Vue.js:
The Progressive Framework

vuejs.org @vuejs vuejs/vue

Evan You

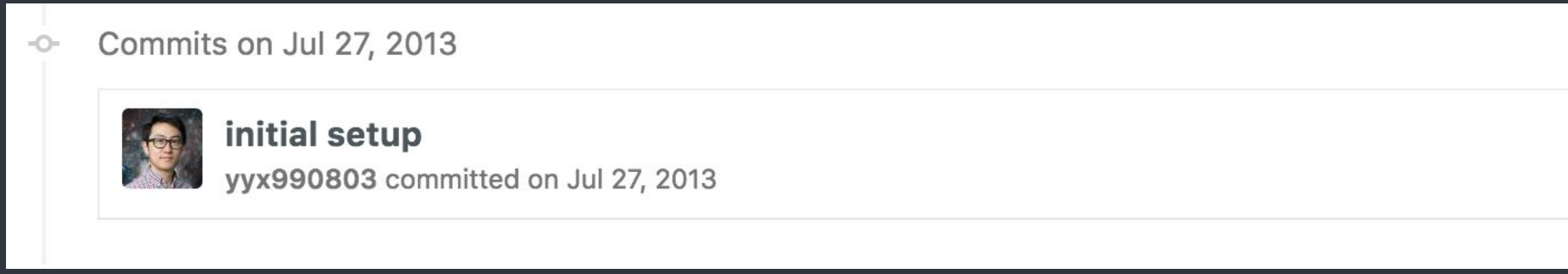
@youyuxi

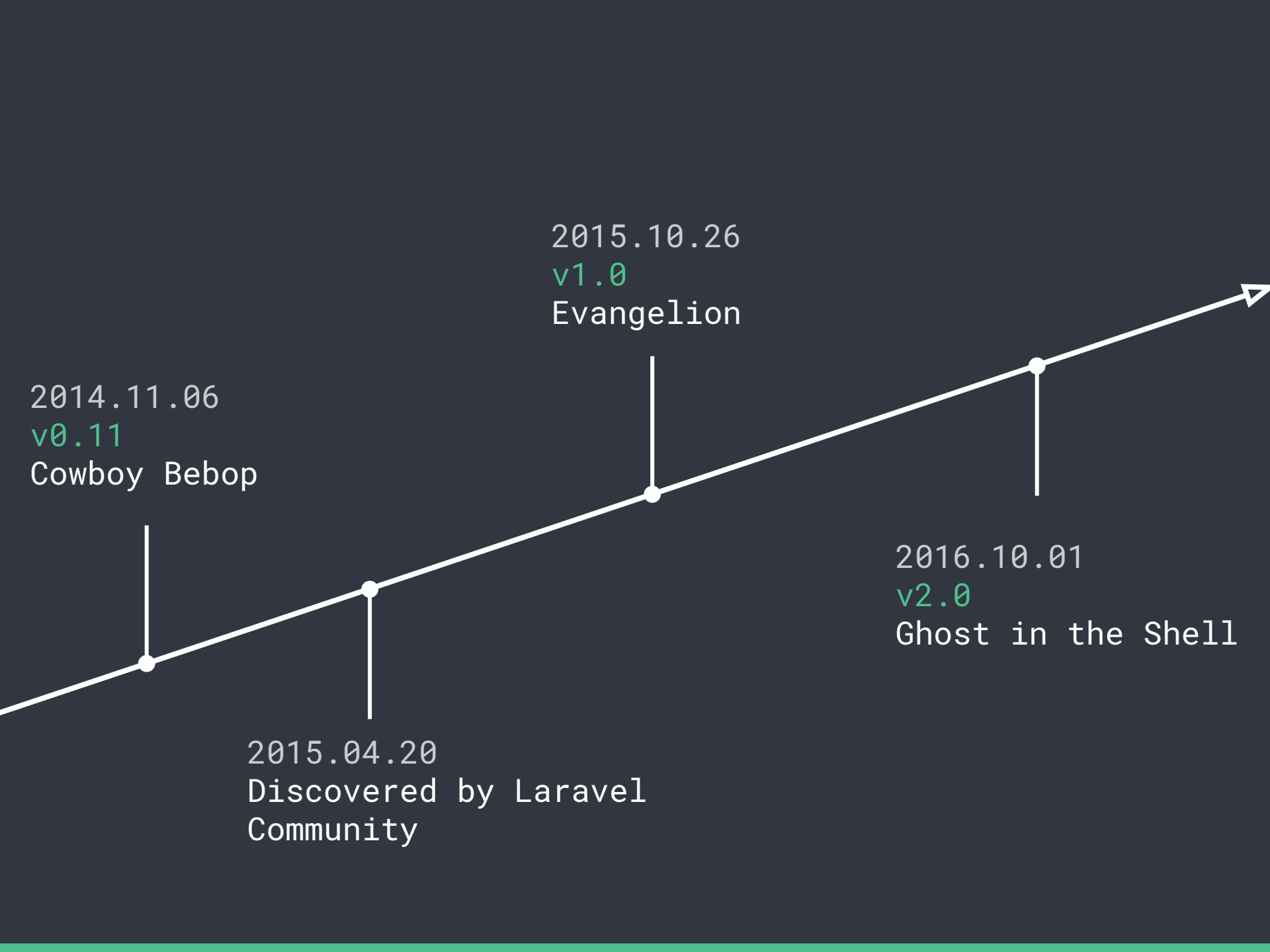
@yyx990803

Currently: full-time open source!
Previously: Meteor, Google Creative Lab

First commit

2013.06.27




2014.11.06
v0.11
Cowboy Bebop
2015.04.20
Discovered by Laravel
Community
2015.10.26
v1.0
Evangelion
2016.10.01
v2.0
Ghost in the Shell

Today

65k+ GitHub Stars

Top 10 All-Time

2nd most-starred JavaScript framework

680k+ monthly NPM downloads

(excluding stats from mirrors in China)

Chrome DevTools Extension
~254k weekly active users

World-wide Commercial Usage

Community

314 GitHub Contributors

across the vuejs organization

Thriving Community Projects

Quasar Framework

iView

Muse-UI

Vux

Vuetify

Vue Material

Evolution

“Just a view layer library”

~~“Just a view layer library”~~

The Progressive Framework

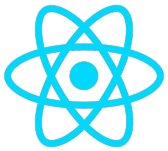
Application Complexity
vs.
Framework Complexity

The Framework Spectrum

Less

More

Templating Engines



React

Vue

Backbone

Angular

Ember

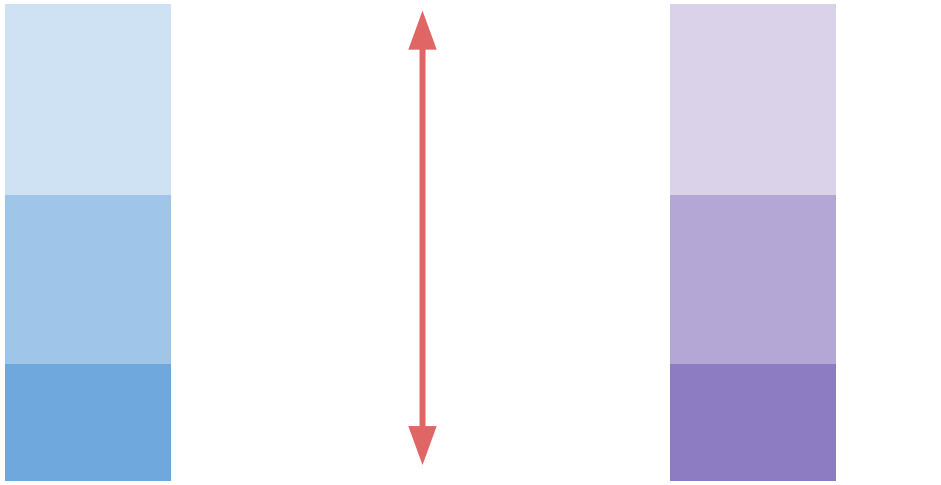
Meteor

Less

More

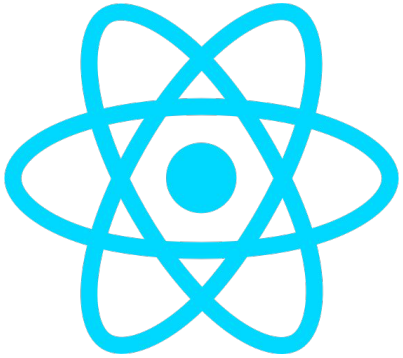

With supporting libraries
React+
Vue+
Templating Engines
React
Vue
Backbone
Angular
ember
Ember
METEOR
Meteor
Less
More

Scalability is Not a One-Way Street

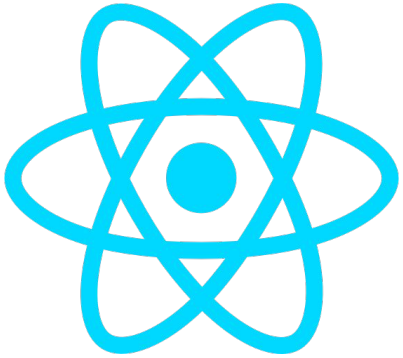


Inherent Complexity

Instrumental Complexity



View Layer Core
+
Optional Support Libraries



View Layer Core

+

Optional Support Libraries?

Eric Clemmons Follow
Creator of React Resolver, Genesis/Evolution for WordPress. Purveyor of a better Developer Experience…
Dec 26, 2015 · 4 min read

Javascript Fatigue

A few days ago, I met up with a friend & peer over coffee.

> *Saul: "How's it going?"*
>
> *Me: "Fatigued."*
>
> *Saul: "Family?"*
>
> *Me: "No, Javascript."*

