Birds-Eye Vue of Ember

Gonçalo Morais
@gnclmorais

Hi 👋

- Based in London
- codebar London organiser
- UX Engineer at CrowdStrike
- Recently moved from **Ruby on Rails** + **Vue.js** to **Ember.js**
- Find me
  - twitter.com/gnclmorais
  - blog.gnclmorais.com
  - dev.to/gnclmorais

What is this talk about?

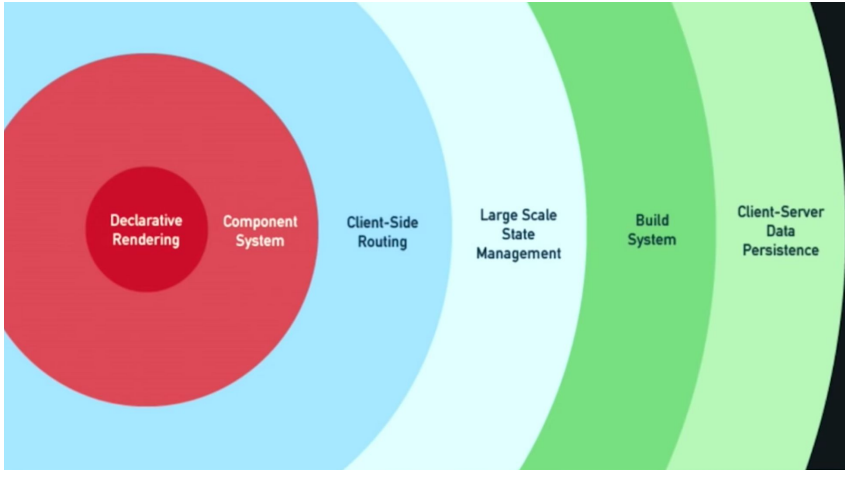
- A brief but useful reference for Vue developers new to Ember
  - Field guide for the most common syntax and features you'd use
  - "I know how to do this in Vue, but how do I do in Ember"
  - "X in Vue looks like Y in Ember"
  - "X in Vue would look like Y in Ember *but* you should do Z instead"
- Focused on Ember Octane and Vue 3

What is this talk not about?

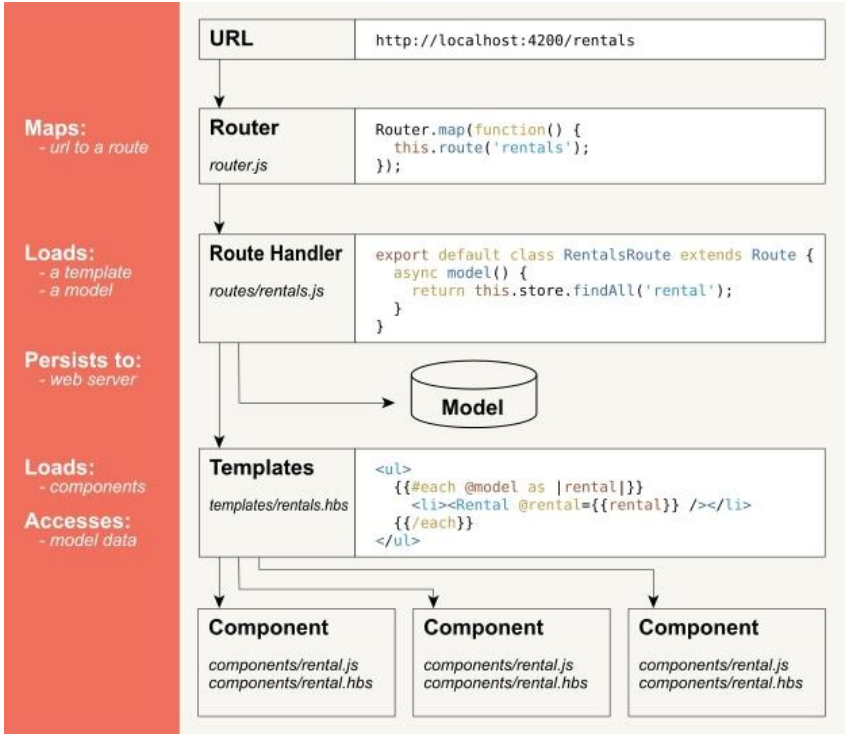
- Not about setting frameworks against each other
  - Nothing to gain from that
- Not about the technical details underlying the differences
  - Not enough time
- Not about favourites
  - Different trade-offs for different projects
- Not an extensive presentation
  - Most useful and popular Vue feature

