Code & Coffee

Vue 3 overview

Matúš Makatura
Software Engineer @ InSchool
12th February 2021

Sections

What is Vue?

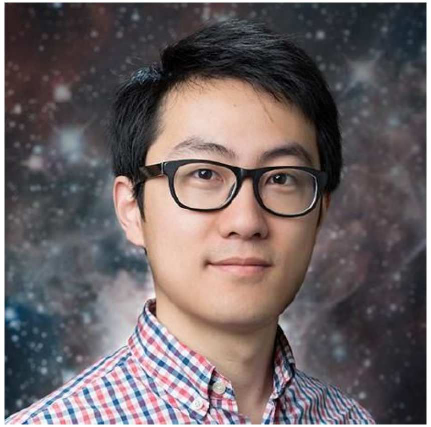
News in Vue 3

Should I migrate?

What is Vue?

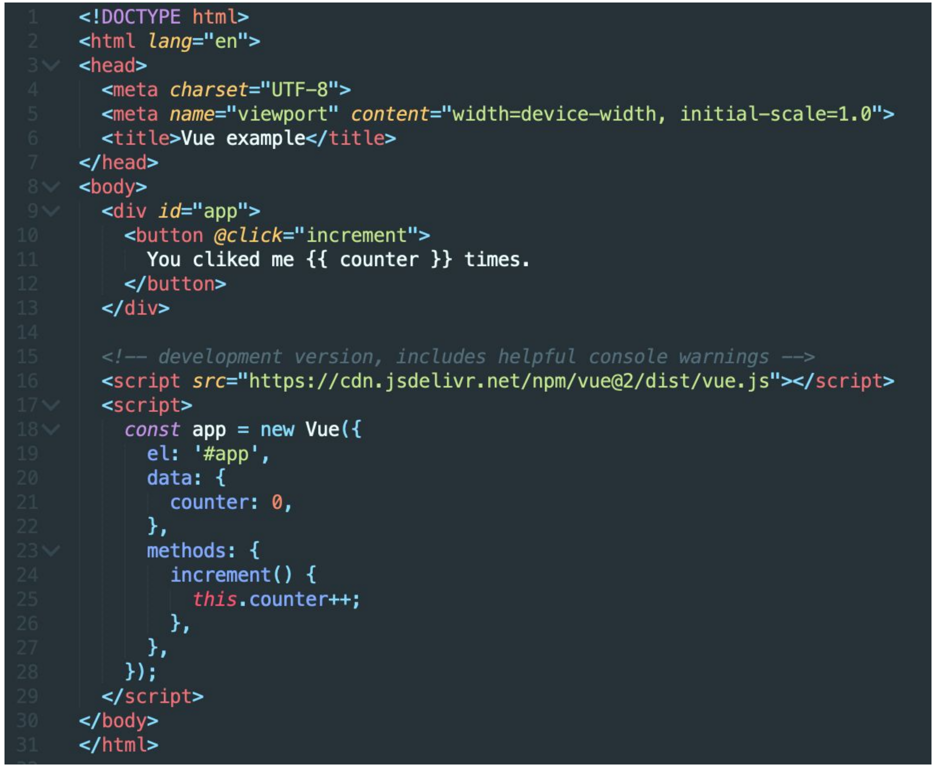
Short history

- Author: **Evan You** (**indie**, ex Google)
- Inspiration from AngularJS, goal to make it simple
- Progressive JS framework (or **library** if you will)
- First commit July 2013, first release January 2014
- 3 epochs
  - **Vue 1** - October 2015
  - **Vue 2** - September 2016
  - **Vue 3** - September 2020



Vue.js - The Documentary

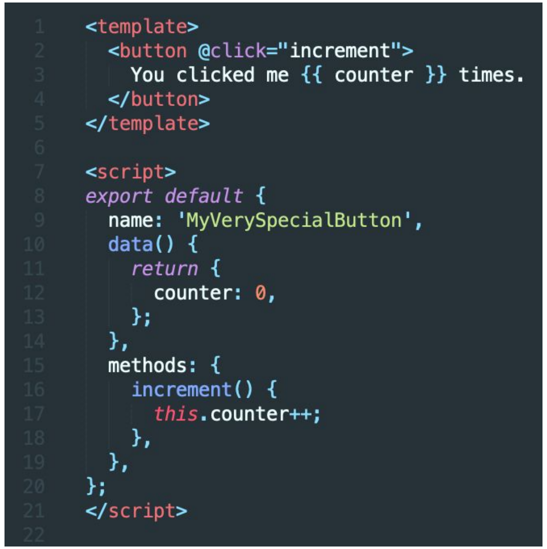
Simplest application



```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Vue example</title>
</head>
<body>
  <div id="app">
    <button @click="increment">
      You cliked me {{ counter }} times.
    </button>
  </div>

  <!-- development version, includes helpful console warnings -->
  <script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
  <script>
    const app = new Vue({
      el: '#app',
      data: {
        counter: 0,
      },
      methods: {
        increment() {
          this.counter++;
        },
      },
    });
  </script>
</body>
</html>
```

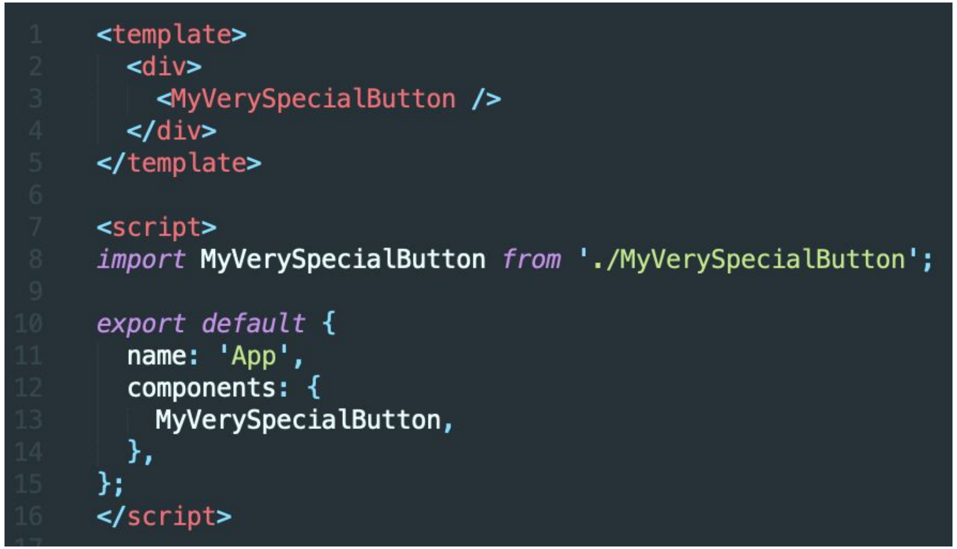
index.html

CLI application



```
<template>
  <button @click="increment">
    You clicked me {{ counter }} times.
  </button>
</template>

<script>
export default {
  name: 'MyVerySpecialButton',
  data() {
    return {
      counter: 0,
    };
  },
  methods: {
    increment() {
      this.counter++;
    },
  },
};
</script>
```

