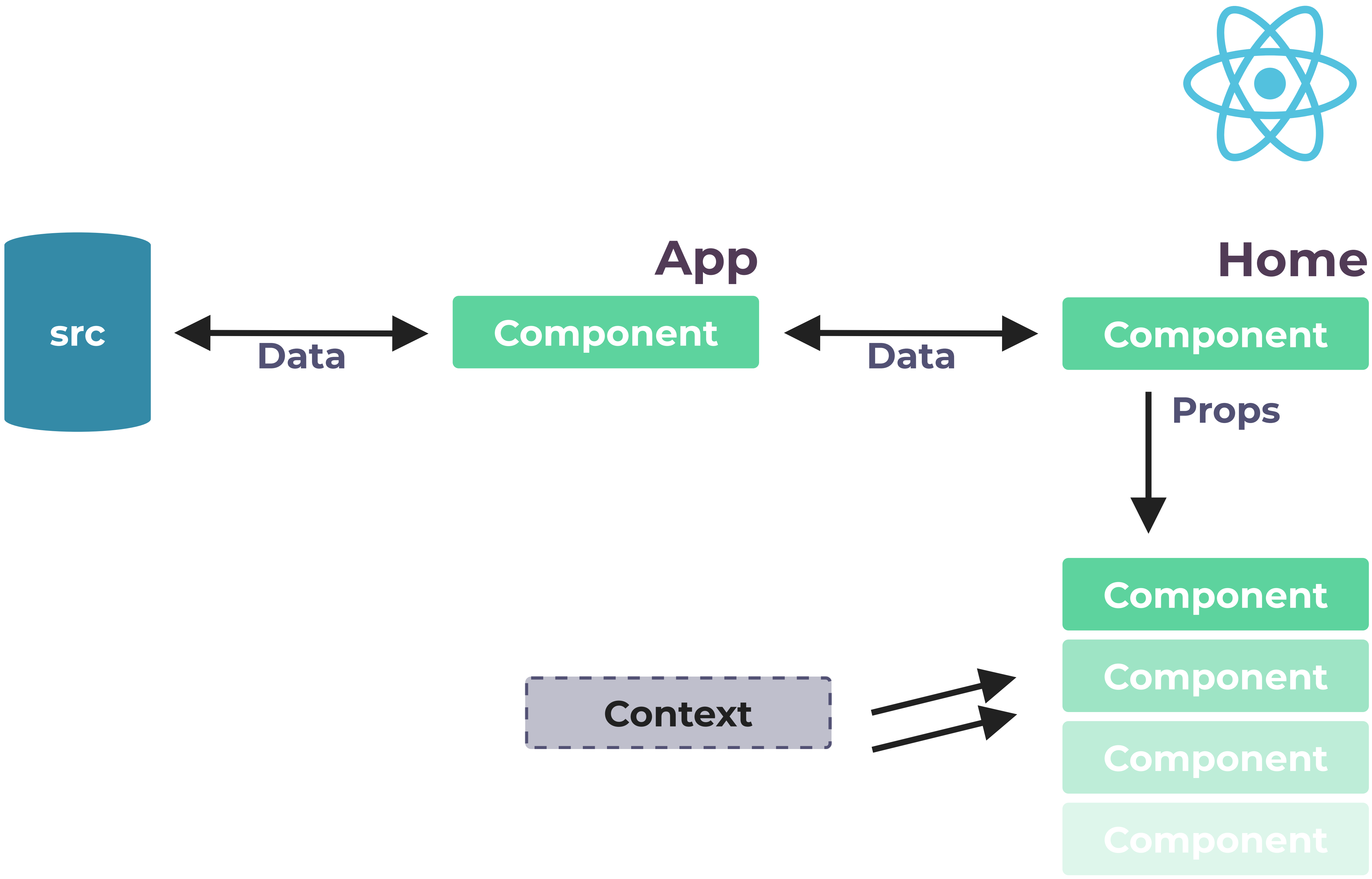
Performance

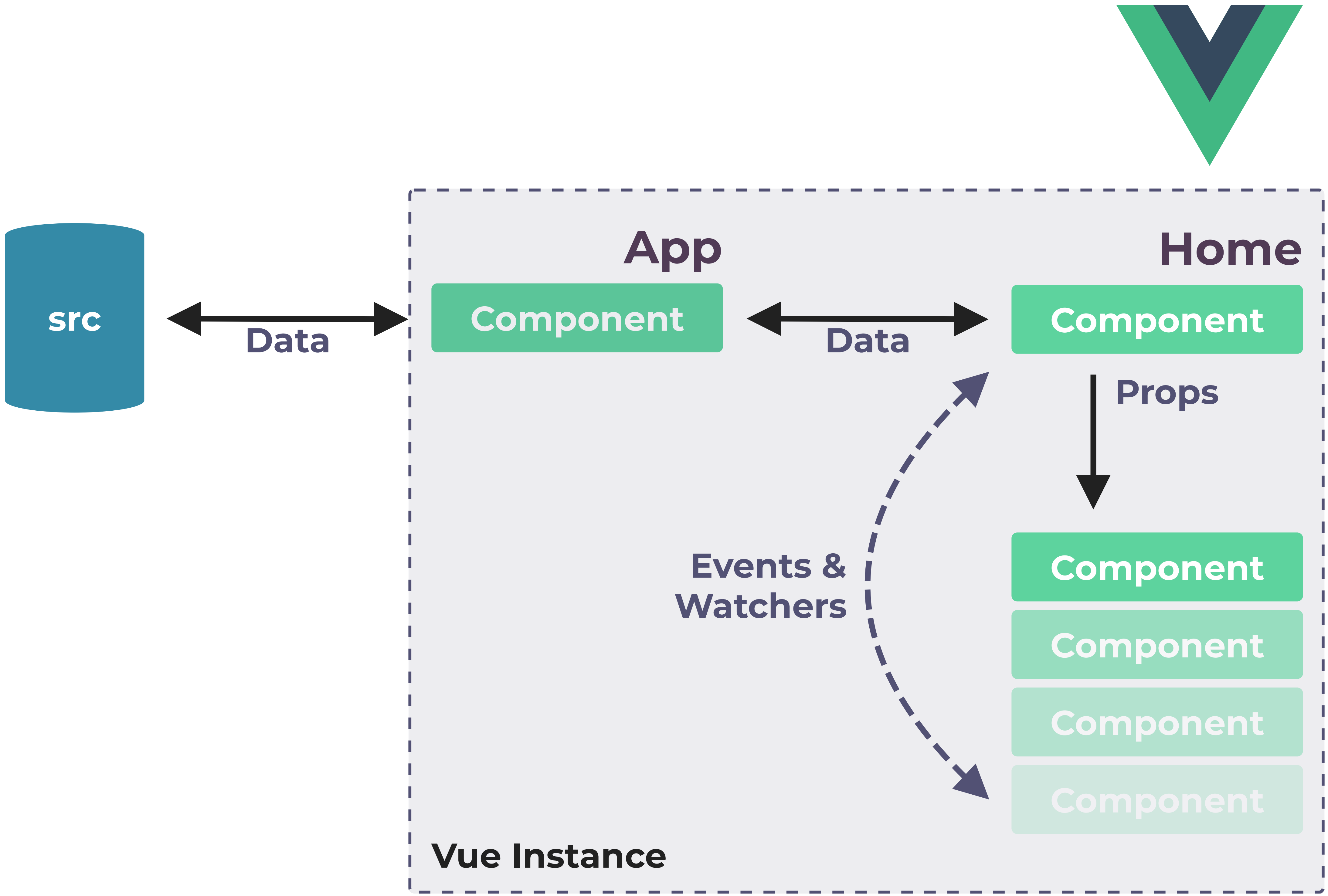
Framework	TTI (s)	JS Bundle (kb)	Lighthouse
Vue	1.6	53.7	100
React	1.8	52.3	100
Ember	3.0	187.3	96
Angular	5.3	523.5	72