Core concepts

Vue



Ember



Command Line Interfaces (CLIs)

Vue

- **`npm install -g @vue/cli`**
- `vue-cli` is the old version (Vue 2)
- Although official, it's entirely optional — i.e. you can just get Vue from a CDN and code
- Pick the pieces you want when starting a new project

Ember

- **`npm install -g ember-cli`**
- "is the official way to create, build, test, and serve the files that make up an Ember app or addon."
- "is like a dependency packager, test runner, optimizer, and local server—all rolled into one. (...) The CLI was also built with the idea that a developer should be able to focus on building great apps, not re-engineering how to fit pieces together throughout an app's lifecycle."

Components

Vue

```
// src/main.js
import { createApp } from 'vue'

const app = createApp({
  template: `<button-counter />`
})

app.component('button-counter', {
  data() {
    return { count: 0 }
  },
  template: `
    <button v-on:click="count++">
      You clicked {{ count }} times
    </button>
  `
})

app.mount('#app')
```

Ember

```
{{!-- 1️⃣ app/components/button-counter.hbs --}}
<button {{on "click" this.handleClick}}>
  You clicked {{ this.count }} times
</button>

// 2️⃣ app/components/button-counter.js
import Component from '@glimmer/component'
import { action } from '@ember/object'
import { tracked } from '@glimmer/tracking'

export default class ButtonCounterComponent extends Component {
  @tracked count = 0

  @action
  handleClick() {
    this.count += 1
  }
}

{{!-- 3️⃣ app/about/template.hbs --}}
<ButtonFade />


// ember generate component ButtonCounter -gc
```

Single File Components

Vue

```vue
// src/components/SimpleComponentExample.vue
<template>
  <button v-on:click="count++">
    You clicked {{ count }} times
  </button>
</template>

<script>
  export default {
    data() {
      return { count: 0 }
    }
  }
</script>

<style scoped>
  button {
    color: rebeccapurple;
  }
</style>
```

Ember

```gjs
// app/components/template-import-example.gjs
import { helper } from '@ember/component/helper';
import { modifier } from 'ember-modifier';

const plusOne = helper(([num]) => num + 1);

const setPosition = modifier(
  (element, [position]) => element.scrollTop = position
);

<template>
  <div class="scroll-container" {{setScroll @scrollPos}}>
    {{#each @items as |item index|}}
      Item #{{plusOne index}}: {{item}}
    {{/each}}
  </div>
</template>

// For more details check:
// npmjs.com/package/ember-template-imports
```

Passing data into a component

Vue (props)

```vue
// 1️⃣ src/components/BlogPost.vue
<template>
  <h3>{{ postTitle }}</h3>
</template>

<script>
  export default {
    props: {
      postTitle: {
        type: String,
        required: true,
        default: 'WIP'
      }
    }
  }
</script>
```

```vue
// 2️⃣ src/components/Home.vue
<template>
  <blog-post post-title="Hello EmberConf!" />
</template>

<script>
  import BlogPost from './BlogPost'

  export default {
    components: { BlogPost }
  }
</script>
```

Ember (arguments)

```hbs
{{!-- 1️⃣ app/components/blog-post.hbs --}}
<h3 ...attributes>
  {{ @postTitle }}
</h3>
```

```hbs
{{!-- 2️⃣ app/about/template.hbs --}}
<BlogPost @postTitle="Tomster's take" />
```

Component state and reactivity

Vue

```vue
// src/components/AspectRatio.vue
<template>
  Aspect ratio is {{ this.aspectRatio }}
</template>

<script>
  export default {
    data() {
      return { width: 16, height: 9 }
    },
    computed: {
      aspectRatio() {
        return this.width / this.height
      }
    },
    watch: {
      width(newValue, oldValue) {
        console.log(`${oldValue} → ${newValue}`)
      }
    }
  }
</script>
```