MyVerySpecialButton.vue



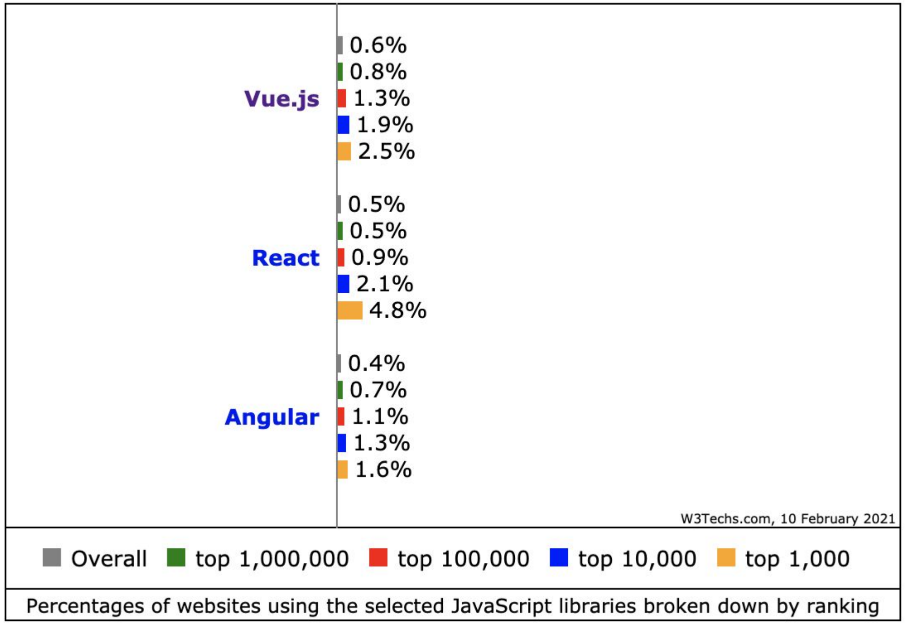
```
<template>
  <div>
    <MyVerySpecialButton />
  </div>
</template>

<script>
import MyVerySpecialButton from './MyVerySpecialButton';

export default {
  name: 'App',
  components: {
    MyVerySpecialButton,
  },
};
</script>
```

App.vue

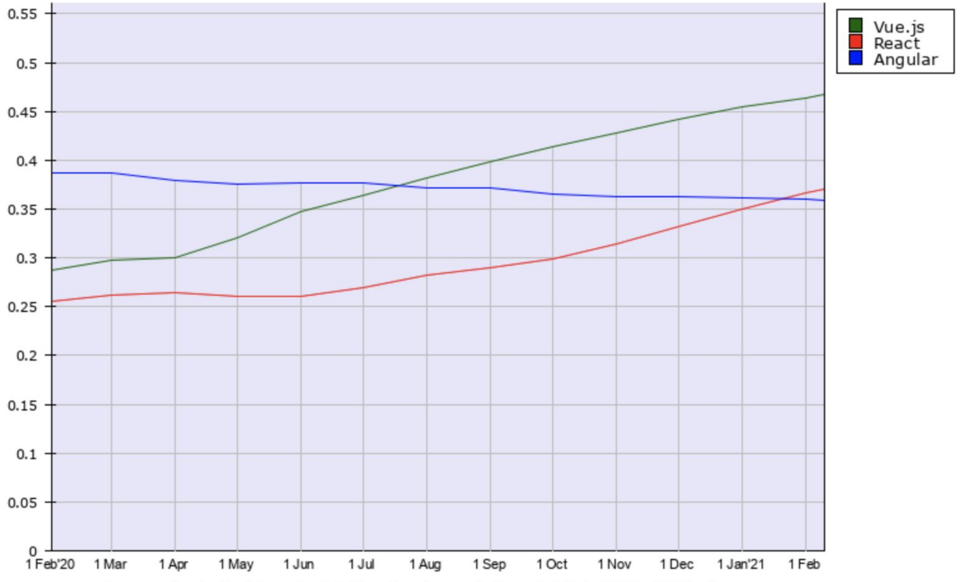
Comparison to Angular & React

- **HTML templates** instead of JSX (but it’s supported for rare cases)
- **One component per one file** (no multiple components, not spread across multiple files)
- Can be **learned in one afternoon** (even for backed dev)
- **Syntactically** similar to **AngularJS**
- Components are **stateful**, not immutable
- More detailed comparisons

Statistics



Percentages of websites using the selected JavaScript libraries broken down by ranking



Usage of selected JavaScript libraries for websites, 10 Feb 2021, W3Techs.com

Source: W3Tech - Comparison of the usage statistics of Vue.js vs. React vs. Angular for websites