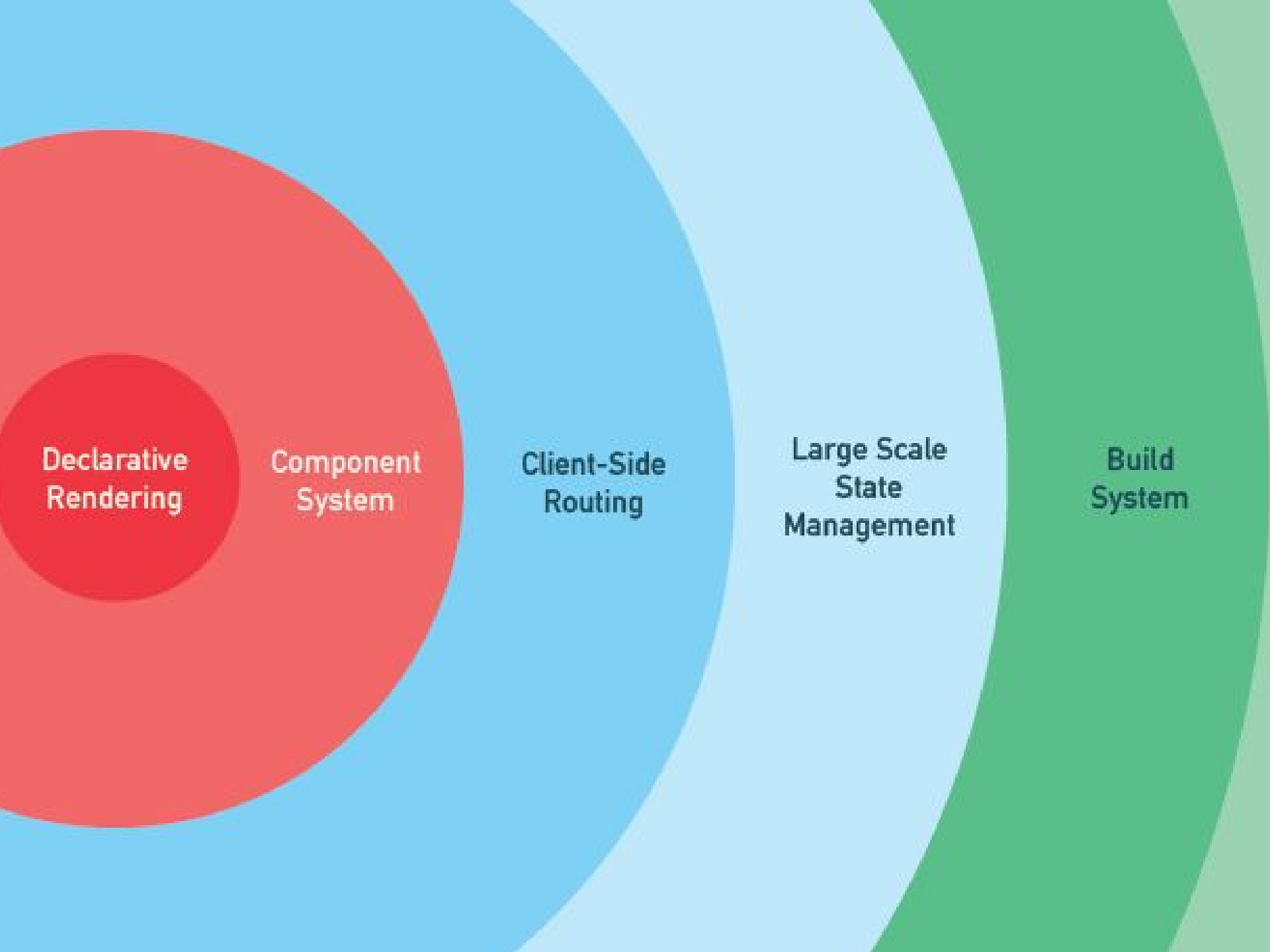
The Progressive Framework

pro·gres·sive

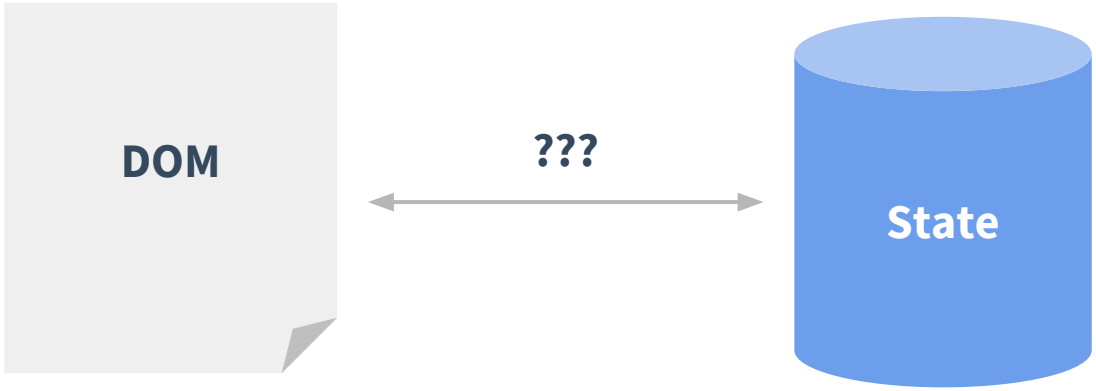
/prəˈgresiv/

adjective

1. happening or developing gradually or in stages; proceeding step by step.
   "a progressive decline in popularity"
   synonyms: continuing, continuous, increasing, growing, developing, ongoing, accelerating, escalating; More


Declarative Rendering
Component System
Client-Side Routing
Large Scale State Management
Build System

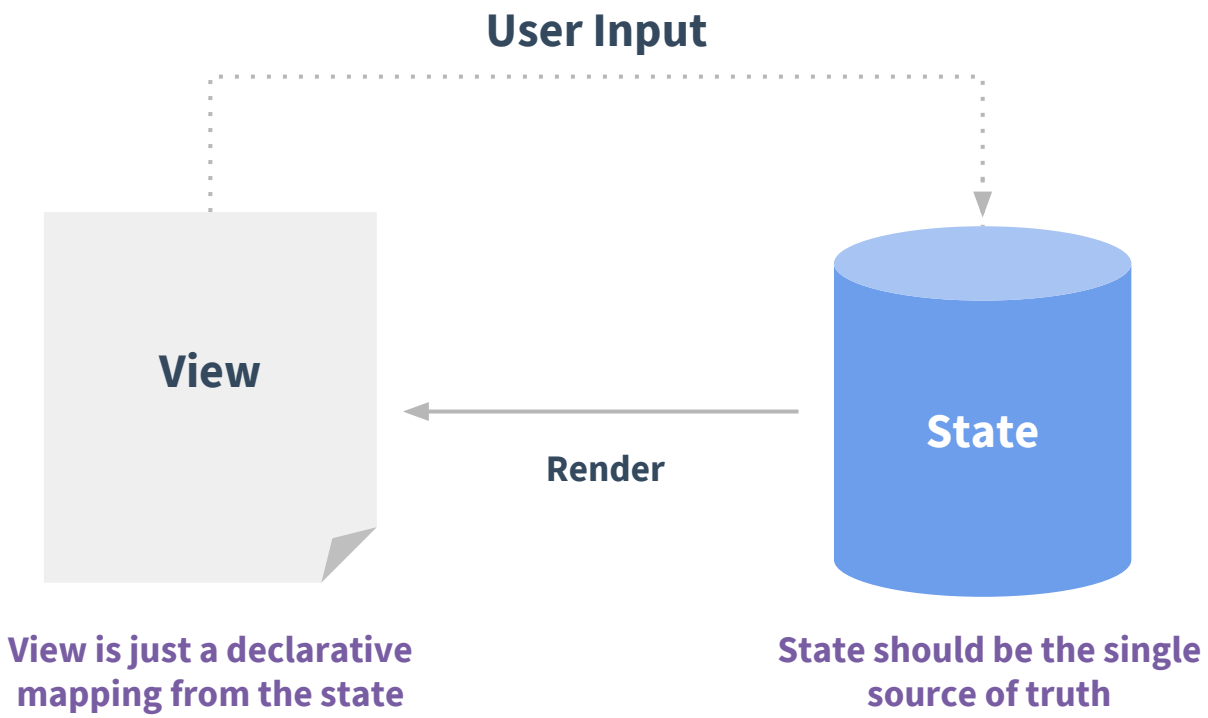
Declarative Rendering


DOM
???
State

Problems with the DOM

- Re-rendering entire chunks of DOM is expensive and disruptive
- Imperatively keeping the DOM in sync with the state is tedious and error-prone

Declarative & Reactive Rendering



No-build-step onboarding:
Hello World in a plain HTML file

```html
<script src="https://unpkg.com/vue/dist/vue.js"></script>

<div id="app">
  <p>{{ message }}</p>
</div>
```

```
new Vue({
  el: '#app',
  data: {
    message: 'Hello Vue.js!'
  }
})
```

Templates

vs.

JSX

vs.

Render Functions

“JavaScript in HTML”

vs.

“HTML in JavaScript”

vs.

“Just JavaScript”

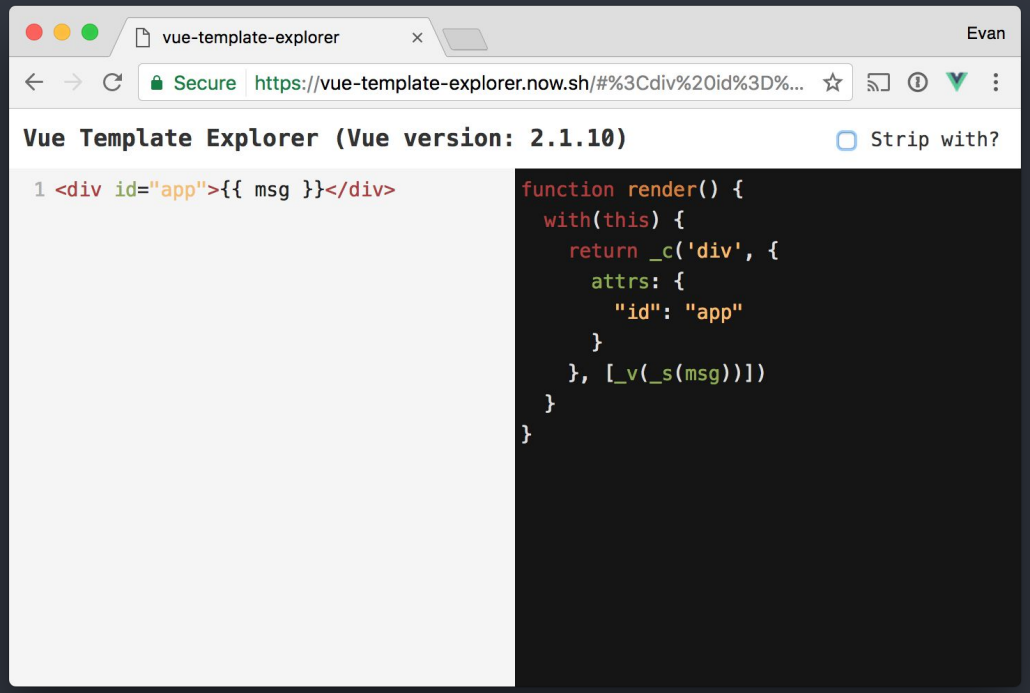
Template vs. JSX

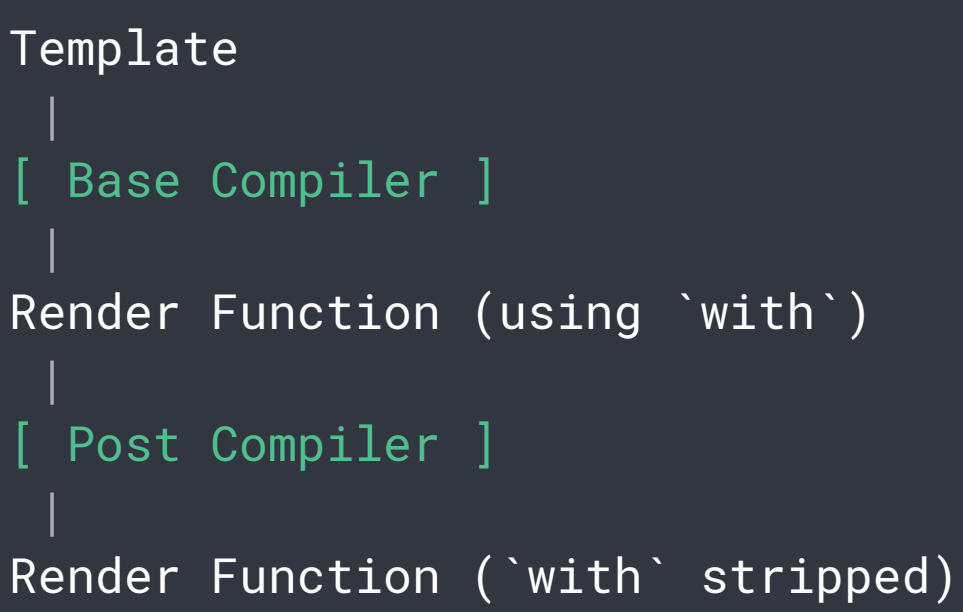
- Both are ways to declaratively map state to desired render output
- Their pros and cons are actually complementary to each other
- Developer background and mindset affect our effectiveness with a certain programming model.