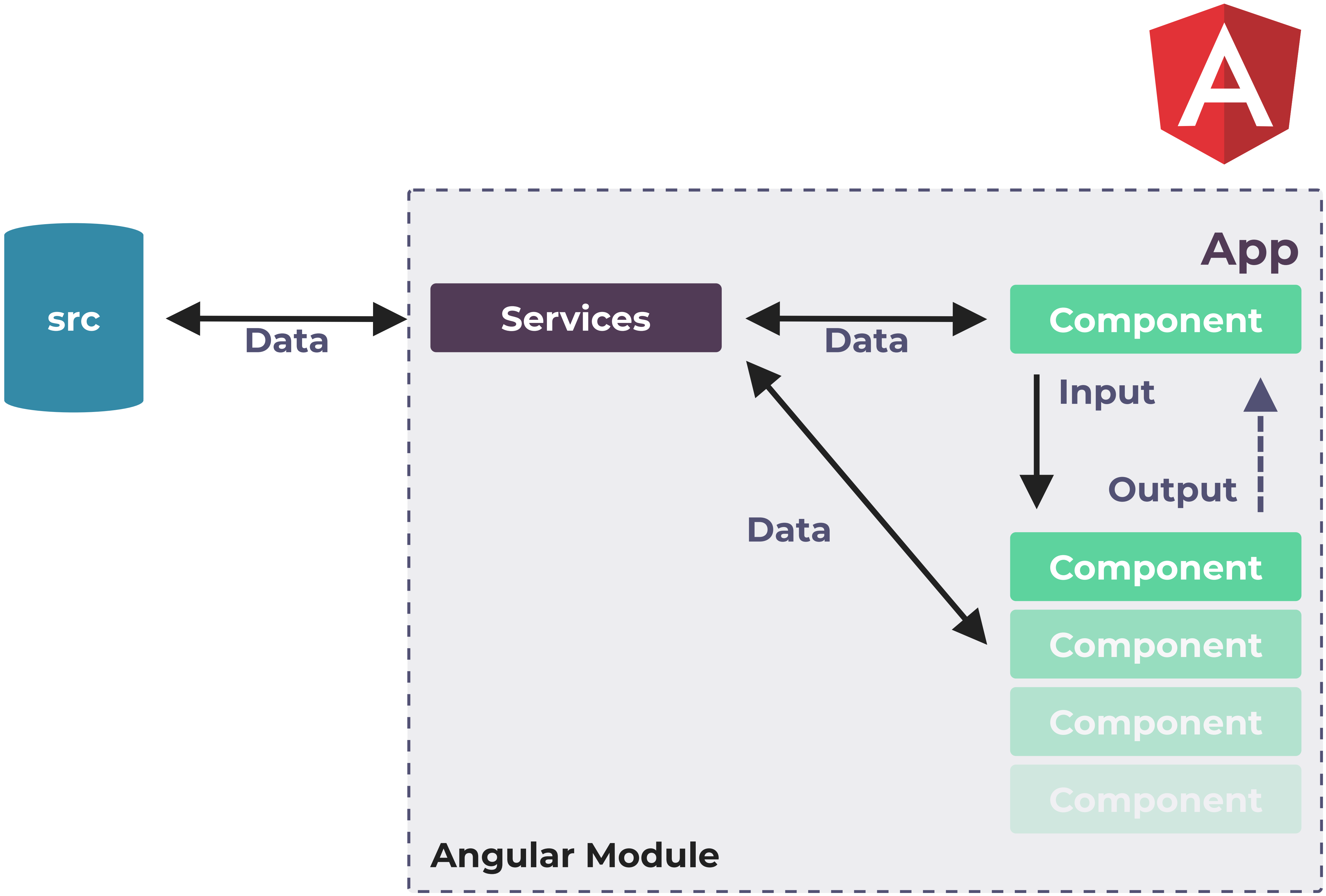
Perf Timelines



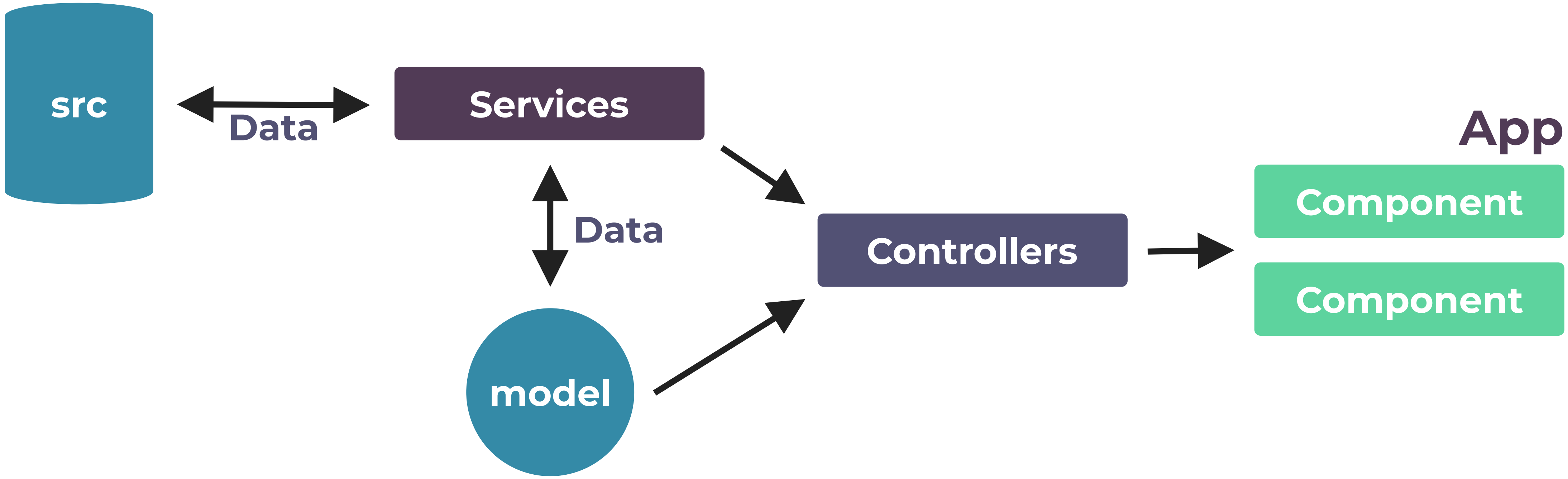
App Architecture


src
Data
App
Component
Data
Home
Component
Props
Component
Component
Component
Component
Context


src
Data
App
Component
Data
Home
Component
Props
Events &
Watchers
Component
Component
Component
Component
Vue Instance


src
Data
Services
Data
App
Component
Input
Output
Data
Component
Component
Component
Component
Angular Module

ember



src

Data

Services

Data

model

Controllers

App

Component

Component

Subjective Comparisons

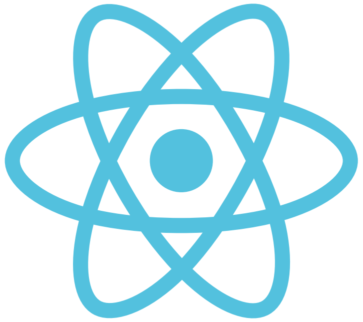
Disclaimer!

Yeah, well, that's just, like,
your opinion, man.

Documentation & CLI

- **React:** 4 / 5
- **Vue:** 4 / 5
- **Angular:** 3.5 / 5
- **Ember:** 3 / 5

General Thoughts

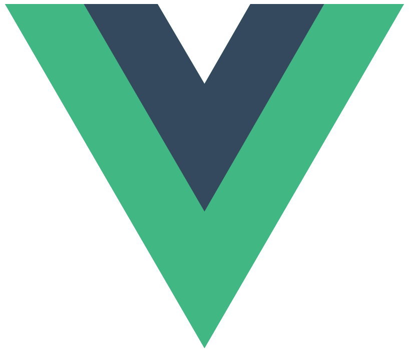


Pros

- The slogan "it's just JavaScript" feels right
- Easy to pass data around
- Feels very "light"; not much boilerplate
- Amazing tooling

Cons

- Some framework-specific conventions
- JSX can be weird at first
- Very JS heavy



Pros

- Very approachable
- Lots of features built-in
- Single file components
- Amazing tooling

Cons

- Can be difficult to share logic
- Missing solutions for some common issues (i.e. fragments)
- Some conventions can be confusing

Pros

- Very defined conventions with some flexibility
- Very detailed documentation
- Still difficult to learn
- Amazing tooling