Who uses Vue

*

Visma as well ->

News in Vue 3

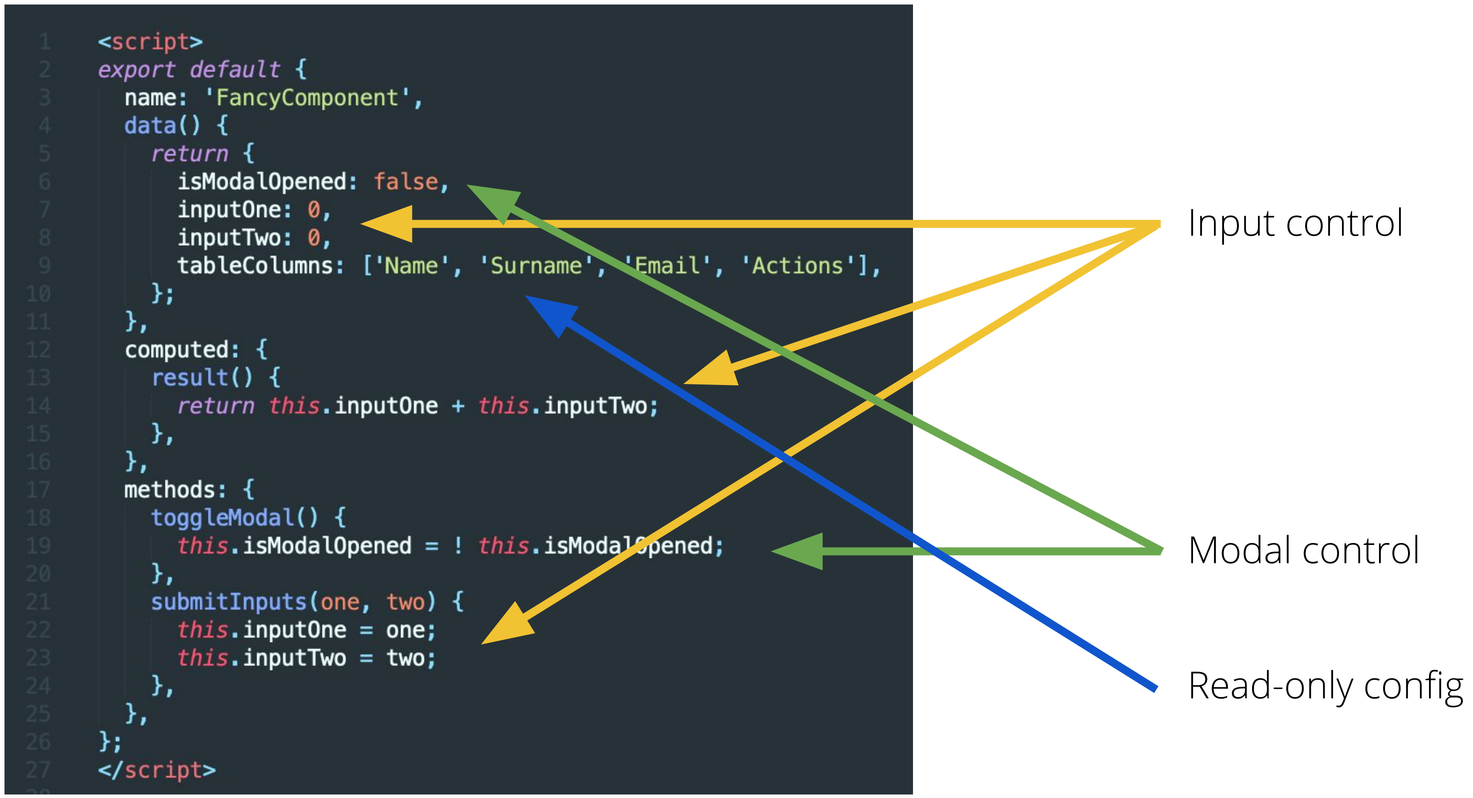
Basic differences

- Complete **rewrite to TypeScript** (but you don't have to use it)
- Restructured to support **tree-shaking**
  - You can now use Vue's part (like reactivity framework) separately from Vue! Yay!
- **Global** APIs **moved to instance** APIs

New features

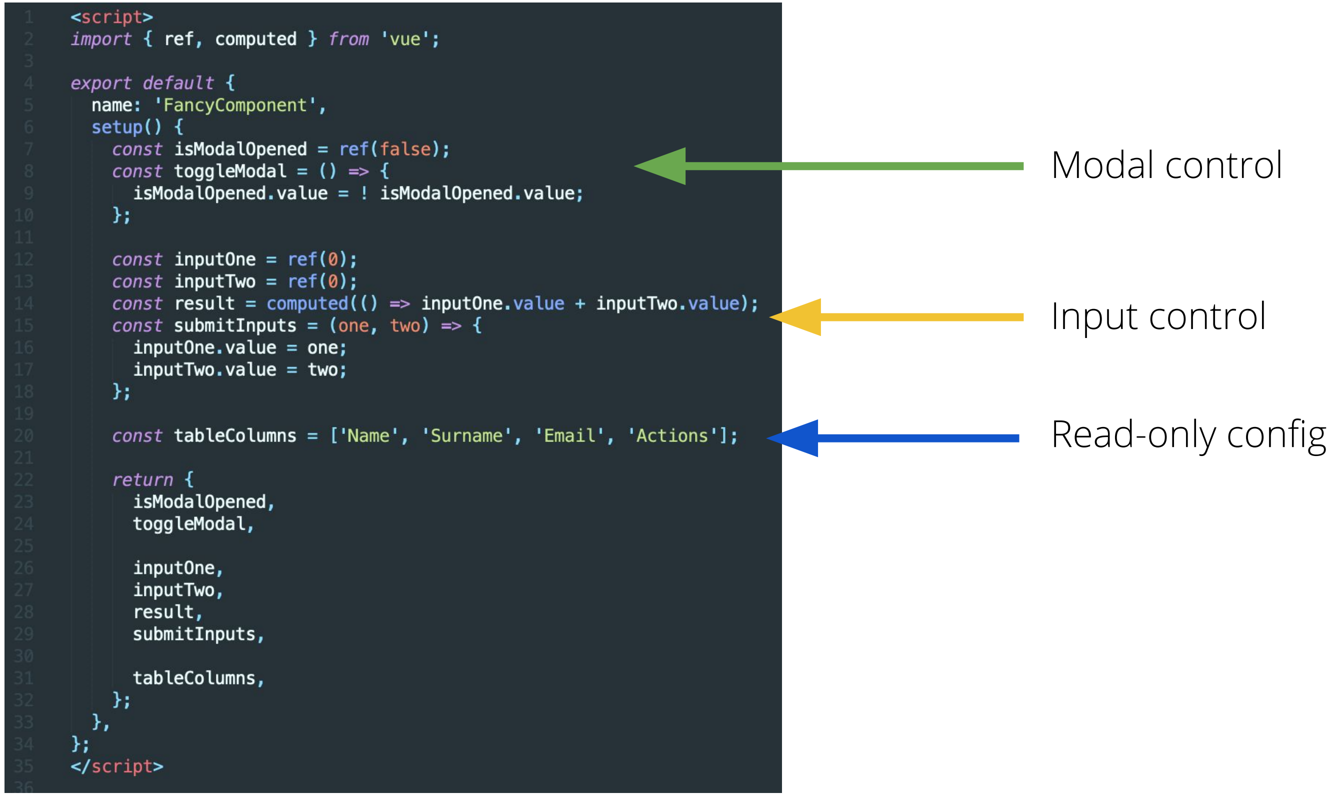
- Composition API
- Teleport
- Fragments
- “Emits” - new component option