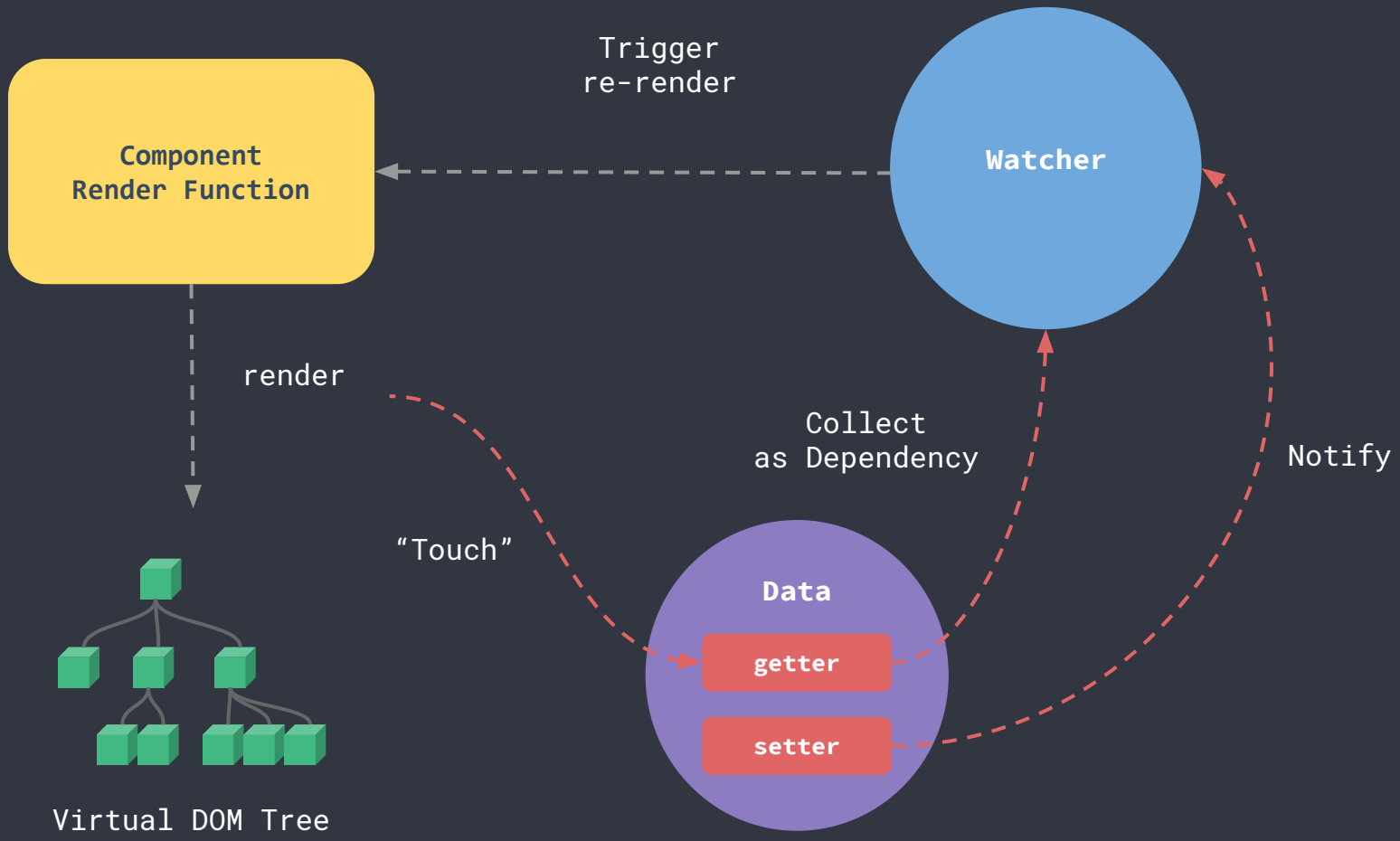
This is why Vue 2 supports both.

Template -> [Parser] -> AST -> [Codegen] -> Render Function

vue-template-explorer.now.sh

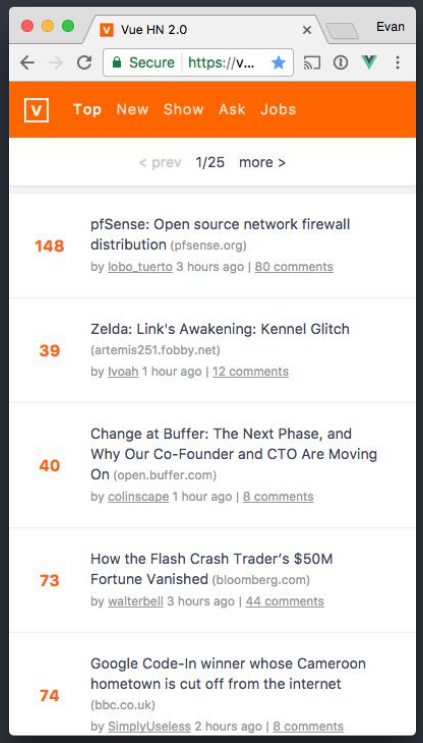



Template
[Base Compiler]
Render Function (using `with`)
[Post Compiler]
Render Function (`with` stripped)

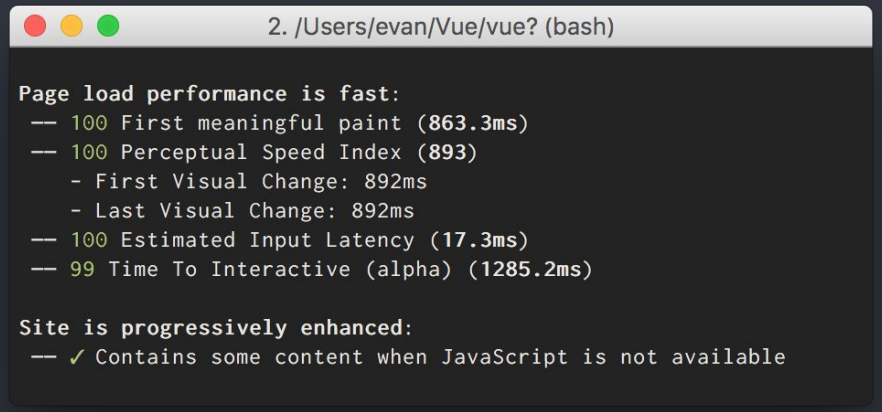

Component
Render Function
Trigger
re-render
Watcher
render
Collect
as Dependency
Notify
“Touch”
Data
getter
setter
Virtual DOM Tree

Server-Side Rendering

[vue-hn.now.sh](vue-hn.now.sh)

