

Look at You, Vue: Chariot SPA Day 2018

Speaker - Ken Rimple

What is Vue?

What is Vue?

- A single-page JavaScript framework
- Created by Evan You, a former Googler who worked on AngularJS projects

Design goal

"I figured, what if I could just extract the part that I really liked about Angular and build something really lightweight without all the extra concepts involved?"

— Source: Wikipedia

Key Vue Technologies

- The Vue instance
- Vue components, directives, filters
- Vue Mixins add *features* to Vue
 - The Vue Router
 - Vuex - an application state manager
 - Numerous component libraries

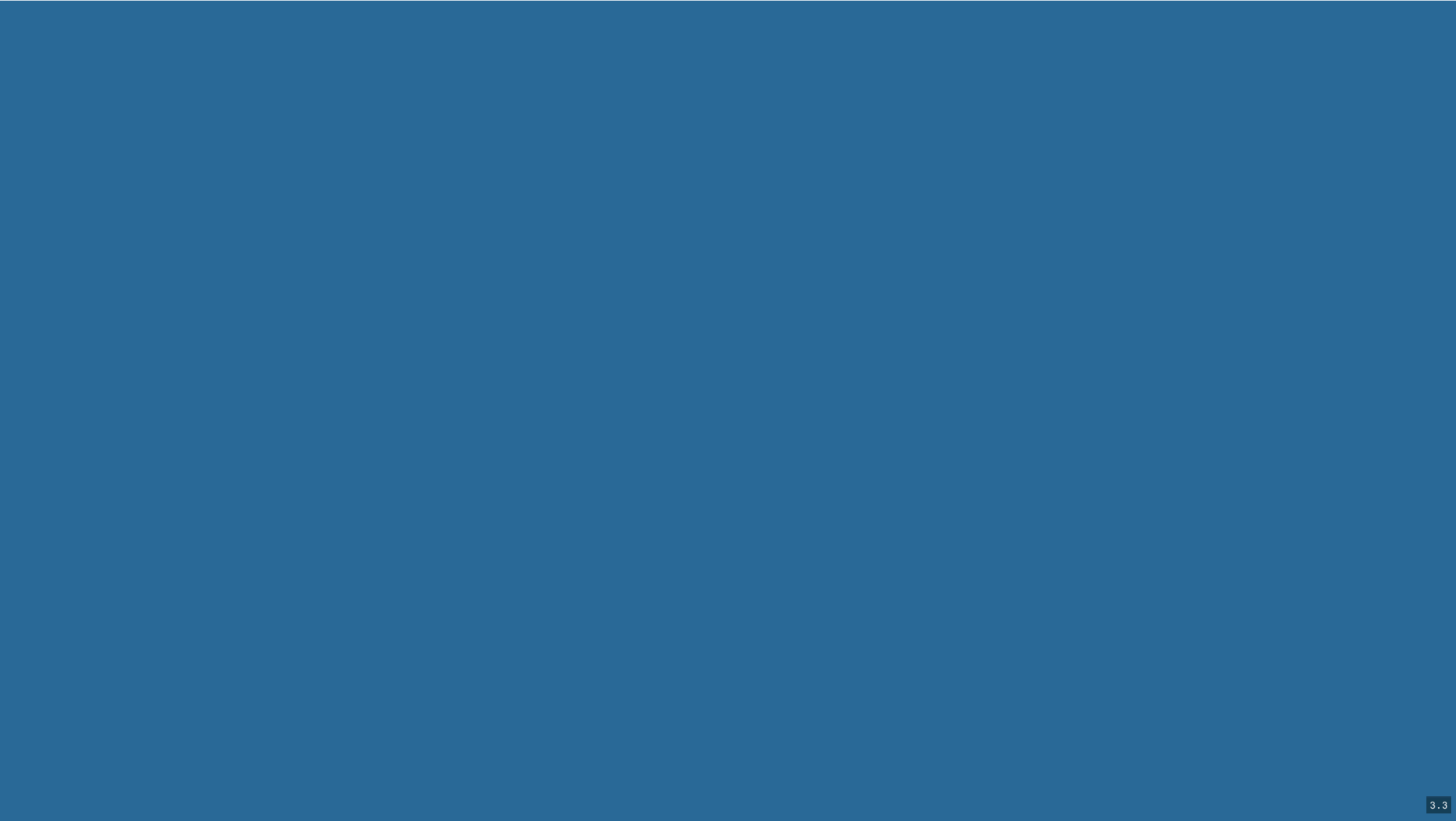
The Vue Instance

Vue apps start with a Vue Instance

```
1  var app = new Vue({
2    el: '#app',           (1)
3    data: { ... },      (2)
4    computed: {...},    (3)
5    methods: { ... },   (4)
6    filters: { ... },   (5)
7    ... etc ...
8  });
```

- 1 The element to assign this Vue instance to
- 2 The properties exposed and watched by Vue
- 3 Computations of derived data to expose
- 4 Methods to execute from events
- 5 Data transformation functions

Demo: a simple app, without ECMAScript modules...