Composition API - Vue 2 Options API



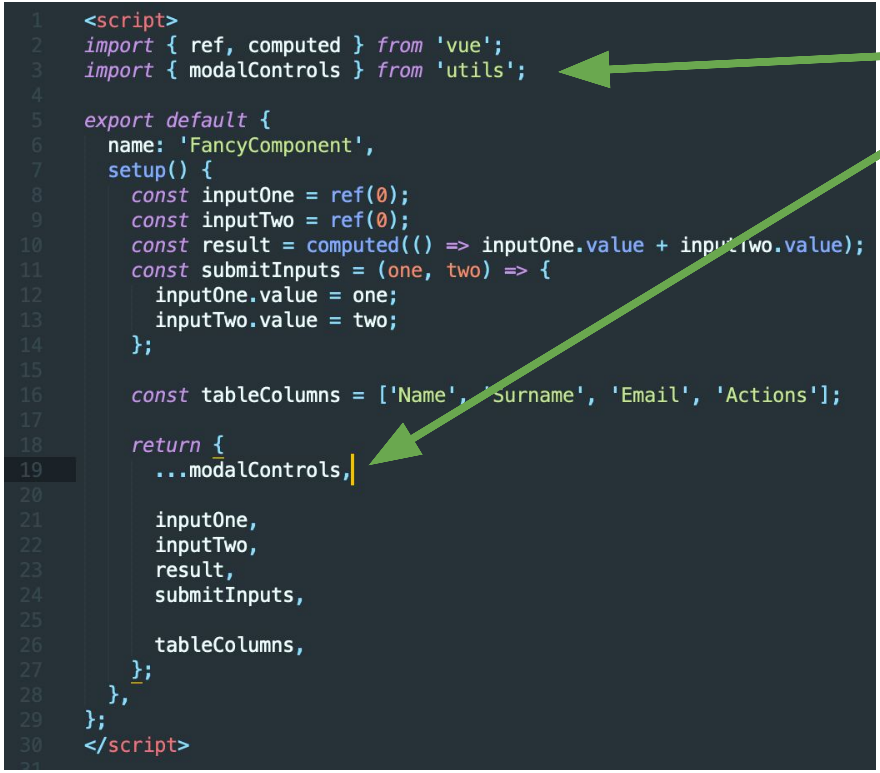
```
<script>
export default {
  name: 'FancyComponent',
  data() {
    return {
      isModalOpened: false,
      inputOne: 0,
      inputTwo: 0,
      tableColumns: ['Name', 'Surname', 'Email', 'Actions'],
    };
  },
  computed: {
    result() {
      return this.inputOne + this.inputTwo;
    },
  },
  methods: {
    toggleModal() {
      this.isModalOpened = ! this.isModalOpened;
    },
    submitInputs(one, two) {
      this.inputOne = one;
      this.inputTwo = two;
    },
  },
};
</script>
```

Composition API - new **alternative**



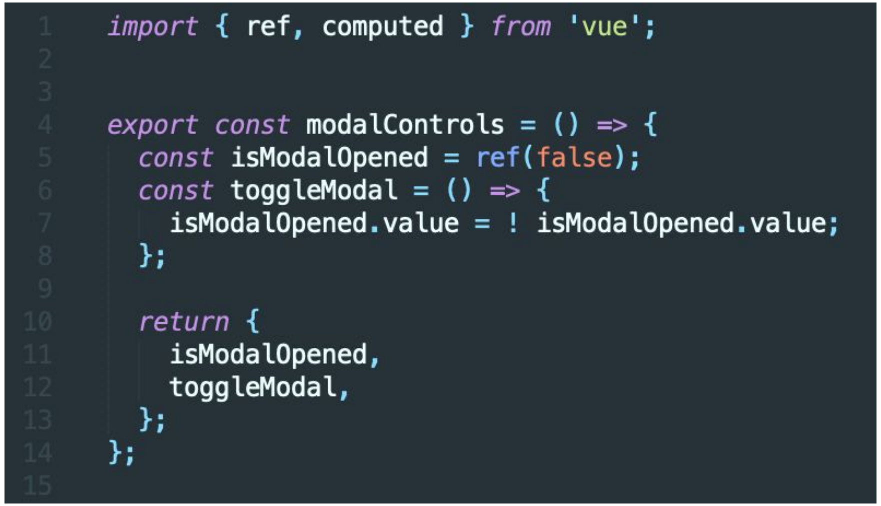
```
<script>
import { ref, computed } from 'vue';

export default {
  name: 'FancyComponent',
  setup() {
    const isModalOpened = ref(false);
    const toggleModal = () => {
      isModalOpened.value = ! isModalOpened.value;
    };

    const inputOne = ref(0);
    const inputTwo = ref(0);
    const result = computed(() => inputOne.value + inputTwo.value);
    const submitInputs = (one, two) => {
      inputOne.value = one;
      inputTwo.value = two;
    };

    const tableColumns = ['Name', 'Surname', 'Email', 'Actions'];

    return {
      isModalOpened,
      toggleModal,

      inputOne,
      inputTwo,
      result,
      submitInputs,

      tableColumns,
    };
  },
};
</script>
```

Composition API - new **alternative**



```
<script>
import { ref, computed } from 'vue';
import { modalControls } from 'utils';

export default {
  name: 'FancyComponent',
  setup() {
    const inputOne = ref(0);
    const inputTwo = ref(0);
    const result = computed(() => inputOne.value + inputTwo.value);
    const submitInputs = (one, two) => {
      inputOne.value = one;
      inputTwo.value = two;
    };

    const tableColumns = ['Name', 'Surname', 'Email', 'Actions'];

    return {
      ...modalControls,

      inputOne,
      inputTwo,
      result,
      submitInputs,

      tableColumns,
    };
  },
};
</script>
```

Modal control



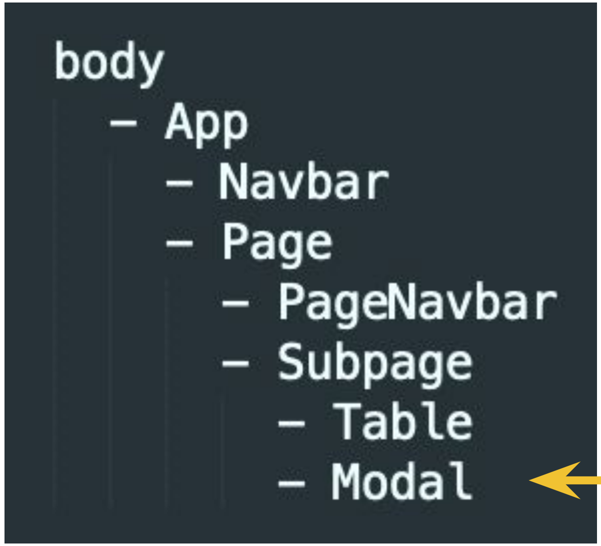
```
import { ref, computed } from 'vue';


export const modalControls = () => {
  const isModalOpened = ref(false);
  const toggleModal = () => {
    isModalOpened.value = ! isModalOpened.value;
  };

  return {
    isModalOpened,
    toggleModal,
  };
};
```

