Suade

Vue 3 Migration: A real life story

Vue Meetup March 2022

The challenge, aka the product:

Suade is a RegTech firm which offers Regulation-as-a-Service to automate regulatory compliance for financial institutions. It is one of the first start-ups to be classified as a true RegTech company.

The Challenge

Our SASS product

- Infomartion orintated product
- 60,000 lines of Javascript code
- 1,500 end to end & unit tests
- Very little dependencices

What we need to update

Internal Libraries

- Design System
- Services

External Libs

- Vue Router
- Vuex
- Vue Multiselect
- Vue Currency Input

The product

- Suade

The hero

It's Me, Hi

- Senior Frontend Engineer at Suade
- 11 years' experience in web orientated software engineering
- Open-Source Contributor
- Lead the development of vue-multiselect
- Founder of TrainFinder, a cheap train ticket finder.

Before the actual migration

01 Migrated our use of filters

02 Migrated our use of event busses

Filters

Remove and change how we were displaying our numerical information

```
<template>
  <p>Balance: {{ balance | currencyFormatter }}</p>
</template>

<script>
  no usages
  export default {
    props: {
      balance: {
        type: Number,
        required: true,
      },
      currency: {
        type: String,
        required: true,
      },
    },
    filters: {
      currencyFormatter(value) {
        return Tools.formatToLocaleNumber(value, this.currency);
      },
    },
  };
</script>
```

```
<template>
  <p>Balance: {{ currencyFormatter(balance) }}</p>
</template>

<script>
  no usages
  export default {
    props: {
      balance: {
        type: Number,
        required: true,
      },
      currency: {
        type: String,
        required: true,
      },
    },
    methods: {
      currencyFormatter(value) {
        return Tools.formatToLocaleNumber(value, this.currency);
      },
    },
  };
</script>
```

Filters

Remove and change how we were displaying our numerical information

```
app.config.globalProperties.formatters.currencyFormatter = Tools.formatToLocaleNumber;
```

```
<template>
  <p>Balance: {{ this.formatters.currencyFormatter(balance, currency) }}</p>
</template>

<script>
  no usages
  export default {
    props: {
      balance: {
        type: Number,
        required: true,
      },
      currency: {
        type: String,
        required: true,
      },
    },
  };
</script>
```

Event Bus

Migrate from using event buses

```
<script>
import eventBus from './eventBus.js'
2 usages
export default {
  mounted() {
    // adding eventBus listener
    eventBus.$on( stuff: 'custom-event', () => {
      console.log('Custom event triggered!')
    })
  },
  beforeDestroy() {
    // removing eventBus listener
    eventBus.$off( stuff: 'custom-event')
  }
}
</script>
```

```
// eventBus.js
const eventBus = new Vue();
3 usages
export default eventBus
```

Event Bus

Use custom events and emits

```
<template>
  <Parent-Component @custom-event="doSomething" />
</template>
```

```
<script>
2 usages
export default {
  emits:['custom-event'],
  mounted() {
    this.$emit('custom-event');
  },
}
</script>
```

Event Bus

Use reactive

```
import {reactive} from 'vue';

// For forms, store the valid status for a submit button, but we need to set it up like this so it is Vue reactive
const formValidStatus = reactive( target: {formValidStatus: {}});

Object.defineProperty(app.config.globalProperties.$suadeLibs,  p: 'formValidStatus',  attributes: {
  get() {
    return formValidStatus.formValidStatus;
  },
  set(value) {
    formValidStatus.formValidStatus = value;
  },
});
```

```
// formSubmit.vue
computed: {
  formValidStatus: function() {
    if (this.submit) {
      return this.$suadeLibs.formValidStatus[this.submit];
    } else {
      return true;
    }
  },
},
```