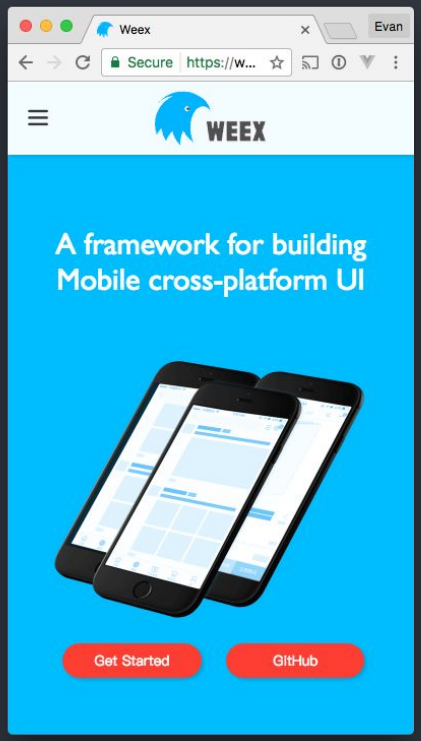


Lighthouse audit results

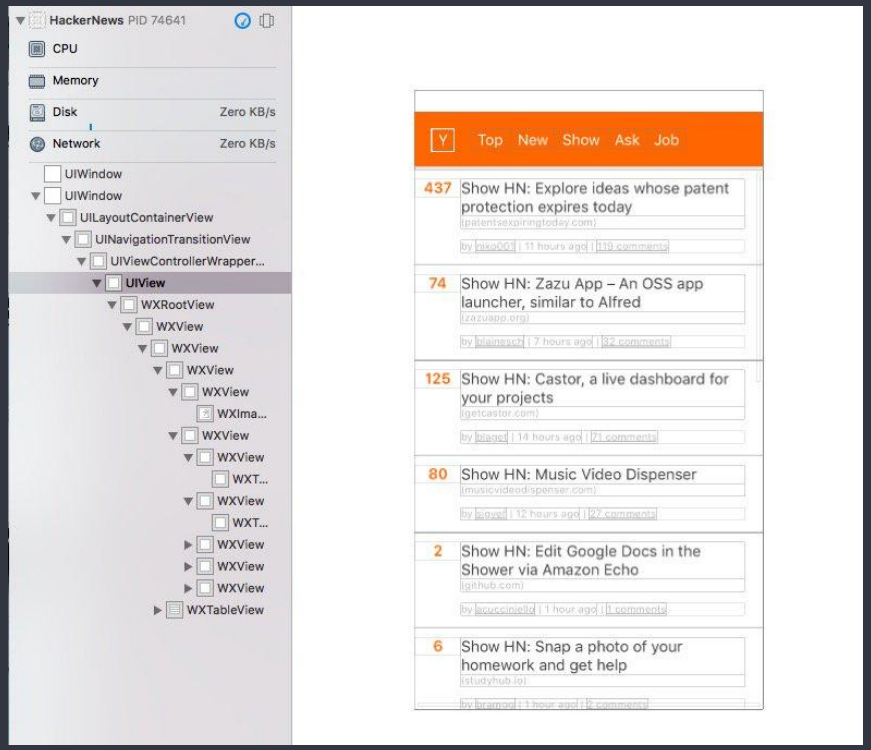


Native Rendering via Weex Project

weex-project.io



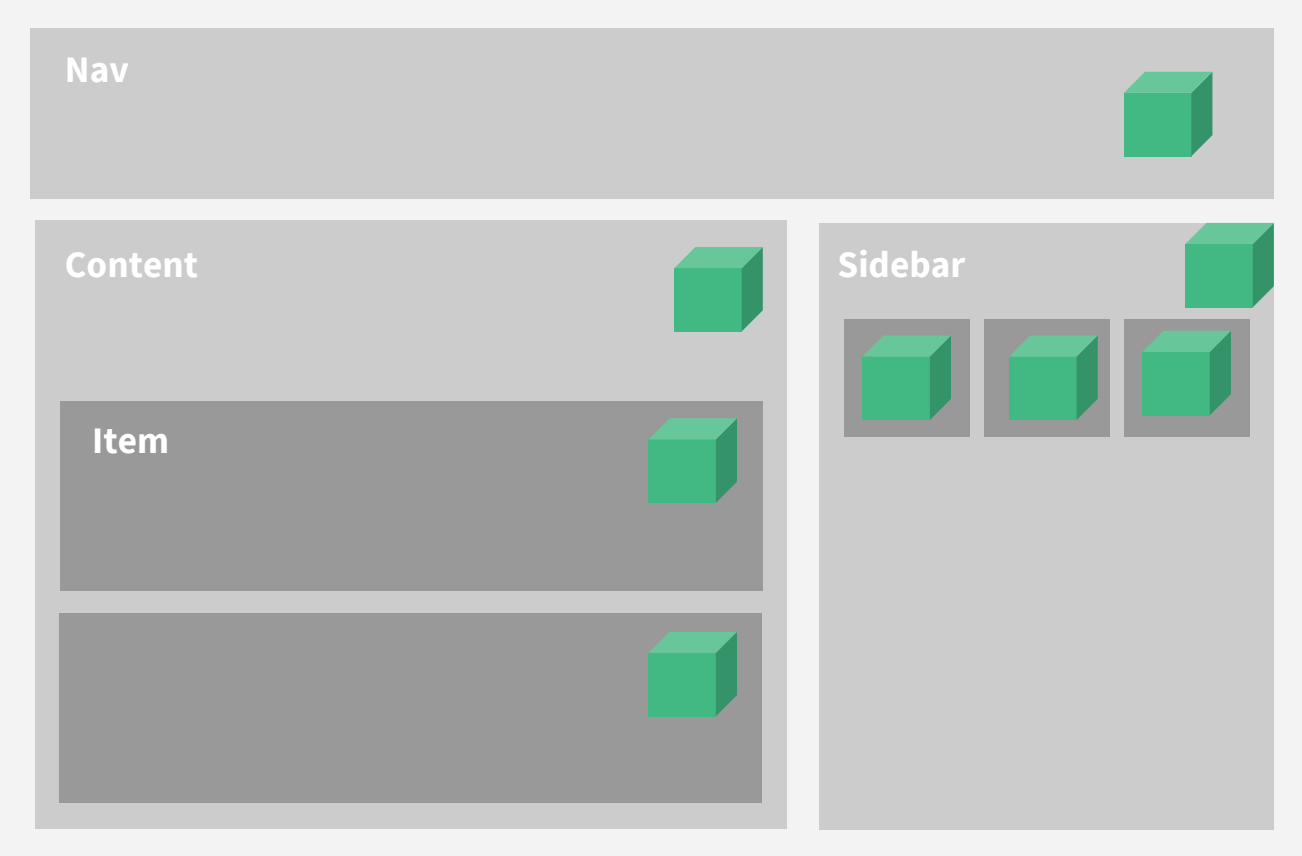
HN demo implemented with Weex



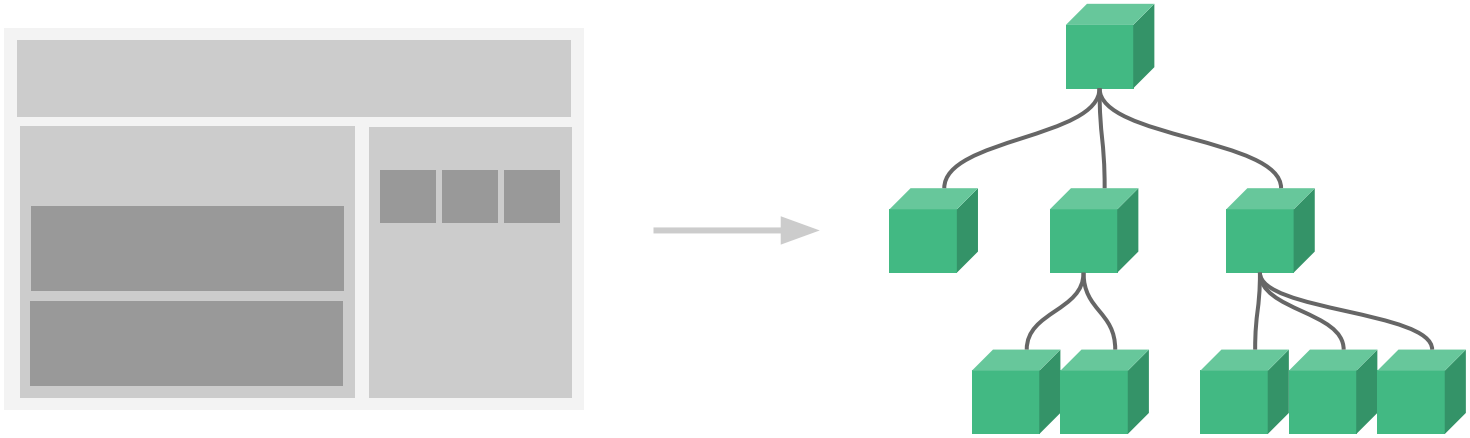
Component System

Most App UIs can be broken down into components

Every component is responsible for
managing a piece of DOM

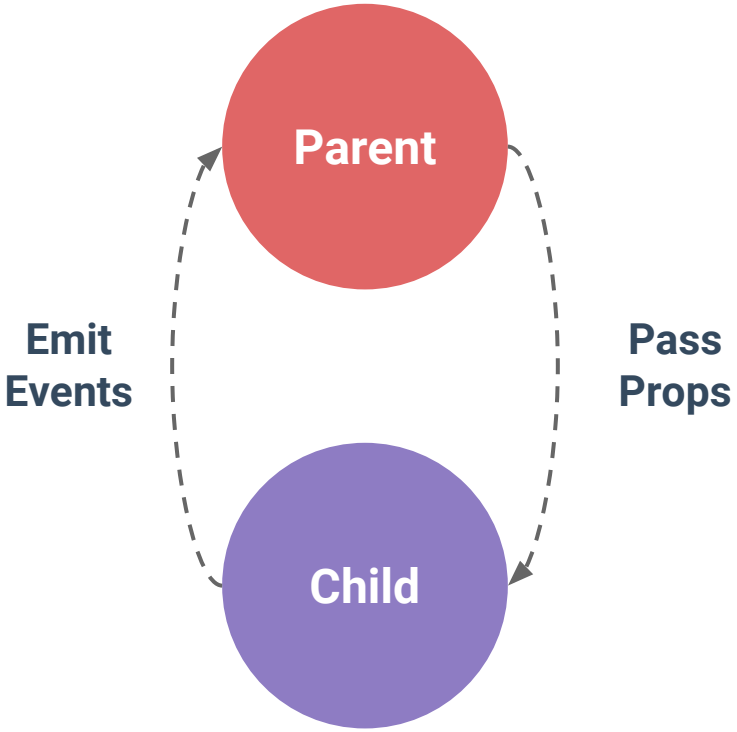


The entire UI can be abstracted
into a tree of components

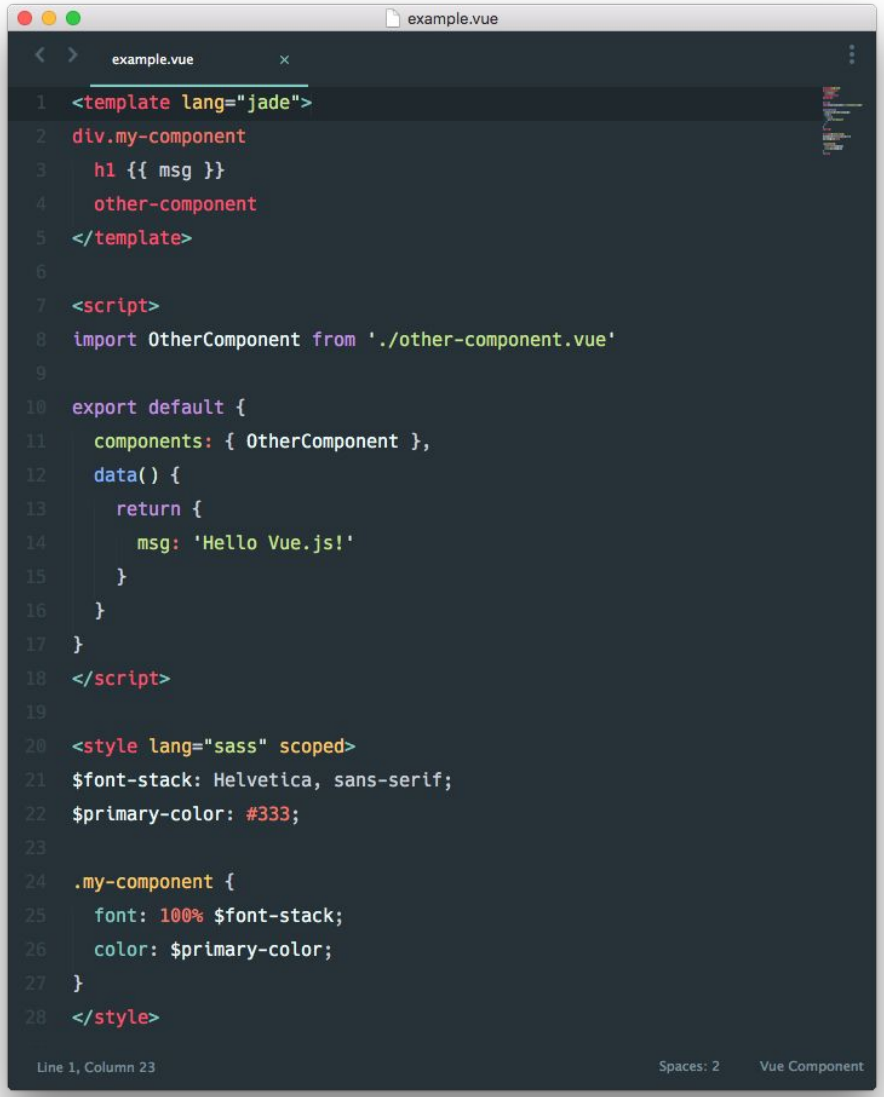


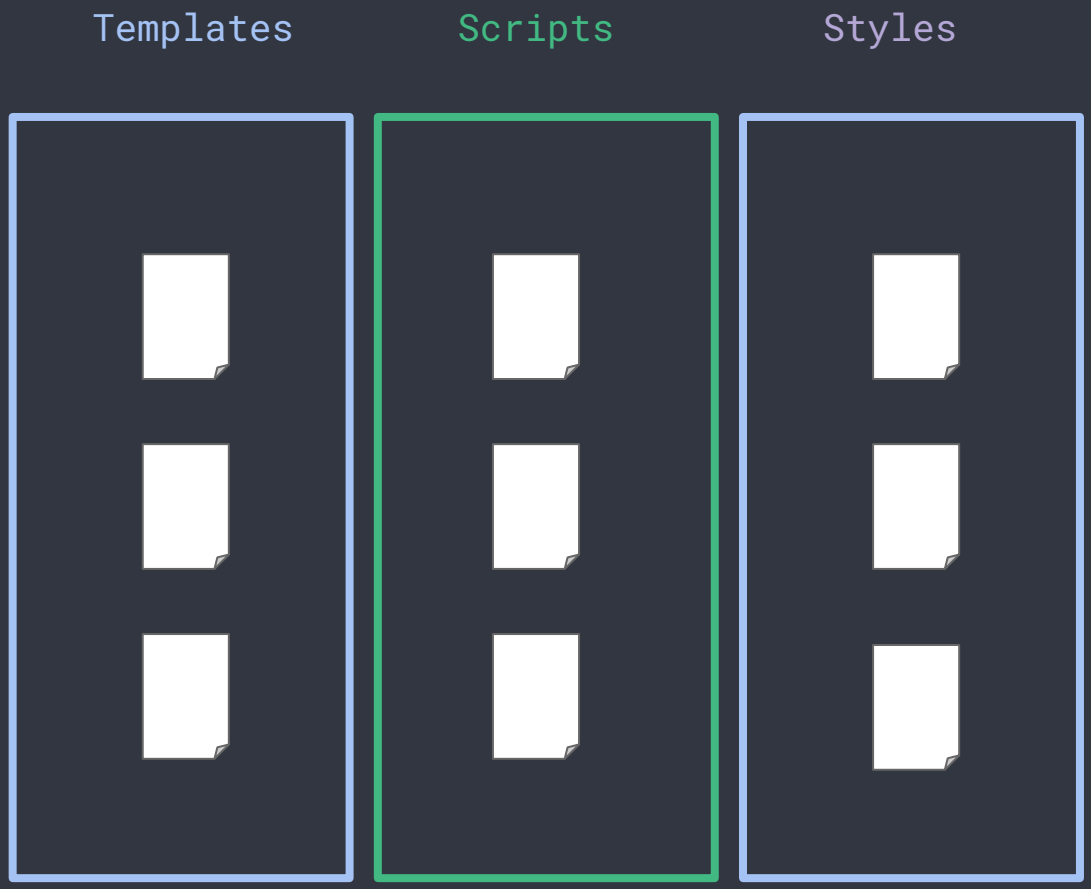
Component Communication:

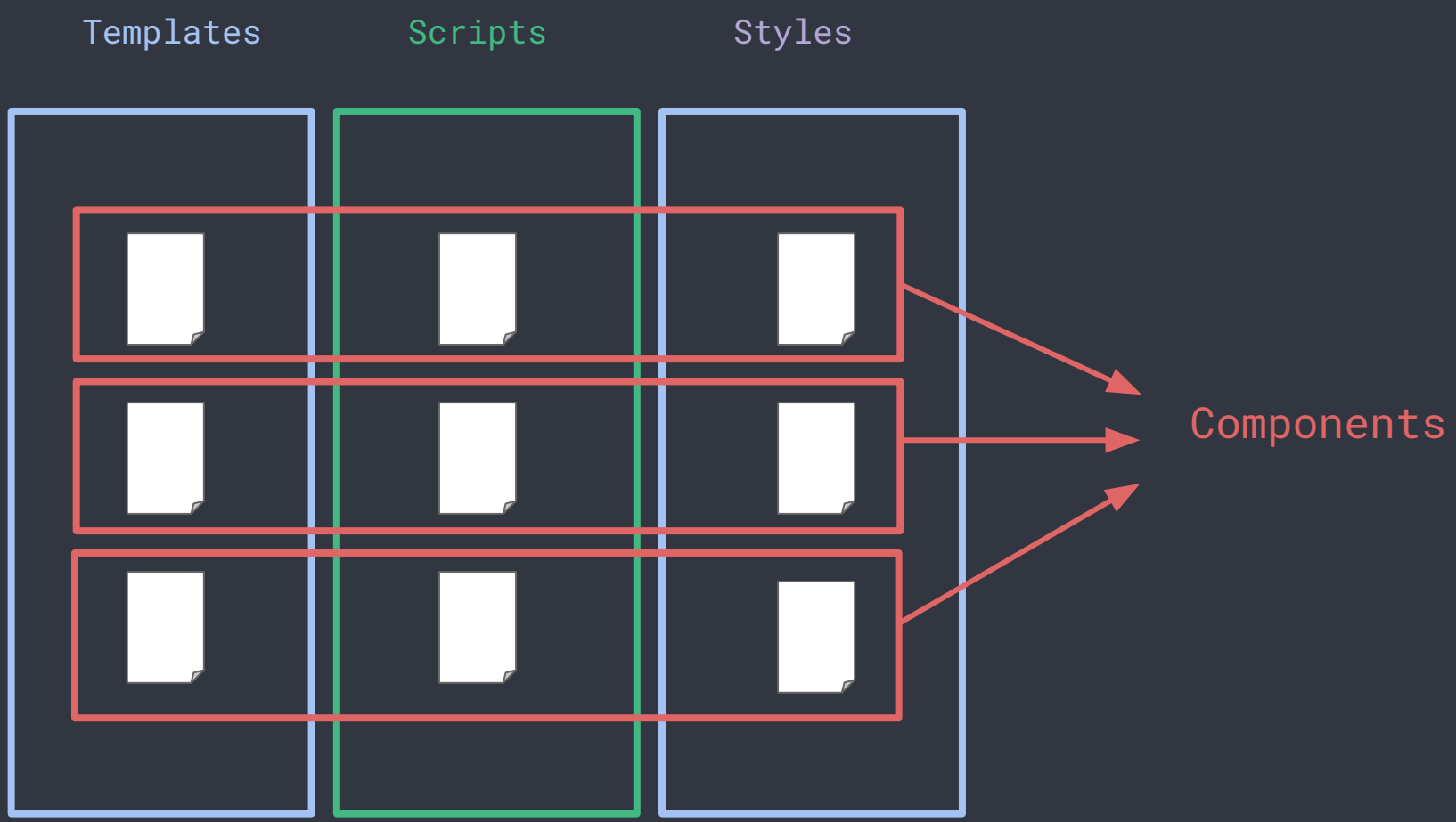
Props in, Events out

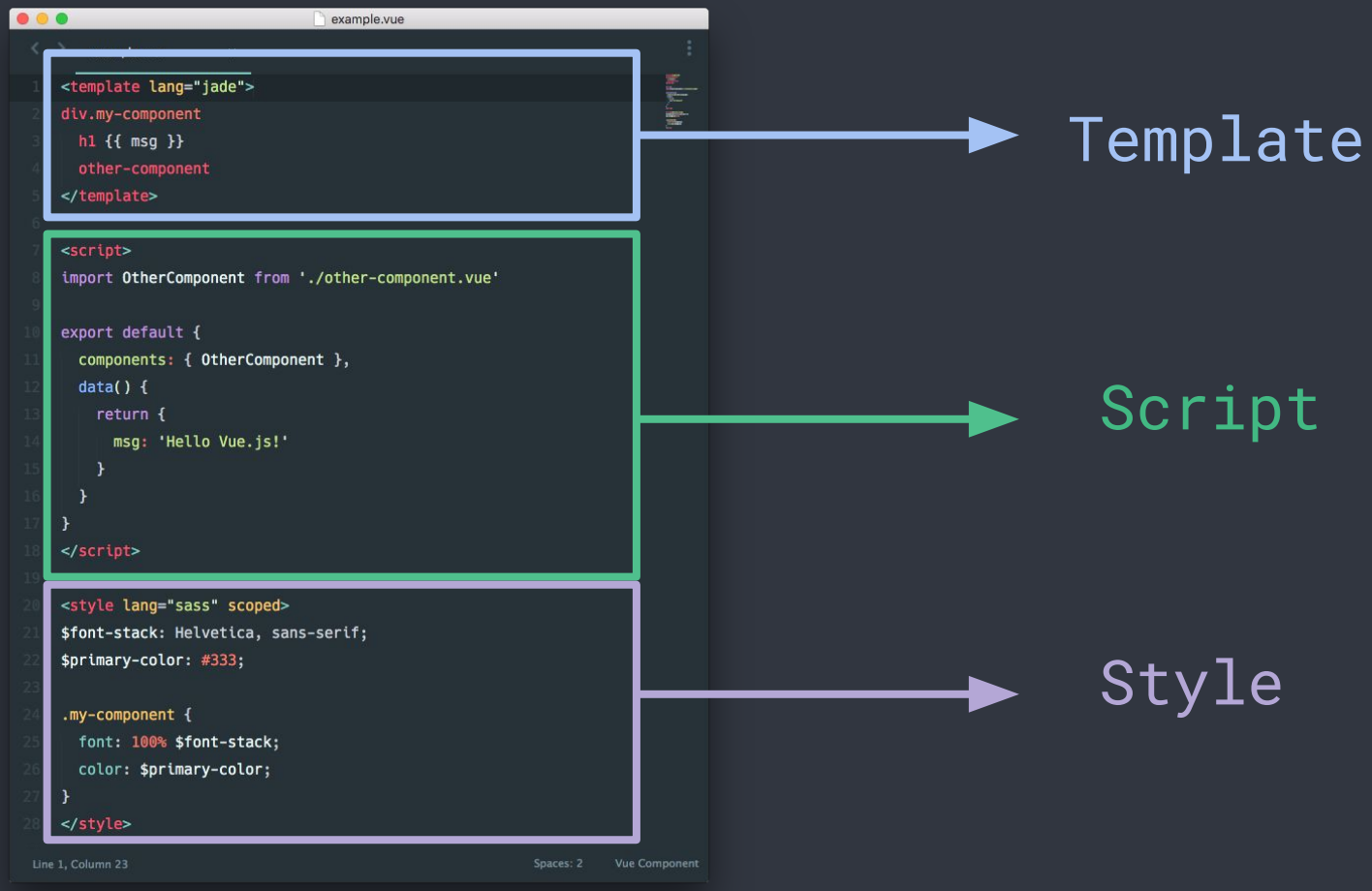


Single File Vue Components



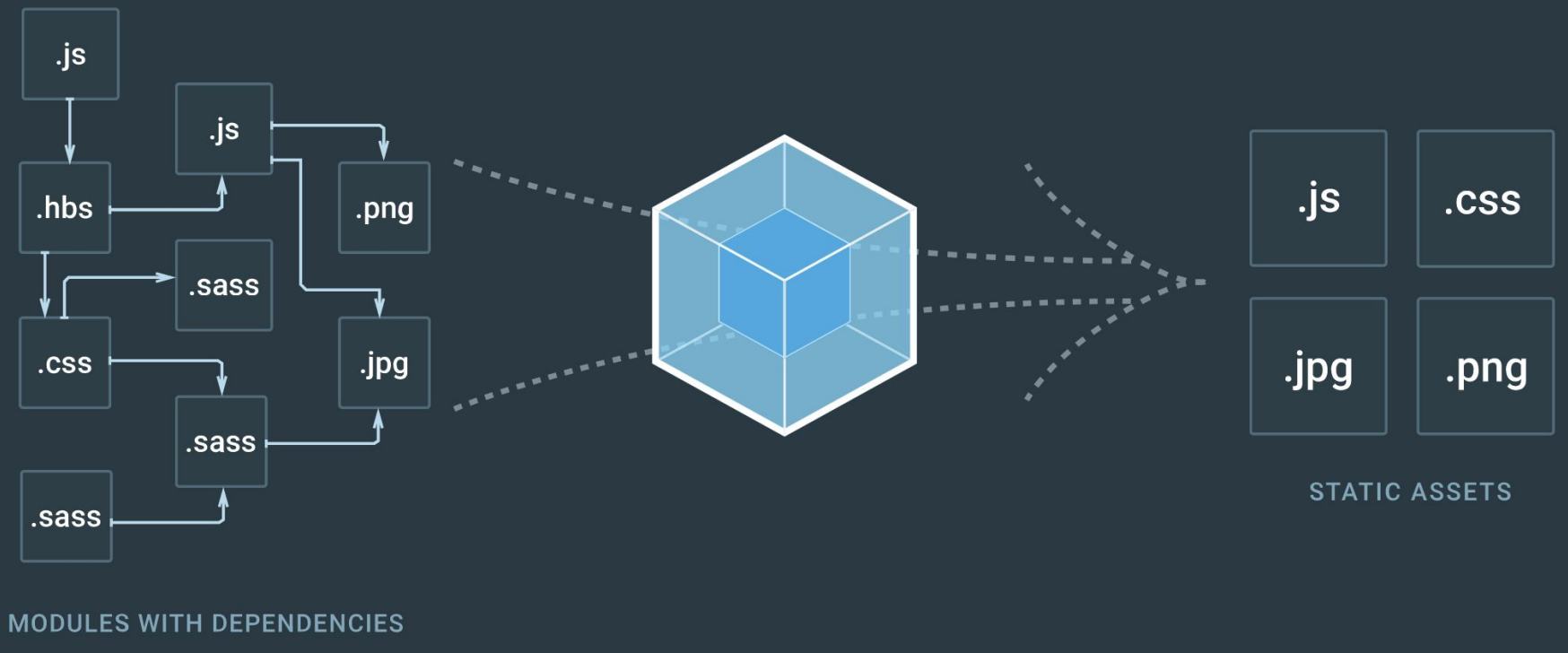

Templates
Scripts
Styles

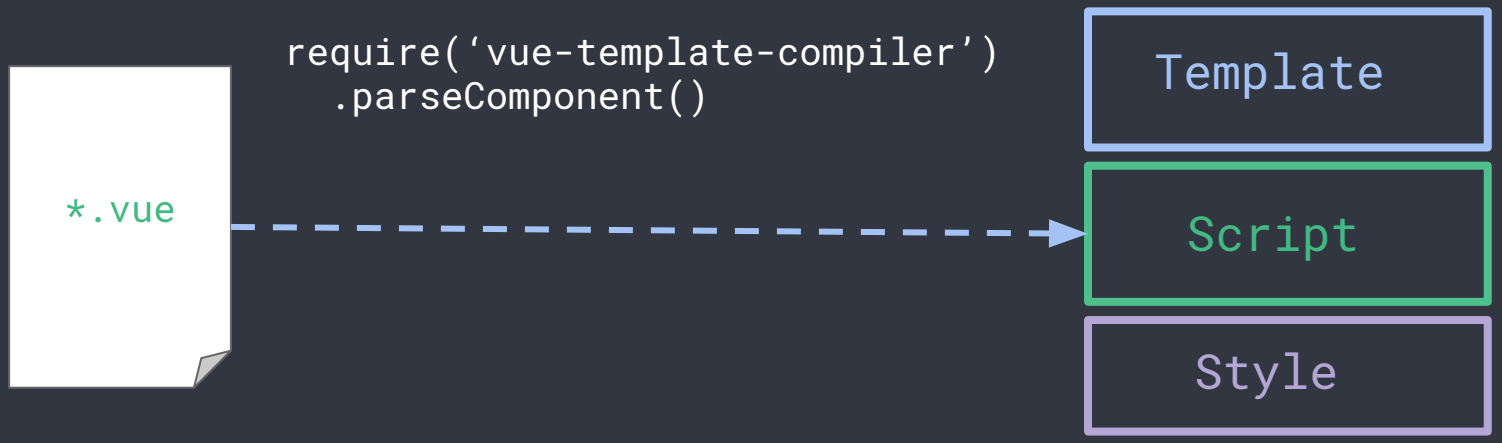

Templates
Scripts
Styles
Components

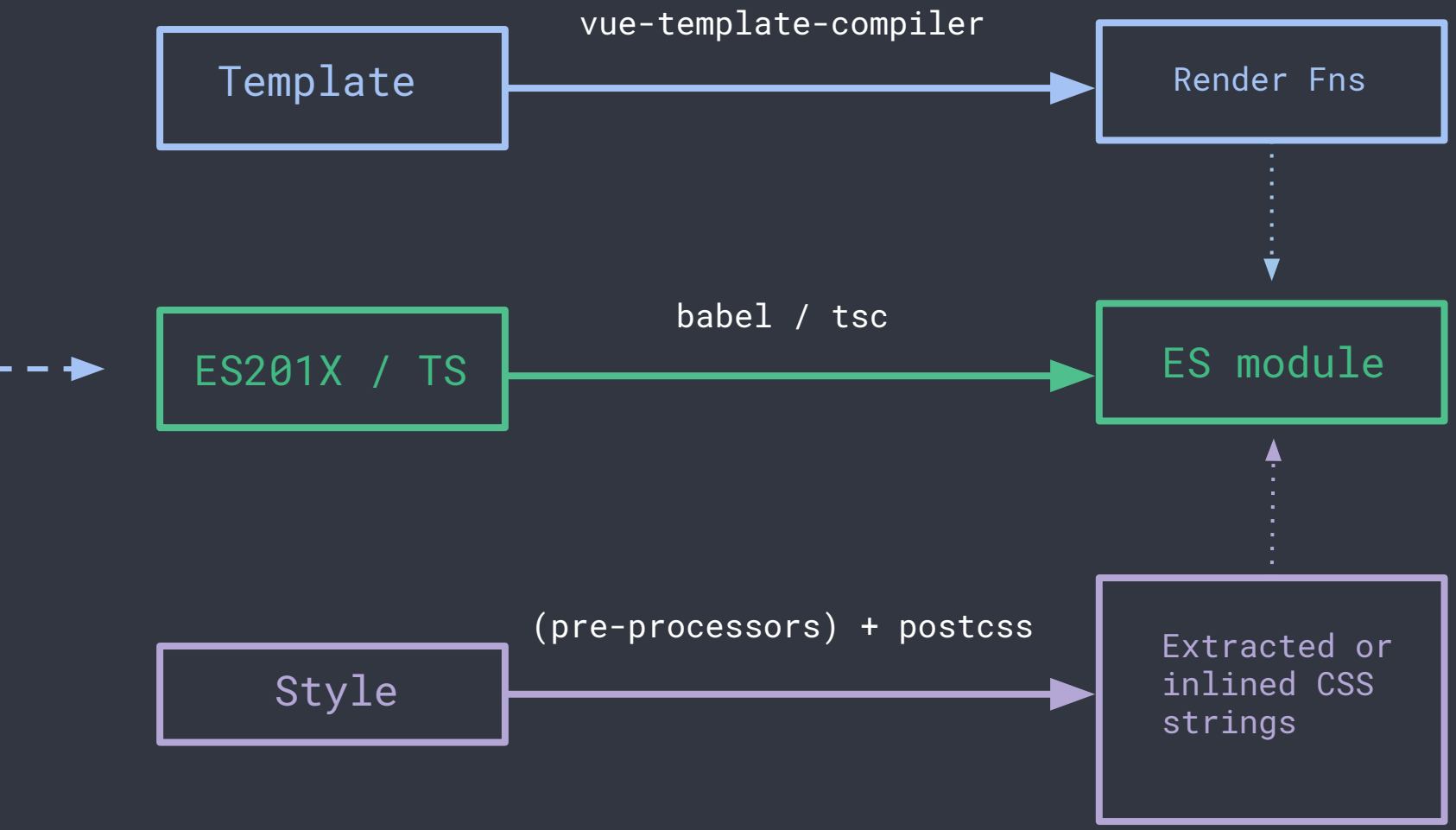

example.vue
<template lang="jade">
div.my-component
h1 {{ msg }}
other-component
</template>
<script>
import OtherComponent from './other-component.vue'
export default {
components: { OtherComponent },
data() {
return {
msg: 'Hello Vue.js!'
}
}
}
</script>
<style lang="sass" scoped>
$font-stack: Helvetica, sans-serif;
$primary-color: #333;
.my-component {
font: 100% $font-stack;
color: $primary-color;
}
</style>
Line 1, Column 23
Spaces: 2
Vue Component
Template
Script
Style

Collocation is awesome!

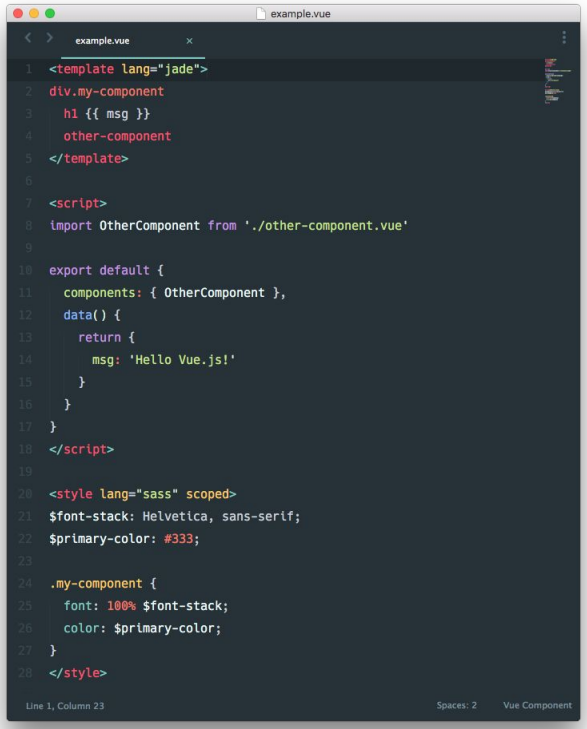
webpack & vue-loader




require('vue-template-compiler')
.parseComponent()
*.vue
Template
Script
Style


Template
vue-template-compiler
Render Fns
ES201X / TS
babel / tsc
ES module
Style
(pre-processors) + postcss
Extracted or inlined CSS strings