Composition API - conclusion

- It’s an **alternative**, not a replacement - pick your favorite or whichever works better!
- It’s **not a copy of React Hooks**, it’s different solution for the same problem
- Reactive programming at its finest
- **Code reusability** - it can be defined outside of the component
- Can **replace Vuex store** in many cases
- Works well for complex components (organisms & pages in atomic design)

Teleport - problem

Due to the way we design components, situations like this will arise:

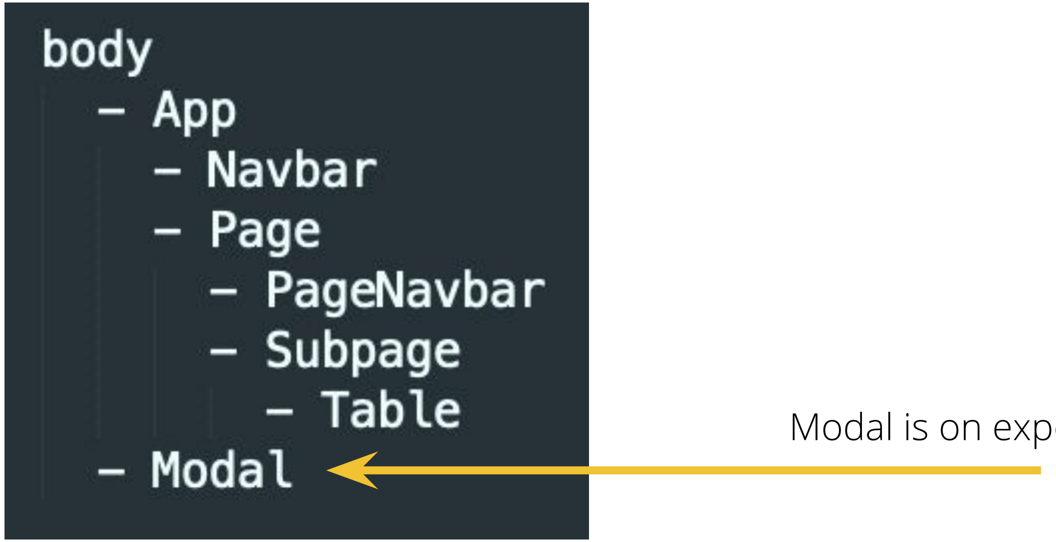


```
body
  - App
    - Navbar
    - Page
      - PageNavbar
      - Subpage
        - Table
        - Modal
```

Modal is nested but shown fullscreen

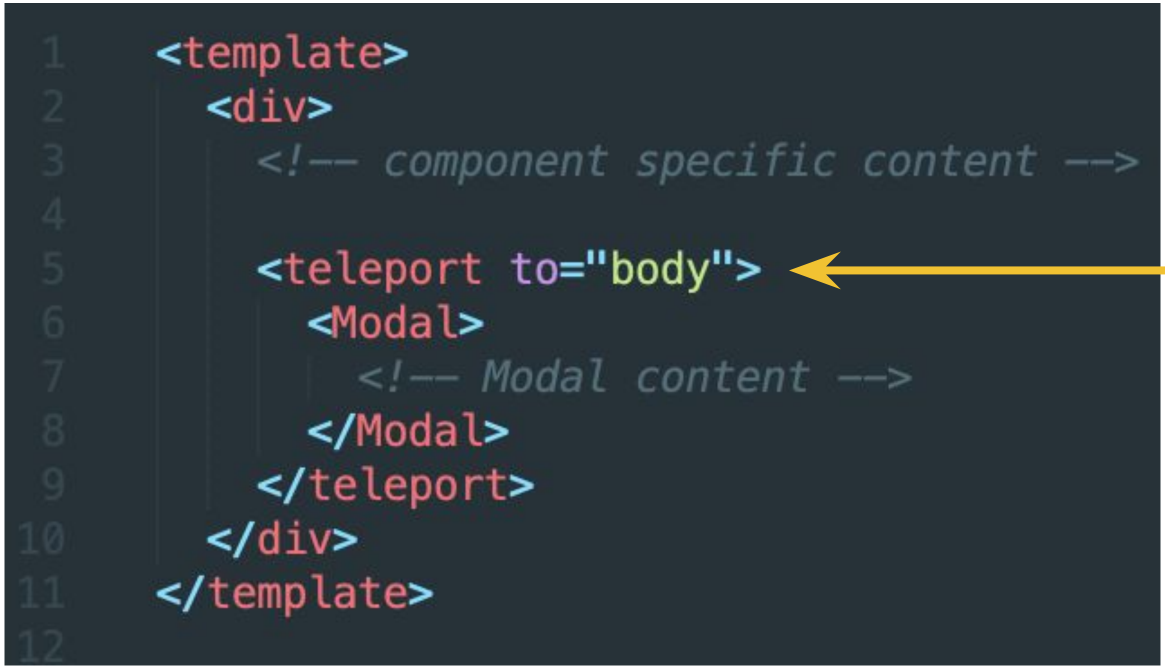
Teleport - desired state

It would make much more sense if it we could do this:



```
body
  - App
    - Navbar
    - Page
      - PageNavbar
      - Subpage
        - Table
  - Modal
```

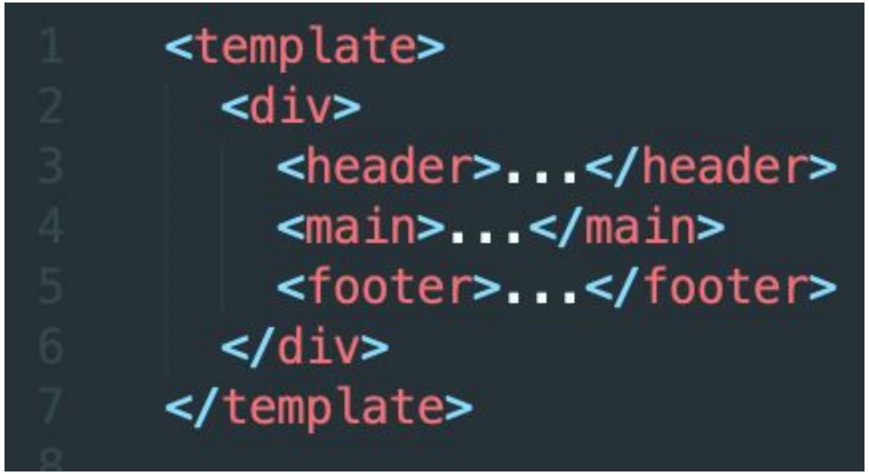
Teleport - implementation



```
<template>
  <div>
    <!-- component specific content -->

    <teleport to="body">
      <Modal>
        <!-- Modal content -->
      </Modal>
    </teleport>
  </div>
</template>
```

CSS selector!