Ember

```ember
{{!-- app/components/aspect-ratio.hbs --}}
Aspect ratio is {{ this.aspectRatio }}

// app/components/aspect-ratio.js
import Component from '@glimmer/component'
import { observer } from '@ember/object'
import { tracked } from '@glimmer/tracking'

export default class AspectRatioComponent extends Component {
  @tracked width = 16
  @tracked height = 9

  constructor() {
    super(...arguments)

    // 🤦‍♂️ Please avoid
    observer('width', () => console.log('Width changed'))
  }

  get aspectRatio() {
    return this.width / this.height
  }
}
```

Events

Vue

```
// src/main.js
import { createApp } from 'vue'

const app = createApp({
  template: `<button-counter />`
})

app.component('button-counter', {
  data() {
    return { count: 0 }
  },
  template: `
    <button v-on:click="count++">
      You clicked {{ count }} times
    </button>
  `
})

app.mount('#app')
```

Ember

```
{{!-- 1️⃣ app/components/button-counter.hbs --}}
<button {{on "click" this.handleClick}}>
  You clicked {{ this.count }} times
</button>

// 2️⃣ app/components/button-counter.js
import Component from '@glimmer/component'
import { action } from '@ember/object'
import { tracked } from '@glimmer/tracking'

export default class ButtonCounterComponent extends Component {
  @tracked count = 0

  @action
  handleClick() {
    this.count += 1
  }
}
```

Custom events

Vue

```vue
// 1️⃣ src/components/CustomButton.vue
<template>
  <button @click="emitCustomClick">
    Click me
  </button>
</template>

<script>
  export default {
    methods: {
      emitCustomClick() {
        this.$emit('customClick')
      }
    }
  }
</script>
```

```vue
// 2️⃣ src/components/HomePage.vue
<template>
  <custom-button @custom-click="log" />
</template>

<script>
  import CustomButton from './CustomButton'

  export default {
    components: { CustomButton },
    methods: {
      log: () => console.log('Custom click received')
    }
  }
</script>
```

Custom events

Ember

```
{{!-- 1️⃣ app/components/custom-button.hbs --}}
<button {{on "click" this.handleClick}}>
  Click me
</button>

// 2️⃣ app/components/custom-button.js
import Component from '@glimmer/component'
import { action } from '@ember/object'

export default class CustomButtonComponent extends Component {
  @action
  handleClick() {
    this.args.customClick();
  }
}

{{!-- 3️⃣ app/components/button-container.hbs --}}
<CustomButton @customClick={{this.log}} />
```

Custom directives/helpers

Vue (directives)

```
// src/directives/pin.js
app.directive('pin', (el, binding) => {
  el.style.position = 'fixed'
  const s = binding.arg || 'top'
  el.style[s] = binding.value + 'px'
})

<!-- Usage example -->
<p v-pin="200">A
  Place me 200px from the top of the page
</p>
```

Ember (helpers)

```
// app/modifiers/pin.js
import { modifier } from 'ember-modifier'

export default modifier(function pin(el, [pixels], { dir } = {}) {
  el.style.position = 'fixed'
  const s = dir || 'top'
  el.style[s] = pixels + 'px'
})

{{!-- Usage example --}}
<p {{pin “200”}}>
  Place me 200px from the top of the page
</p>



// For more details check
// github.com/ember-modifier/ember-modifier
```

Slots/blocks

Vue (slot)

```vue
// 1️⃣ src/components/ButtonWithSlot.vue
<template>
  <button class="btn-primary">
    <slot></slot>
  </button>
</template>

// 2️⃣ src/components/AddTodo.vue
<template>
  <todo-button>Add todo</todo-button>
</template>

<script>
  import ButtonWithSlot from './ButtonWithSlot'

  export default {
    components: { 'todo-button': ButtonWithSlot }
  }
</script>
```

Ember (block)

```hbs
{{!-- 1️⃣ app/components/todo-button.hbs --}}
<button class="btn-primary">
  {{yield}}
</button>

{{!-- 2️⃣ app/about/template.hbs --}}
<TodoButton>
  Add todo
</TodoButton>
```

```hbs
<!-- Resulting HTML -->
<button class="btn-primary">
  Add todo
</button>
```