Vue Components...

Components...

- Hold state (with instance variables)
- Accept incoming parameters
- May emit events to parent components
- Define their own methods
- Wire methods to DOM events
- Contain their own templates
- Are styled via CSS

Component Syntax

Components can be globally registered

```
1 Vue.component('alert', {  
2   props: ['message'],           (1)  
3   name: 'alert',               (2)  
4   template: `<div>Alert:  {{message}}</div>` (3)  
5 });
```

- 1 Define props to accept input data
- 2 Define the component name
- 3 Define a template to render the component

Component mounted in an app, bound to an element...

```
1 Vue.component('alert', {
2   props: ['message'],
3   name: 'alert',
4   template: `<div class='box'>ALERT: {{message}}</div>
5 });
6
7 const app = { template: `
8   <div><alert :message="'10 minutes left'"></alert><,
9   ` };
10
11 new Vue({
12   el: '#app',
13   render: view => view(app)
14 });
```

i Most developers use Single-File Components instead...

Single File Component: alert.vue

```
1 <template>
2   <div class="box">{{ message }}</div>
3 </template>
4
5 <script>
6   export default {
7     el: 'alert',
8     props: ['message']
9   }
10 </script>
11
12 <style>
13   .box { ... }
14 </style>
```

SFCs and Styles

With **node-sass** installed you can compile **sass** templates in SFCs

```
1 <style lang="scss">
2   $input-radius: 6px;
3
4   .form {
5     select {
6       border-radius: $input-radius;
7     }
8   }
9 </style>
```

💡 SFCs require a Webpack build and the Vue compiler

Vue Style Scoping

You can namespace your SASS.

```
1 <style lang="scss" scoped> (1)
2   input {
3     background-color: red;
4   } (2)
5 </style>
```

- 1 Now the mounted component and styles have an attached hash attribute, computed for this component and *its children*
- 2 The input style won't bleed to other external components

Advanced: Vue Styles with CSS Modules

Vue also supports SASS Modules

```
1 <template>
2   <input class="$styles.foo" />
3 </template>
4 <style lang="scss" module>
5   input.foo {                                (1)
6     background-color: red;                  (2)
7   }
8 </style>
```

- 1 Adding module to the style declaration enables CSS Modules
- 2 You must add classes for all HTML elements, or they will affect everything (are not namespaced without an attached class)

Good Blog Article on Scoped styles -vs- CSS Modules

<http://www.netguru.co/codestories/vue.js-scoped-styles-vs-css-modules>

💡 Use CSS Scopes for simple designs, and CSS Modules when you want precise control for widget libraries

Changing Data

Using the **created** lifecycle method - <https://codesandbox.io/s/k58z4qvv3>

```
1 export default {
2   data: function() { return {msg: 'Hello'}; },
3   created: function() {
4     setTimeout(() => {
5       this.msg = 'Goodbye!';
6     }, 4000);
7   },
8   template: `
```

💡 Objects placed in the data property are reactive and update their view when changed

Vue and Events

A simple method in a SFC - <https://codesandbox.io/s/4w6j0m8vn4>

```
1 <template>
2   <button @click="poke">Poke!</button>
3   <span> Poked {{ pokeCount }} times</span>
4 </template>
5 <script>
6   export default {
7     data: function () { return { pokeCount: 0 }; },
8     methods: {
9       function poke () {
10        this.pokeCount++;
11      }
12    }
13  }
14 </script>
```

A simple Vue Form

```
1 <form class="form" @submit.prevent="submit">
2   <input type="text" required v-model="form.name">(1)
3   <input type="text" required v-model="form.email">(2)
4   <select required v-model="form.treatment">      (2)
5     <option v-for="(option, index) in treatments"
6       :key="index">                                (3)
7       {{ option }}
8     </option>
9   </select>
10  <button>Register!</button>
11 </form>
```