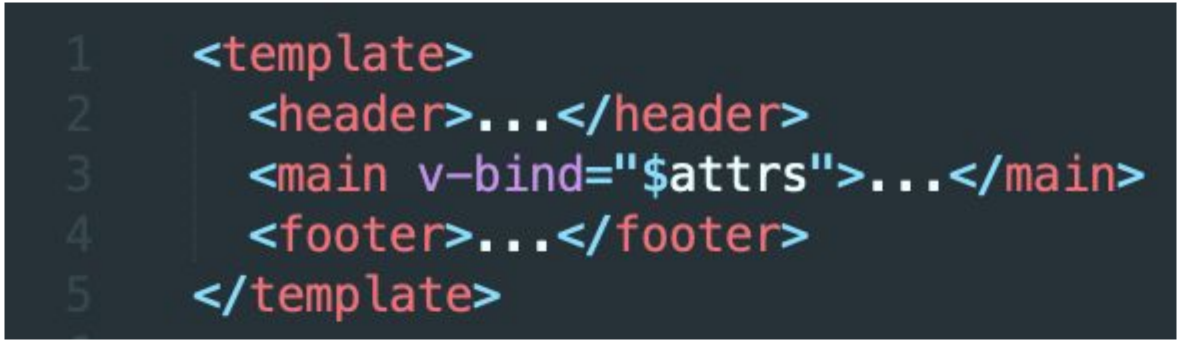
Fragments - multiple root components



```
<template>
  <div>
    <header>...</header>
    <main>...</main>
    <footer>...</footer>
  </div>
</template>
```

Vue 2 has no support - parent div is required

Fragments - multiple root components

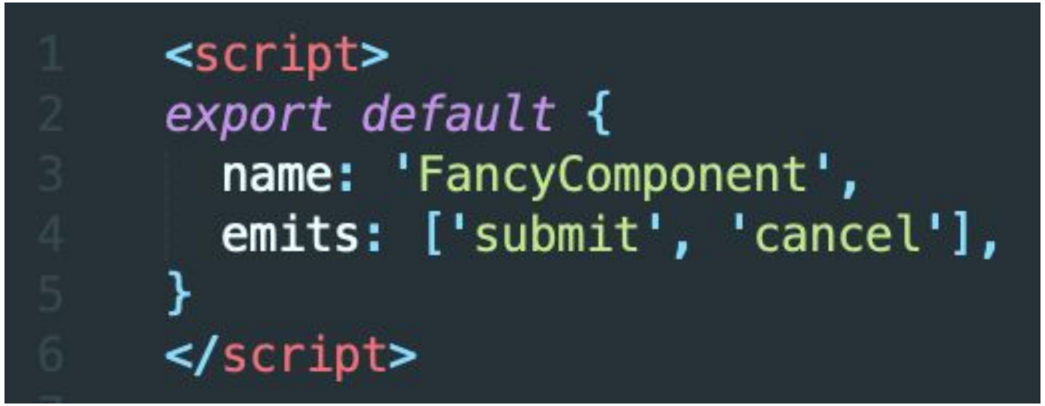


```
<template>
  <header>...</header>
  <main v-bind="$attrs">...</main>
  <footer>...</footer>
</template>
```

It works! But you have to explicitly specify which component will inherit parent’s attributes

“Emits” - new component option

- Define what events your component emits
- Intellisense, documentation

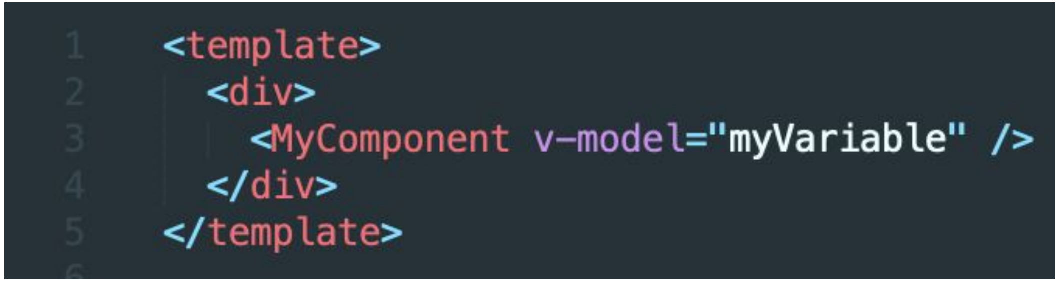


```
<script>
export default {
  name: 'FancyComponent',
  emits: ['submit', 'cancel'],
}
</script>
```

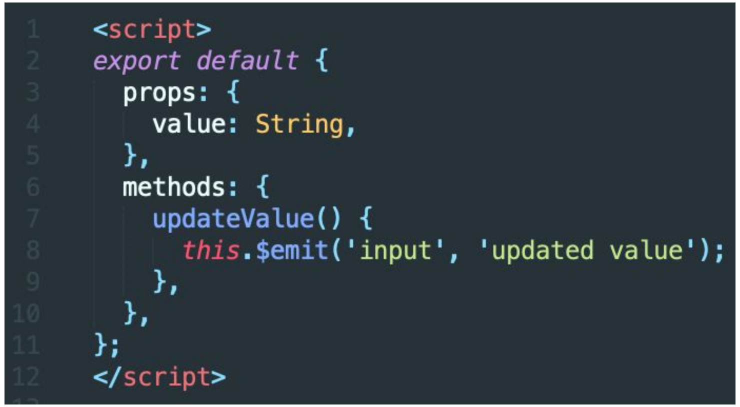
Changes (some of them)