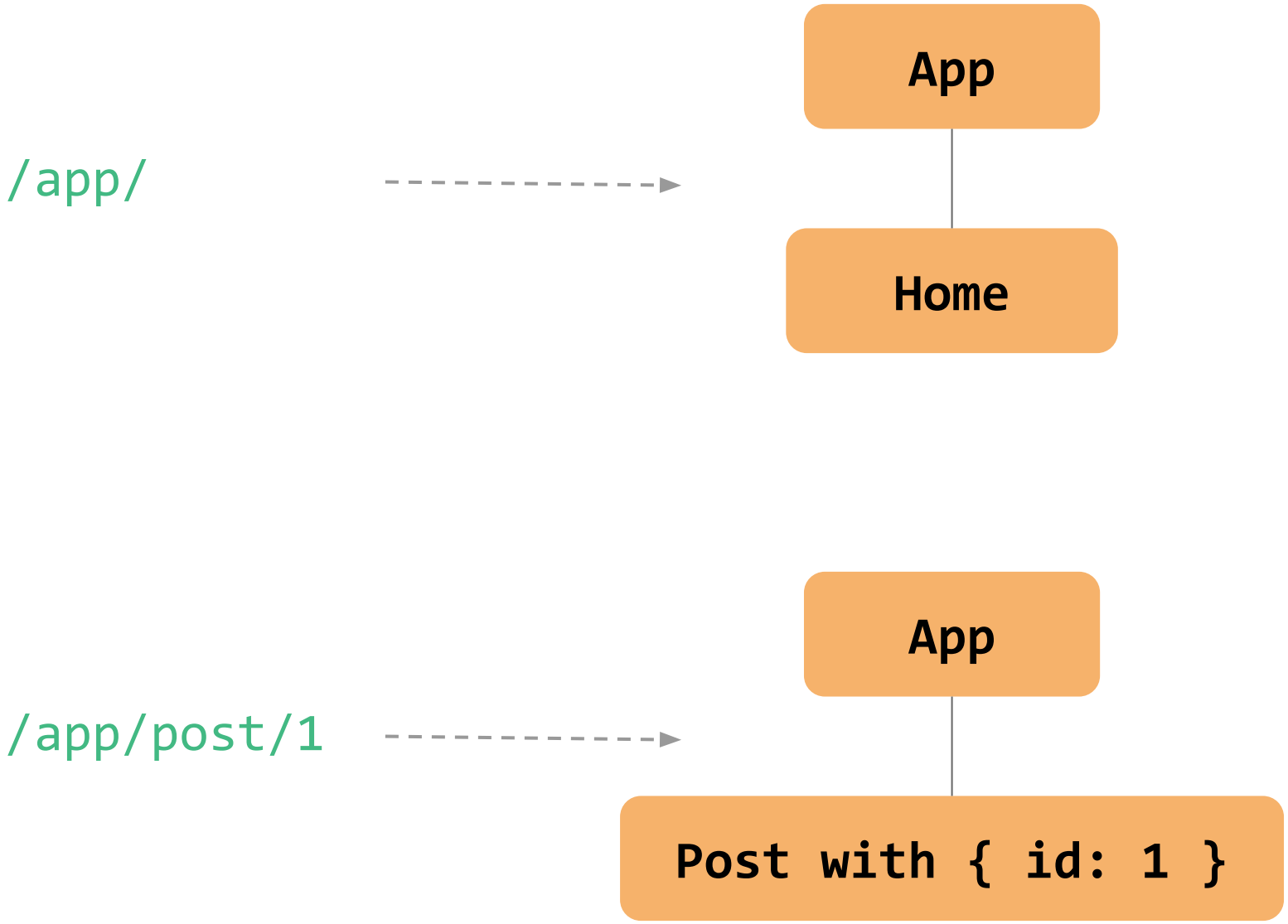
Single File Vue Components



- Collocation of Template, Logic & Style
- Just use what you already know: HTML, CSS & JavaScript
- Imported as a ES2015 module (thus easily testable)
- Embedded pre-processor support: seamlessly use Babel, SASS or even Pug in the same file (as long as there's a webpack loader for it)
- Hot-reload out of the box
- Component-scoped CSS with a single attribute
- First class support for CSS modules

