Introducing Vue.js

The progressive framework

Introducing Vue.js

The progressive framework

Thorsten Lünborg

GitHub: LinusBorg

Twitter: @linus_borg

Vue.js Core Team

Who has heard of Vue.js?

Who has played with Vue.js?

Who has worked with Vue.js?

- The library: How to build UIs with Vue.js
- The framework: How to build apps with Vue.js
- The ecosystem: Beyond Vue itself
- Where we’re heading: Vue 3.0

The Vue.js project

Timeline

- 2014: 0.11 released
- October 2015: 1.0 released
- October 2016 2.0 released
- October 2018: Conceptul work in 3.0 started
- Today: https://github.com/vuejs/rfcs

Stats

February 2019

~700k weekly active devtool users

~800k weekly downloads on NPM

~461M hits/month on jsDelivr CDN

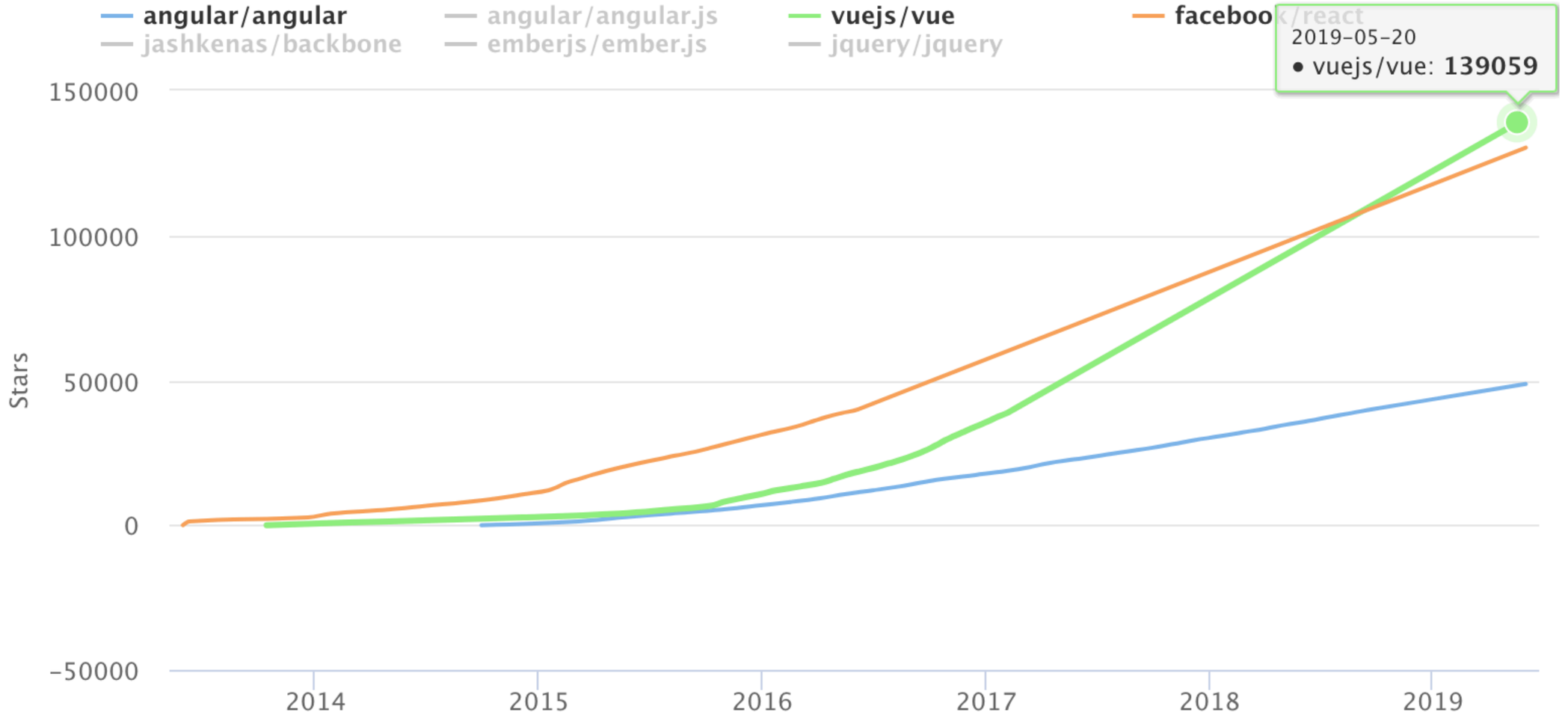
Stats

Today

~900k weekly active devtool users

~1M weekly downloads on NPM

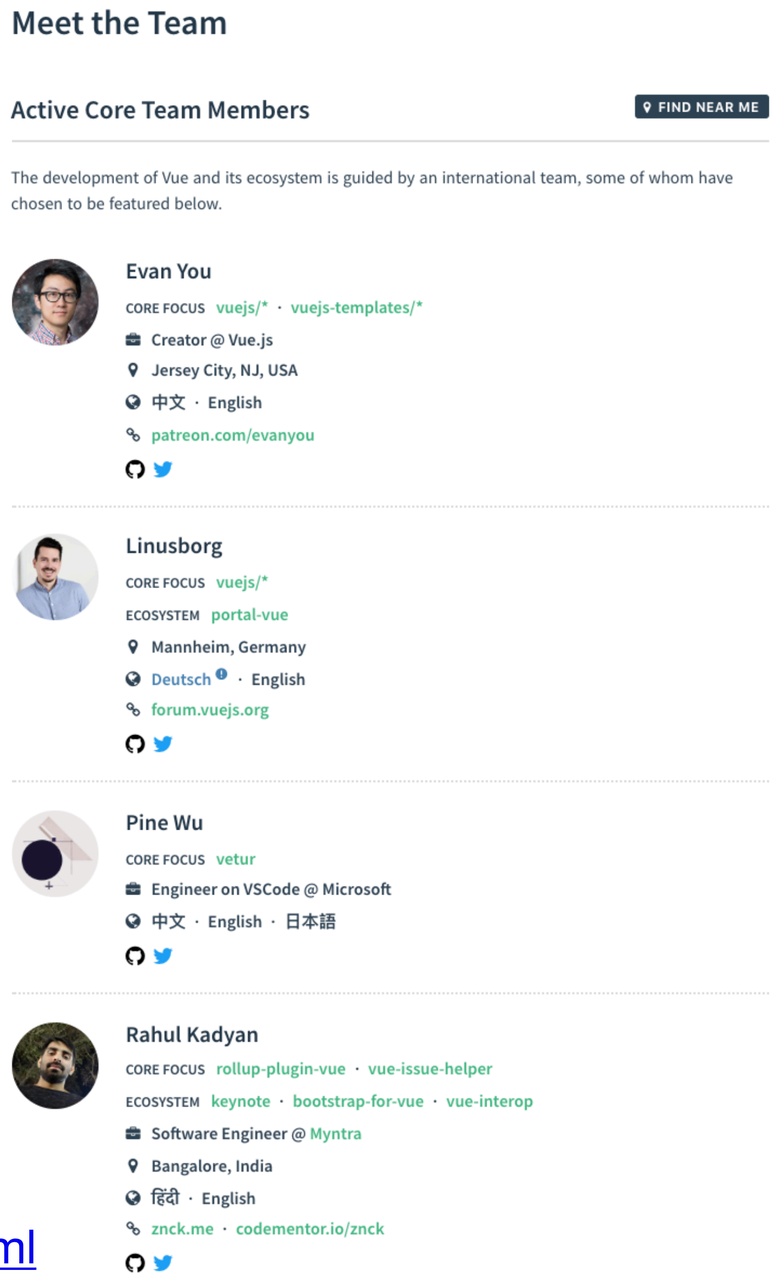
~722M hits/month on jsDelivr CDN



★ Unstar 141,918

Team

- 21 core contributors
- 6 Emeriti
- 33 community partners
- 276 contributors to the core lib alone



Meet the Team

Active Core Team Members