- *v-model* behavior in target component
- *v-if* vs *v-for* precedence
- Removed support for KeyCode modifiers
- Removed filters
- Renamed *destroyed/beforeDestroy* lifecycle callbacks

v-model behavior in target component

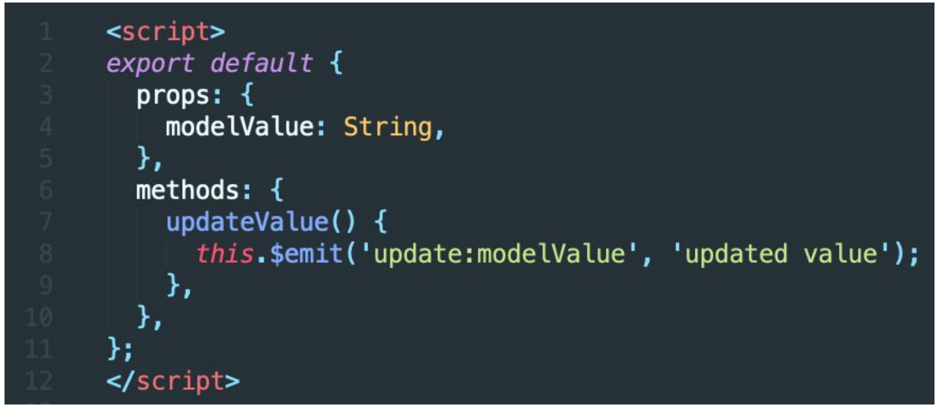


```
<template>
  <div>
    <MyComponent v-model="myVariable" />
  </div>
</template>
```



```
<script>
export default {
  props: {
    value: String,
  },
  methods: {
    updateValue() {
      this.$emit('input', 'updated value');
    },
  },
};
</script>
```

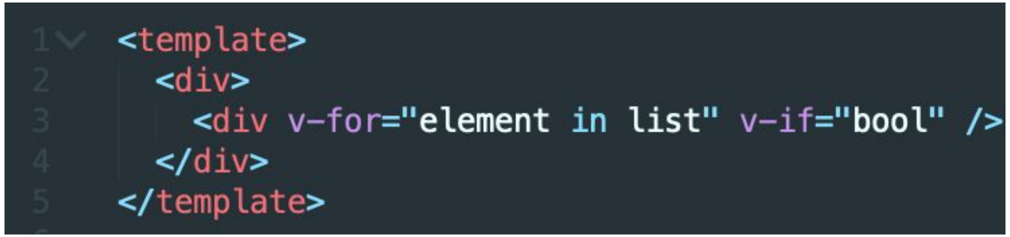
Vue 2



```
<script>
export default {
  props: {
    modelValue: String,
  },
  methods: {
    updateValue() {
      this.$emit('update:modelValue', 'updated value');
    },
  },
};
</script>
```

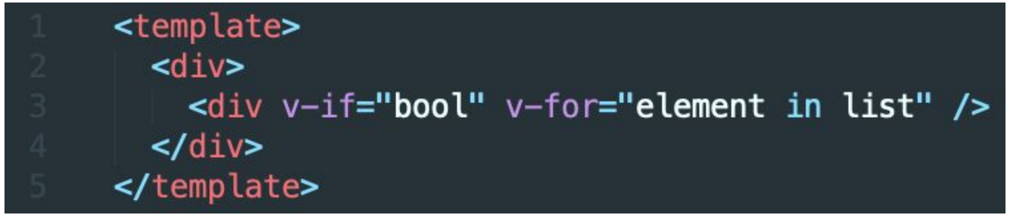
Vue 3

v-if vs *v-for* precedence



```
<template>
  <div>
    <div v-for="element in list" v-if="bool" />
  </div>
</template>
```

Vue 2



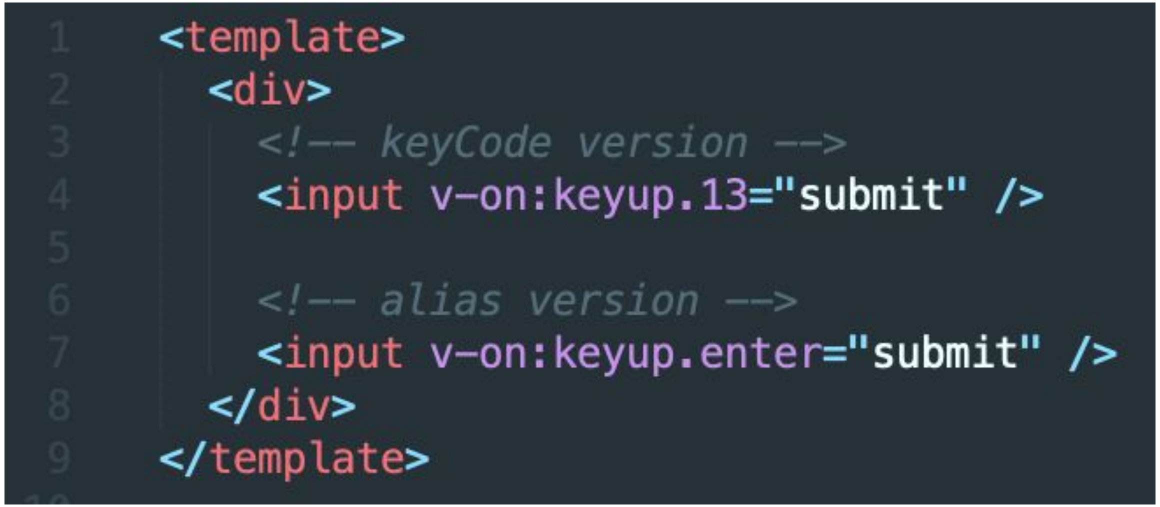
```
<template>
  <div>
    <div v-if="bool" v-for="element in list" />
  </div>
</template>
```

Vue 3

Don’t use neither!

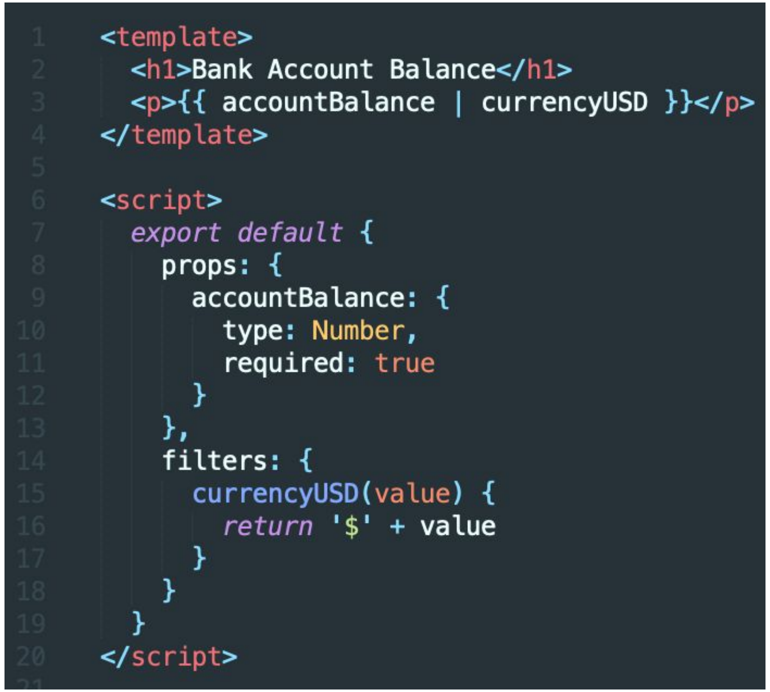
Removed support for KeyCode modifiers

- *KeyboardEvent.keyCode* is deprecated, so Vue dropped support
- Alias is better anyway