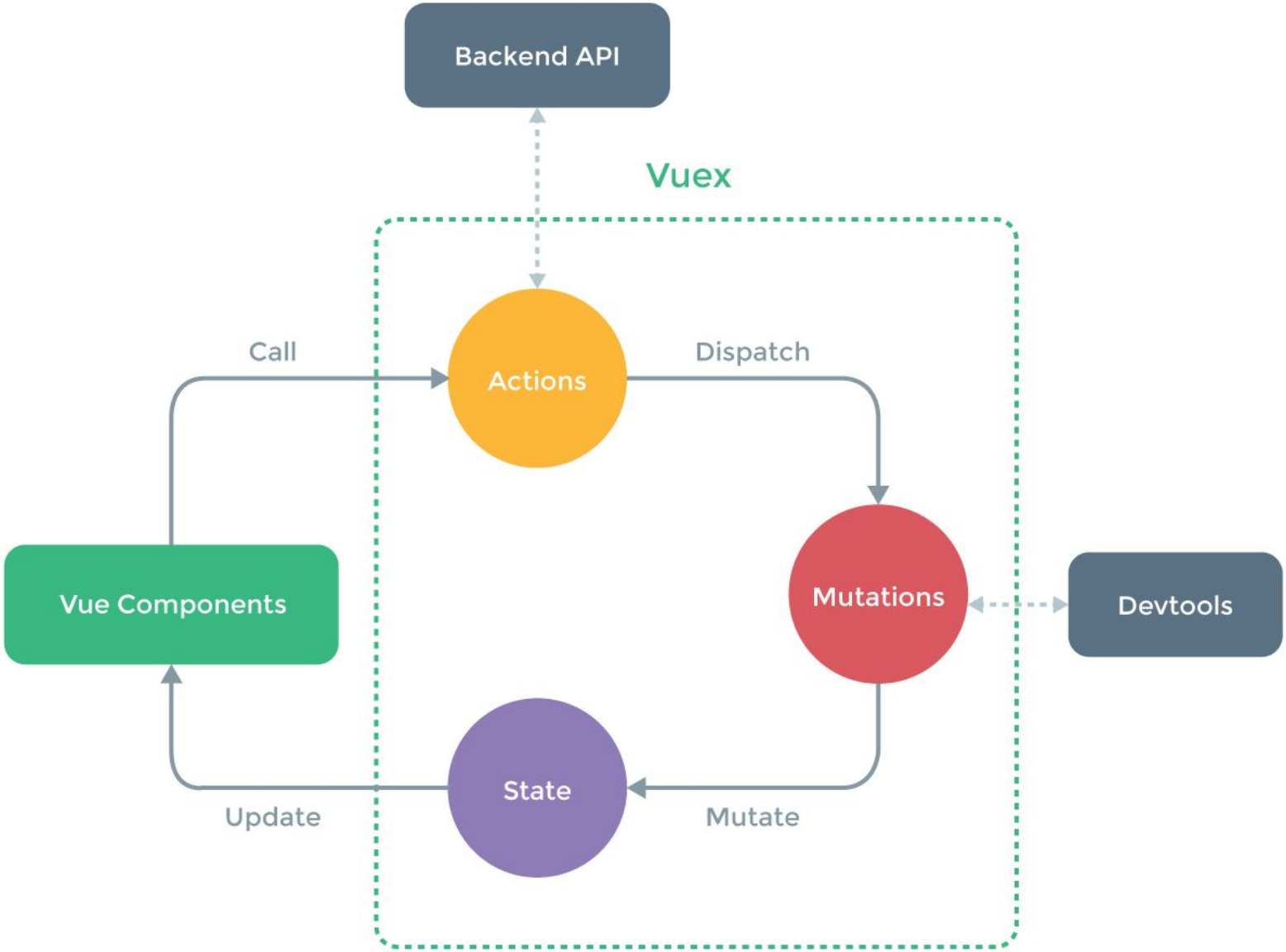
Client-Side Routing

https://github.com/vuejs/vue-router

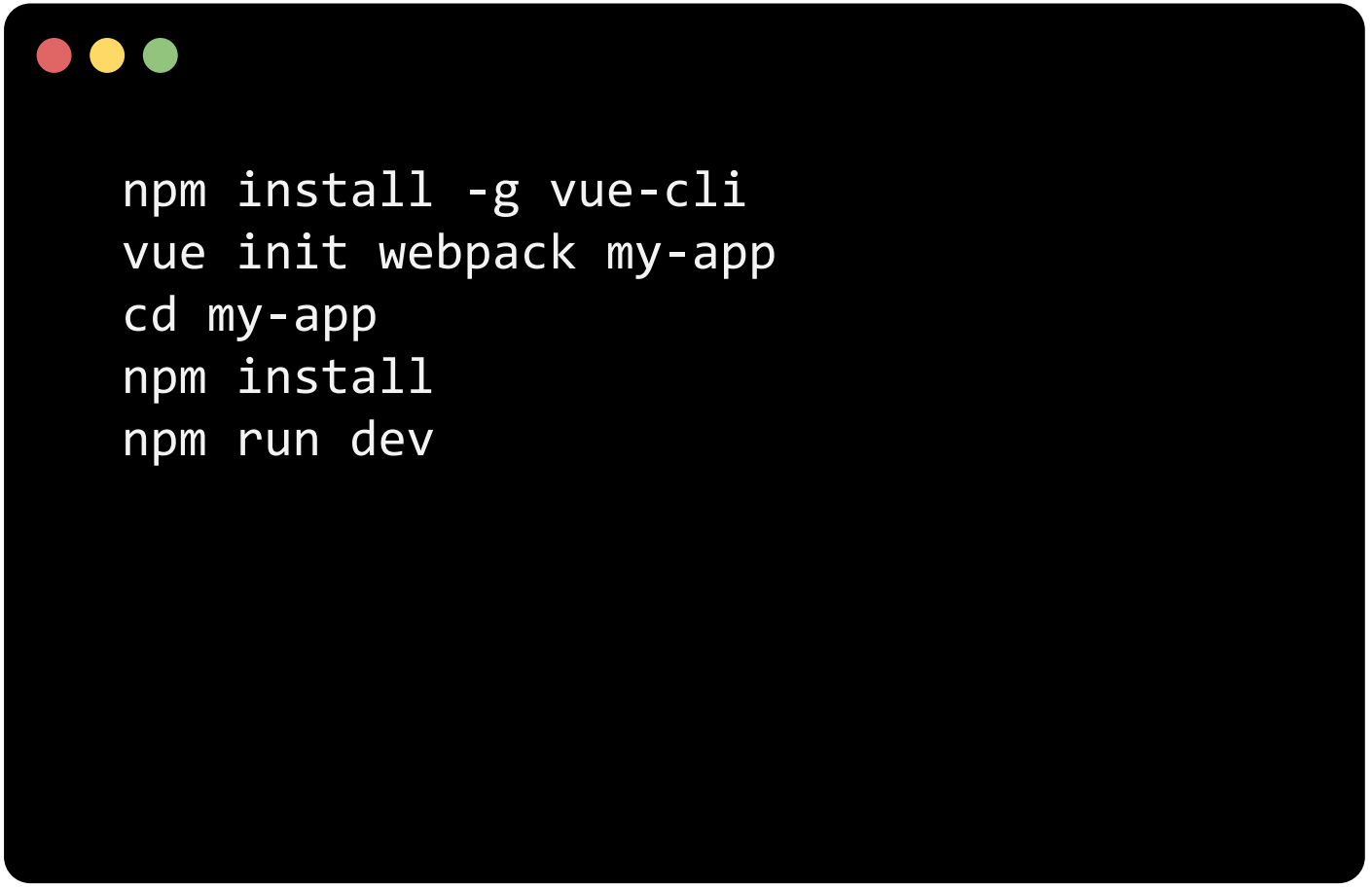

/app/
App
Home
/app/post/1
App
Post with { id: 1 }

Large-Scale State Management

https://github.com/vuejs/vuex

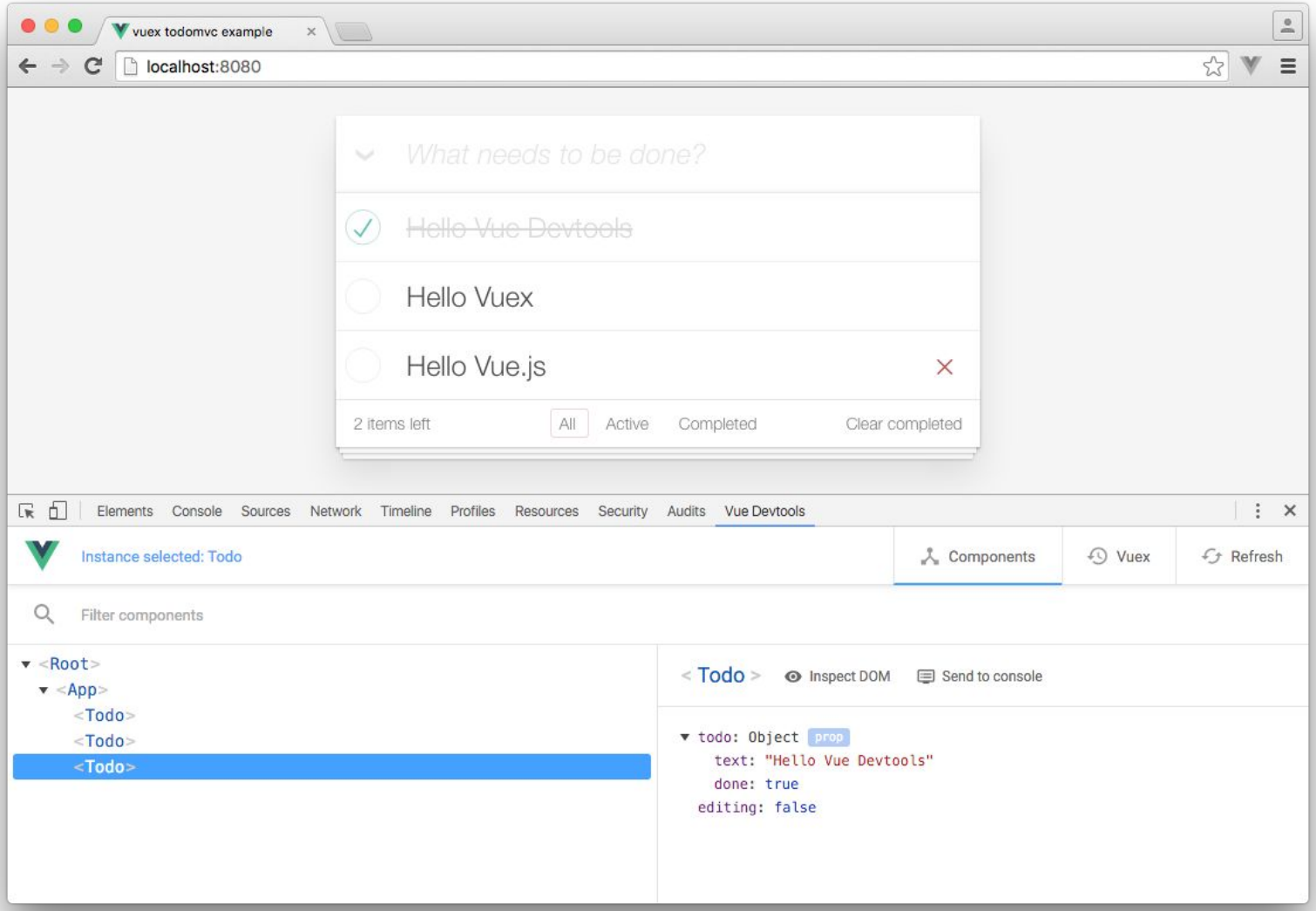

Backend API
Vuex
Call
Actions
Dispatch
Vue Components
Mutations
Devtools
State
Update
Mutate

Build System