Cons

- Observables
- Lots of boilerplate
- Very JS Heavy
- TypeScript

ember

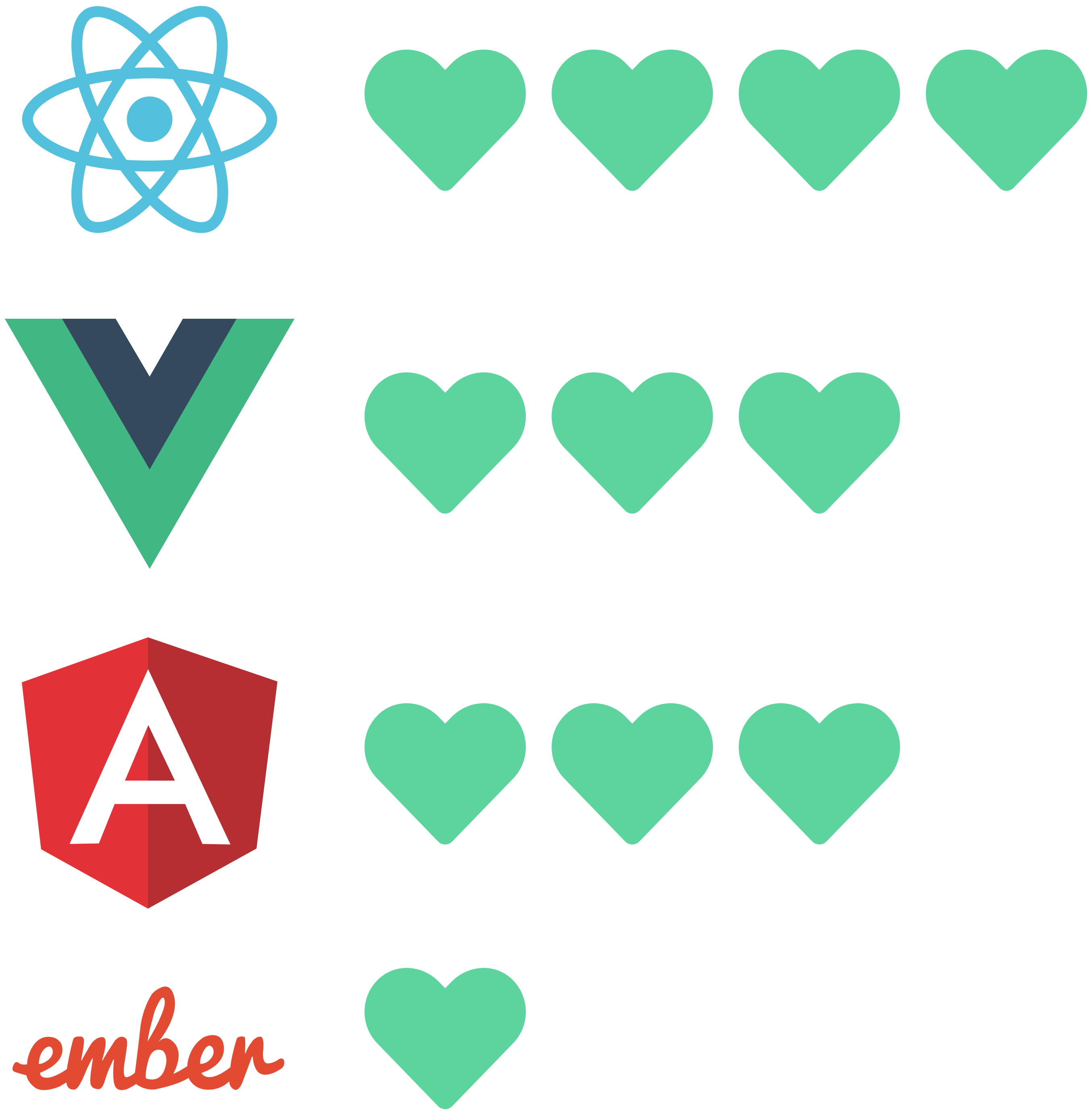
Pros

- Everything has a place
- Helpful community

Cons

- Mixed conventions currently
- Docs out of date
- Difficult paradigms / conventions
- Easy tasks are difficult

Enjoyment



T. HANK