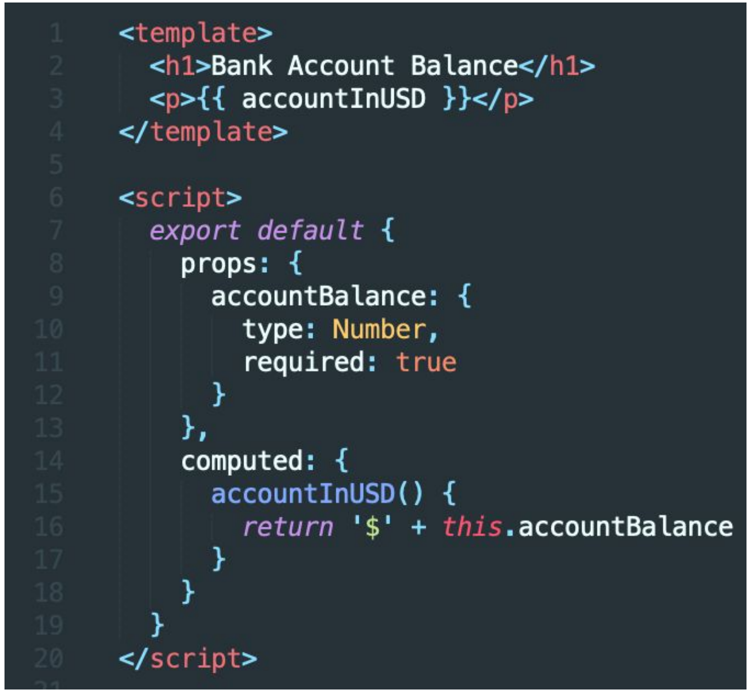
```
<template>
  <div>
    <!-- keyCode version -->
    <input v-on:keyup.13="submit" />

    <!-- alias version -->
    <input v-on:keyup.enter="submit" />
  </div>
</template>
```

Removed filters



```
<template>
  <h1>Bank Account Balance</h1>
  <p>{{ accountBalance | currencyUSD }}</p>
</template>

<script>
  export default {
    props: {
      accountBalance: {
        type: Number,
        required: true
      }
    },
    filters: {
      currencyUSD(value) {
        return '$' + value
      }
    }
  }
</script>
```

Vue 2

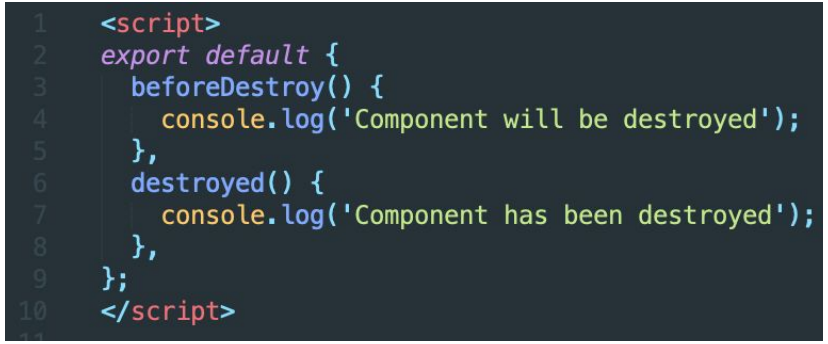


```
<template>
  <h1>Bank Account Balance</h1>
  <p>{{ accountInUSD }}</p>
</template>

<script>
  export default {
    props: {
      accountBalance: {
        type: Number,
        required: true
      }
    },
    computed: {
      accountInUSD() {
        return '$' + this.accountBalance
      }
    }
  }
</script>
```

Vue 3

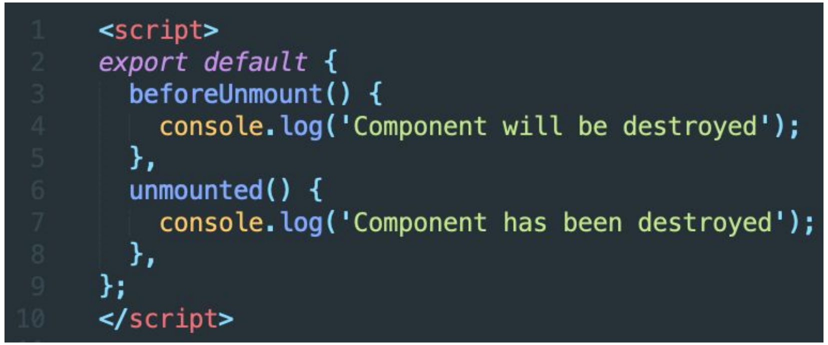
VISMA

Renamed lifecycle callbacks



```
<script>
export default {
  beforeDestroy() {
    console.log('Component will be destroyed');
  },
  destroyed() {
    console.log('Component has been destroyed');
  },
};
</script>
```

Vue 2



```
<script>
export default {
  beforeUnmount() {
    console.log('Component will be destroyed');
  },
  unmounted() {
    console.log('Component has been destroyed');
  },
};
</script>
```

Vue 3

Conclusion

- It’s still the same, beloved Vue
- There are no fundamental changes in the Vue 2 API
- No new paradigm is being enforced - both TypeScript and Composition API are optional
- New features are interesting and helpful
- Removed features were usually unused

Should I migrate?

Well yes, but actually no

Facts

- Vue 3.0 is stable
- Official libraries offer full support
- Vue 2 is still recommended for new projects
- Other libraries might take time to migrate
    - (Vuetify Q3 2021)

My recommendations

- Small to medium projects - go ahead!
- Enterprise projects
  - Get familiar with breaking changes
  - Refactor code to match Vue 3 standards
  - Check dependencies for upgrade timelines
  - Wait for migration build
- You can always start with Vue 2 and migrate when ready

Questions? :)

VISMA

Respect

Reliability

Innovation

Competence

Team spirit