Named slots/blocks

Vue (named slots)

```
// 1️⃣ src/components/ModalLayout.vue
<template>
  <div class="container">
    <header>
      <slot name="header"></slot>
    </header>
    <main>
      <slot></slot>
    </main>
    <footer>
      <slot name="footer"></slot>
    </footer>
  </div>
</template>
```

```
// 2️⃣ src/components/SimpleModal.vue
<template>
  <modal-layout>
    <template v-slot:header>
      <h1>Here might be a page title</h1>
    </template>

    <template v-slot:default>
      <p>A paragraph for the main content.</p>
      <p>And another one.</p>
    </template>

    <template v-slot:footer>
      <p>Here's some contact info</p>
    </template>
  </modal-layout>
</template>

<script>
  import ModalLayout from './ModalLayout'

  export default {
    components: { 'modal-layout': ModalLayout }
  }
</script>
```

Named slots/blocks

Ember (named blocks)

```hbs
{{!-- 1️⃣ app/components/modal-layout.hbs --}}
<div class="container">
  <header>
    {{yield to="header"}}
  </header>
  <main>
    {{yield to="body"}}
  </main>
  <footer>
    {{yield to="footer"}}
  </footer>
</div>
```

```hbs
{{!-- 2️⃣ app/components/simple-modal.hbs --}}
<ModalLayout>
  <:header>
    <h1>Here might be a page title</h1>
  </:header>

  <:body>
    <p>A paragraph for the main content.</p>
    <p>And another one.</p>
  </:body>

  <:footer>
    <p>Here's some contact info</p>
  </:footer>
</ModalLayout>
```

Global/application state

Vue

- Vuex for store(s)
- Provide/Inject in Vue 3

Ember

- Services
- Ember-data

Application routing

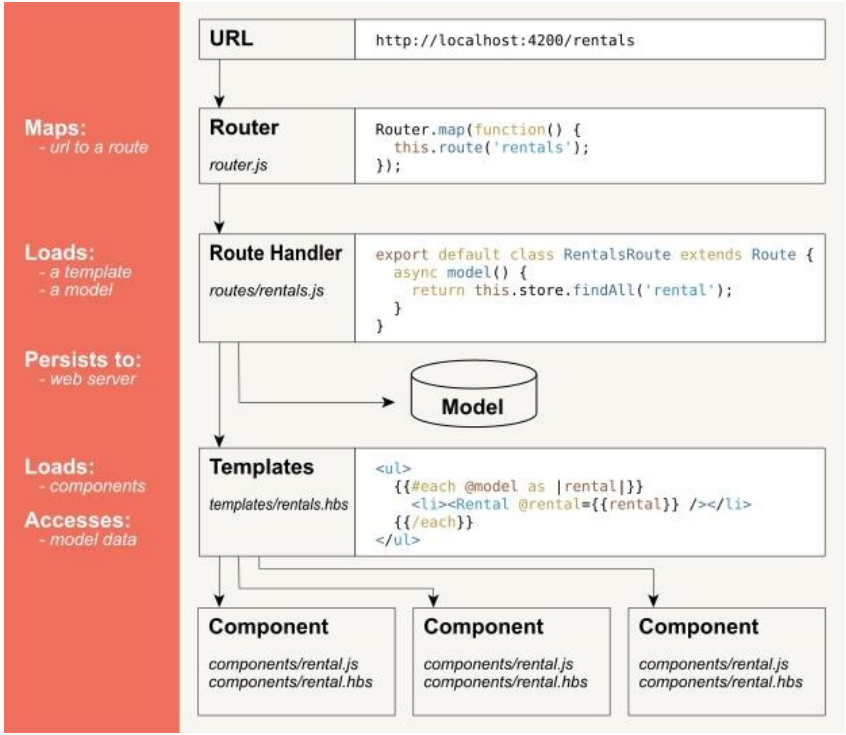
Vue

```
import { createApp } from 'vue'
import { createRouter, createWebHistory } from 'vue-router'

const Home  = { template: `<h1>Home</h1>`  }
const About = { template: `<h1>About</h1>` }

const routes = [
  { path: '/',      component: Home  },
  { path: '/about', component: About },
]
const router = createRouter({
  history: createWebHistory(),
  routes,
})

createApp({
  template: `
    <ul>
      <li><router-link to="/">Home</router-link></li>
      <li><router-link to="/about">About</router-link></li>
    </ul>
    <router-view></router-view>
  `,
})
.use(router)
.mount('#app')
```