- 1 @submit event, modified with .prevent to prevent post action
- 2 v-model binds form fields to data properties, allows for validation
- 3 v-for iterator similar to AngularJS / Angular, needs :key to improve re-render performance like React

Vue Router

Vue Router

- A router developed by the Vue team
 - Provides component navigational management
 - Configured using the **Router** class

Sample Router config

provide in a file like `router-config.js`

```
1 // Add Router mixin...
2 Vue.use(Router);
3
4 export default new Router({
5   routes: [
6     {path: '/', redirect: '/schedule'},
7     {path: '/schedule', component: Registration},
8     {path: '/register/:id', component: Registration},
9     ...
10  ]
11 });
```

i Vue configures routes in JavaScript, similar to Angular

Mounting the Router

Add the Router to the Vue instance properties

```
1 import router from './router-config.js';
2
3 new Vue({
4   data: function() { ... },
5   ...
6   router,                                (1)
7   ...,
8   render: h => h(App)
9 }).$mount('#app');
```

1 Shorthand for router: router, a feature of ECMAScript 2015.

i Once installed, the `$route` and `$router` properties are made available to components

Using the Router

```
1 <template>
2   <h1>SPA Day</h1>
3   <router-link to="/">Home</router-link>
4   <router-link to="/registration">Register</router-link>
5   <hr/>
6   <router-outlet></router-outlet>
7 </template>
```

i the Vue Router router-outlet component renders all views below the horizontal rule

Other Router Link Examples

```
1 <template>
2   <router-link to="/register/12">
3     Register for session 12
4   </router-link>
5
6   <router-link
7     :to="{ path: '/register', params: {id: currentId}}
8     Register
9   </router-link>
10 </template>
```

💡 `currentId` above could be a data or computed property

VueX State Management

VueX is...

- A state management library that...
 - Maintains state as an object graph
 - Dispatches *actions* to request state changes
 - Manages state with *mutations*

A Vuex store

provide in a file like `store-conf.js`

```
1 // Add the Vuex mixin to Vue
2 Vue.use(Vuex);
3
4 // Create the store...
5 export default new Vuex.Store({
6   state: {...},
7   actions: {...},
8   mutations: {...},
9   getters: {...},
10  ...
11  })
```

Mounting Vuex in your Vue instance

```
1 import store from 'store-conf.js';
2
3 new Vue({
4   data: function() { ... },
5   ...
6   router,
7   store,
8   ...
9   render: h => h(App)
10 }).$mount('#app');
```

i Once installed, components can access the `$store` property and helper methods

Installing the store in your Vue instance

Import the store and install it as a property in your Vue instance

```
1 import store from './store';  
2 ...  
3 new Vue({  
4   router,  
5   store,  
6   ...  
7 }).$mount('App');
```


VueX and Redux compared

Activity	VueX	Redux
Changes to State	"Simple" Mutation Methods	Immutable Reducer Function
Request A Change	Call an Action	Dispatch an Action
Bindings to Components	<code>mapState</code> , <code>mapAction</code> helpers, <code>getters</code>	Higher-order Components and connect
Asynchronous Operations	Action methods (built-in)	Middleware-driven (select one)
Challenges	Most mutations just work, except collections...you need to replace them or use a supported function (like filter)	Everything <i>should</i> be treated as immutable, <code>reducers</code> can be tough to reason about

Wrap-up

Strengths

- Vue *single page components* nicely contain scripts, styles, templates
- *VueX* is not as complicated as Redux with its reducer and forced immutability everywhere
- The *Vue CLI* is well thought-out and does just what it needs to do

Strengths

- Vue *slots* make wrapping nested content nicer than Angular
- **Typescript** support in CLI
- CLI supports modifying Webpack config without 'ejecting'
- **Chrome and Firefox Vue DevTools** is useful and cover components, events, Vuex

Weaknesses

- No built-in service layer
- No built-in dependency injection system
- Pedantic eslint / tslint settings with **Vue CLI**
- Vue's exceptions are not wonderful. They tend to be cryptic and confusing
- Vue feels like a riff on AngularJS and React

Idiosyncracies

- VueX (and Vue) state mutation is a bit complex - since it holds proxies it can be tough to figure out what's going on
- Things to watch with VueX...
 - You *can* mutate data without making it immutable, not in all cases
 - Not if you normally use *push* to add to an array
 - Instead use a new object - `[...oldobj, newentry]`
 - Or use `Vue.set` to make changes

Code Walkthrough: The SPA App