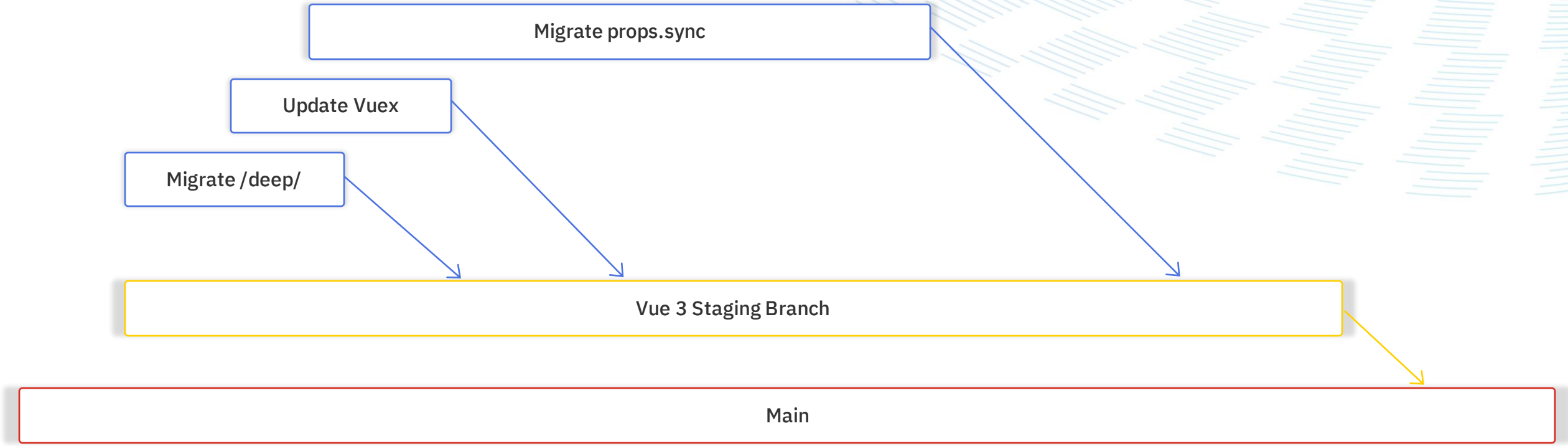
Migration Plan

01 Update props.sync -> v-model:custom

02 /deep/ -> :deep()

03 Update dependencies

Migration Plan



Update props.sync -> v-model:custom

The old way

```
<template>
  <sl-input-text :value.sync="searchText" />
</template>

<script>
2 usages
export default {
  data() {
    return {
      searchText: ''
    }
  },
}
</script>
```

```
<template>
  <input type="text" v-model="internalText"/>
</template>

<script>
2 usages
export default {
  name: "sl-input-text",
  props: {
    value: {
      type: String,
      required: true
    },
  },
  computed: {
    internalText: {
      get() {
        return this.value;
      },
      set(val) {
        this.$emit('update:value', val);
      }
    }
  }
}
</script>
```

Update props.sync -> v-model:custom

The new way

```
<template>
  <sl-input-text v-model="searchText" />
</template>

<script>
2 usages
export default {
  data() {
    return {
      searchText: ''
    }
  },
}
</script>
```

```
<template>
  <input type="text" v-model="internalText"/>
</template>

<script>
2 usages
export default {
  name: "sl-input-text",
  props: {
    value: {
      type: String,
      required: true
    },
  },
  computed: {
    internalText: {
      get() {
        return this.value;
      },
      set(val) {
        this.$emit('update:value', val);
      }
    }
  }
}
</script>
```

/deep/ -> :deep()

Old to new

```scss
<style lang="scss" scoped>
    .sl-loading-panel /deep/ .loading-overlay{
      z-index: 38;

      /deep/ .message{
        font-weight: 700;
      }
    }
</style>
```

```scss
<style lang="scss" scoped>
    .sl-loading-panel :deep(.loading-overlay){
      z-index: 38;

      .message{
        font-weight: 700;
      }
    }
</style>
```

Update all the dependencies

Vue Router and Vuex had breaking changes too

Breaking Changes

Changes are ordered by their usage. It is therefore recommended to follow this list in order.

new Router becomes createRouter

Vue Router is no longer a class but a set of functions. Instead of writing `new Router()`, you now have to call `createRouter`:

```js
// previously was
// import Router from 'vue-router'
import { createRouter } from 'vue-router'

const router = createRouter({
  // ...
})
```

Breaking Changes

Installation process

To align with the new Vue 3 initialization process, the installation process of Vuex has changed. To create a new store, users are now encouraged to use the newly introduced createStore function.

```js
import { createStore } from 'vuex'

export const store = createStore({
  state () {
    return {
      count: 1
    }
  }
})
```

To install Vuex to a Vue instance, pass the `store` instead of Vuex.

```js
import { createApp } from 'vue'
import { store } from './store'
import App from './App.vue'

const app = createApp(App)

app.use(store)

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

When I do this all again, what I would do different

01 Update dependencies to the most recent pre-breaking change before the migration

02 Reduce some tech debt

03

Takeaways

01 Prep beforehand with tests and pre-migration work

02 Plan the project. What are some areas of code freezes?

03 Read more about how migrations have gone & sign join the Discord chats

Thank you!

Any questions?

devtr.ee/mattelen

@matt_elen