Ember

```
// app/router.js
Router.map(function () {
  this.route('index', { path: '/' })
  this.route('about')
})

{{!-- app/templates/application.hbs --}}
<ul>
  <li><LinkTo @route="index">Go to Home</LinkTo></li>
  <li><LinkTo @route="about">Go to About</LinkTo></li>
</ul>
{{outlet}}

// /app
// ├── /about
// │   └── template.hbs
// └── /index
//     └── template.hbs
```

Application routing



Ember

```
// app/router.js
Router.map(function () {
  this.route('index', { path: '/' })
  this.route('about')
})

{{!-- app/templates/application.hbs --}}
<ul>
  <li><LinkTo @route="index">Go to Home</LinkTo></li>
  <li><LinkTo @route="about">Go to About</LinkTo></li>
</ul>
{{outlet}}

// /app
// ├── /about
// │   └── template.hbs
// └── /index
//     └── template.hbs
```

Data sources

Vue

- No official guidance on this
- Use whatever you want
- Vuex, store(s) and POJOs
  - Plain Old JavaScript Objects
- GraphQL
- Other ORM packages

Ember

- Ember-data is the default approach
- GraphQL (e.g. Ember-apollo)
- Service

Testing

Vue

- Loose official guidance
- Recommendations and popular tools
  - Vue Test Utils
  - Vue Testing Library (@testing-library/vue)
  - Cypress.io
  - Nightwatch.js
  - Puppeteer
  - TestCafe

Ember

- “gives you the power to write tests and be productive from day one”
- “Every Ember app comes with QUnit and QUnit DOM”
- Ember CLI
  - ember g model-test student
  - ember g component-test student
  - ember g acceptance-test students

Animations & transitions

Vue

```
// src/components/ButtonFade.vue
<template>
  <div id="demo">
    <button @click="show = !show">Toggle</button>

    <transition name="slide-fade">
      <p v-if="show">Hello EmberConf</p>
    </transition>
  </div>
</template>

<script>
  export default {
    data() {
      return { show: true }
    }
  }
</script>
```

```
<style>
  .slide-fade-enter-active {
    transition: all .3s ease;
  }
  .slide-fade-leave-active {
    transition: all .8s ease;
  }
  .slide-fade-enter, .slide-fade-leave-to {
    transform: translateX(10px);
    opacity: 0;
  }
</style>
```