/
Development Experience

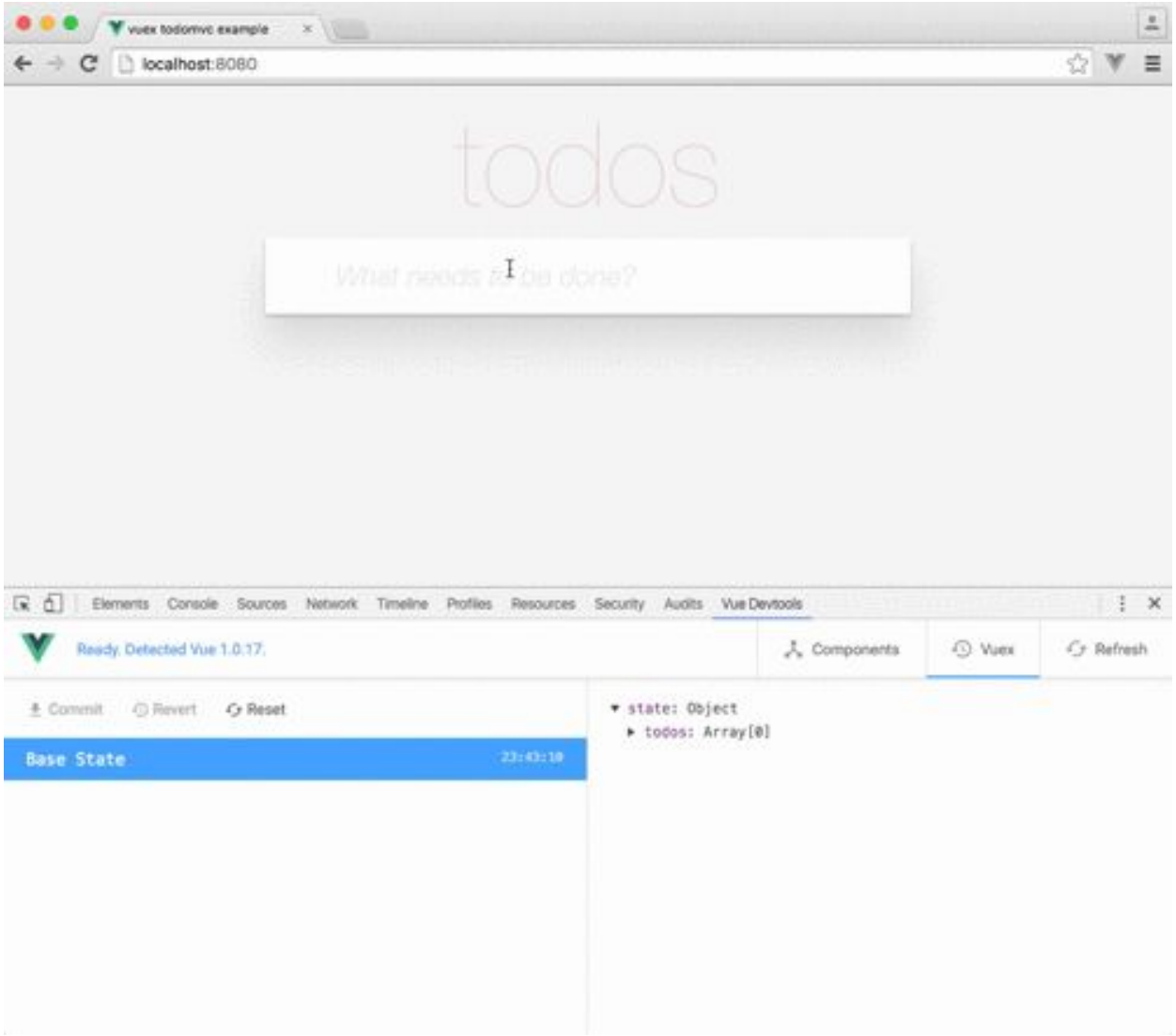


```
npm install -g vue-cli
vue init webpack my-app
cd my-app
npm install
npm run dev
```

Official Chrome DevTools Extension

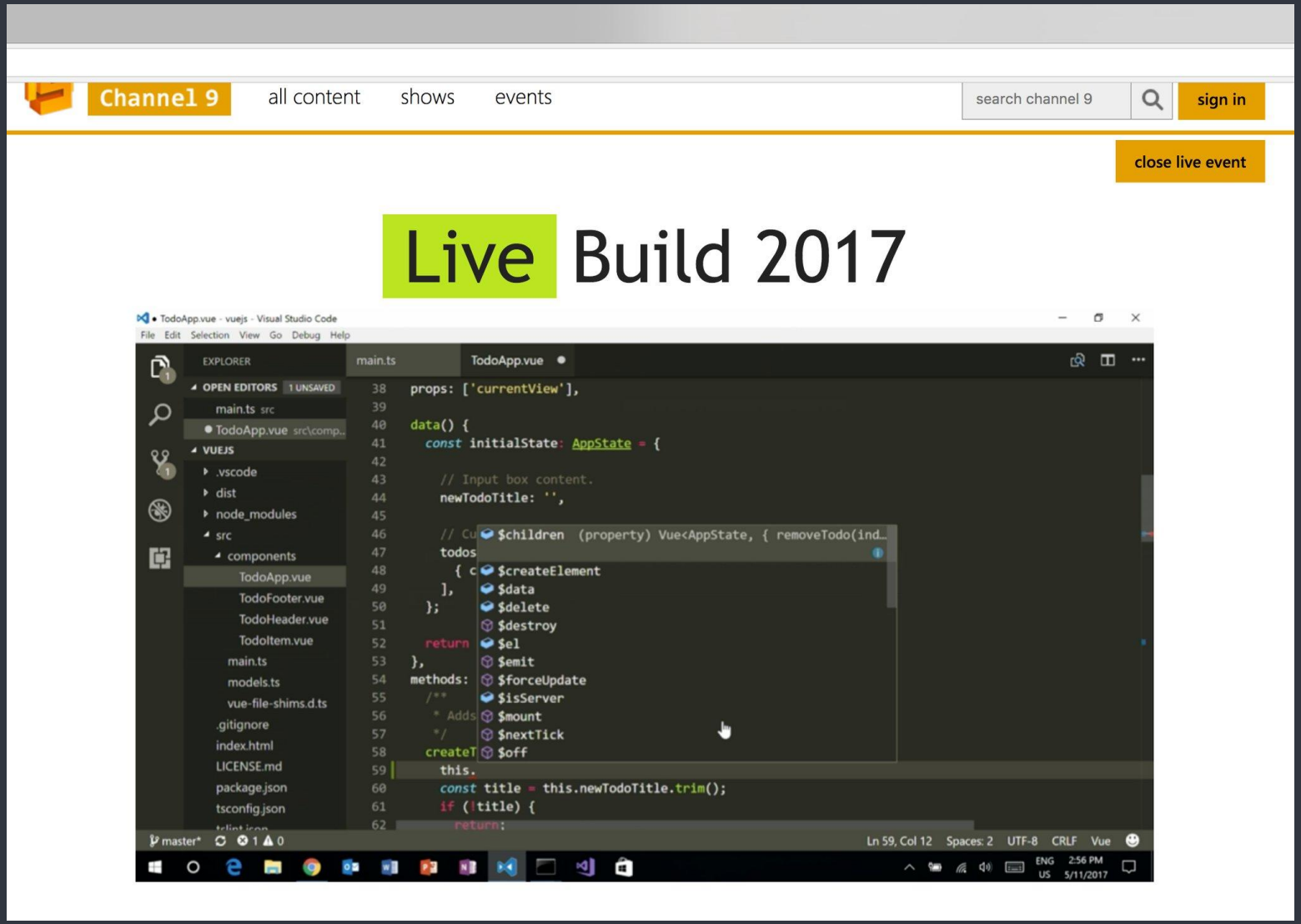


Out of the box time-travel debugging with Vuex



Microsoft Build 2017

Anders Hejlsberg presenting Vue + VSCode + TS 2.3



Thanks!

vuejs.org

@vuejs

vuejs/vue