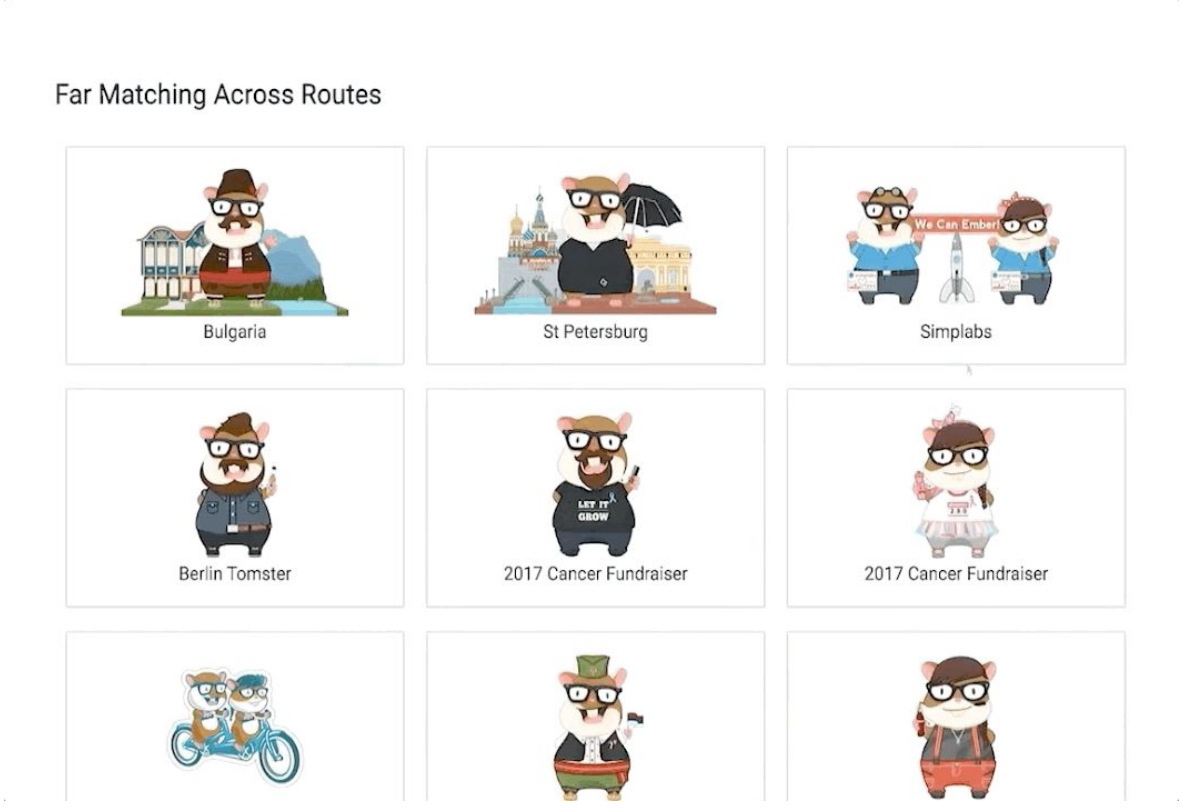
Animations & transitions

Ember (ember-css-transitions)

```
{{!-- 1️⃣ app/components/button-fade.hbs --}}
<button {{on "click" (fn (mut this.show) (not this.show))}}>
  Toggle
</button>

{{#if this.show}}
  <div {{css-transition "slide-fade"}}>
    Hello EmberConf
  </div>
{{/if}}

/* 2️⃣ app/styles/app.css */
.slide-fade-enter-active {
  transition: all .3s ease;
}
.slide-fade-leave-active {
  transition: all .8s ease;
}
.slide-fade-enter, .slide-fade-leave-to {
  transform: translateX(10px);
  opacity: 0;
}

/* For more details:
   github.com/peec/ember-css-transitions */
```

Animations & transitions

Ember (ember-animated)



template.hbs

```
{{#animated-each guests use=transition}}

  <Icon 'user' />

{{/animated-each}}
```

component.js

```
import Component from '@ember/component';

import fade from 'ember-animated/transitions/fade';



export default Component.extend({

  transition: fade,
```

Where can you read more?

- 🔗 emberatlas.com
  - 🔗 Ember for Vue developers
- 🔗 emberobserver.com
- 🔗 twitter.com/embertimes
- 🔗 discord.com/invite/emberjs

</talk>

Gonçalo Morais
@gnclmorais

More state: v-bind & v-model

Vue

- `v-bind`
- `v-model`

Ember

- `{{variable}}`
- `<Input>` and `<Textarea>`

Code inheritance/sharing

Vue

- Component inheritance
  - Move common code to a component that doesn't work by itself (i.e. only JavaScript gets inherited)
- Mixins
- Vue 3: Composition API

Ember

- Vanilla JS class extension
- Mixins
- Services
- @tracked
  - Gets us closer to (parts of) the Composition API

Jim Schofield

- router (the router in ember is much much more like nuxt!)
- component (folder structure registers components for you! And you can namespace components! :yay: Reactivity is all about creating an object with tracked properties, which has many gotchas I had to deal with (like an array of objects not triggering changes)
- props (syntax! what is this @ we see now?)
- computed (this is a weird beast I don't understand in ember, but I see plenty of get foo() now and that seems to do the trick. I know Vue caches things by default, but I'm unsure how to cache things in Ember
- watch (Does ember have a concept of watch? I guess you can kind of in a getter)
- single file components (Don't exist in ember, right?)
- stores (You can use redux in Ember... but really you lean on the model that is tied to the route... I suppose you could create a service! services are a nice first class thing in ember I don't think exists in Vue until Vue 3)
- tests (I'm bad at tests)
- slots (I miss this in Ember, but I think it's coming soon:tm:. I know there are addons that do that)
- Vue 3 is heading headlong into the instance-less, more functional world React is pioneering with hooks. I think Ember embraces the Class and uses it in a really composable way.
- Also, Ember really relies on routes as the main object of interest. In Vue, there was this hard-to-discern difference between "views" and "components". In Ember it's much more defined and useful
- ooh, and talking about ember-data would probably be interesting... because we had nothing out-of-the-box like that in vue
- oooh, transitions