FIND NEAR ME

The development of Vue and its ecosystem is guided by an international team, some of whom have chosen to be featured below.

Evan You
CORE FOCUS vuejs/* · vuejs-templates/*
Creator @ Vue.js
Jersey City, NJ, USA
中文 · English
patreon.com/evanyou

Linusborg
CORE FOCUS vuejs/*
ECOSYSTEM portal-vue
Mannheim, Germany
Deutsch · English
forum.vuejs.org

Pine Wu
CORE FOCUS vetur
Engineer on VSCode @ Microsoft
中文 · English · 日本語

Rahul Kadyan
CORE FOCUS rollup-plugin-vue · vue-issue-helper
ECOSYSTEM keynote · bootstrap-for-vue · vue-interop
Software Engineer @ Myntra
Bangalore, India
हिंदी · English
znck.me · codementor.io/znck

https://vuejs.org/v2/guide/team.html

Output

@vue/jsx

CLI

VueRouter

vue-loader

Vue.js

vue-devtools

TestUtils

Vuex

eslint-plugin-vue

rollup-plugin-vue

Philosophy

The progressive framework

- Lightweight core library
- Runs in the browser without tooling
- Official Extensions: Sensible defaults for common needs (routing, state)
- Additional Tooling: for complex projects and big teams

Vue can scale and grow with any project

So who’s using this?

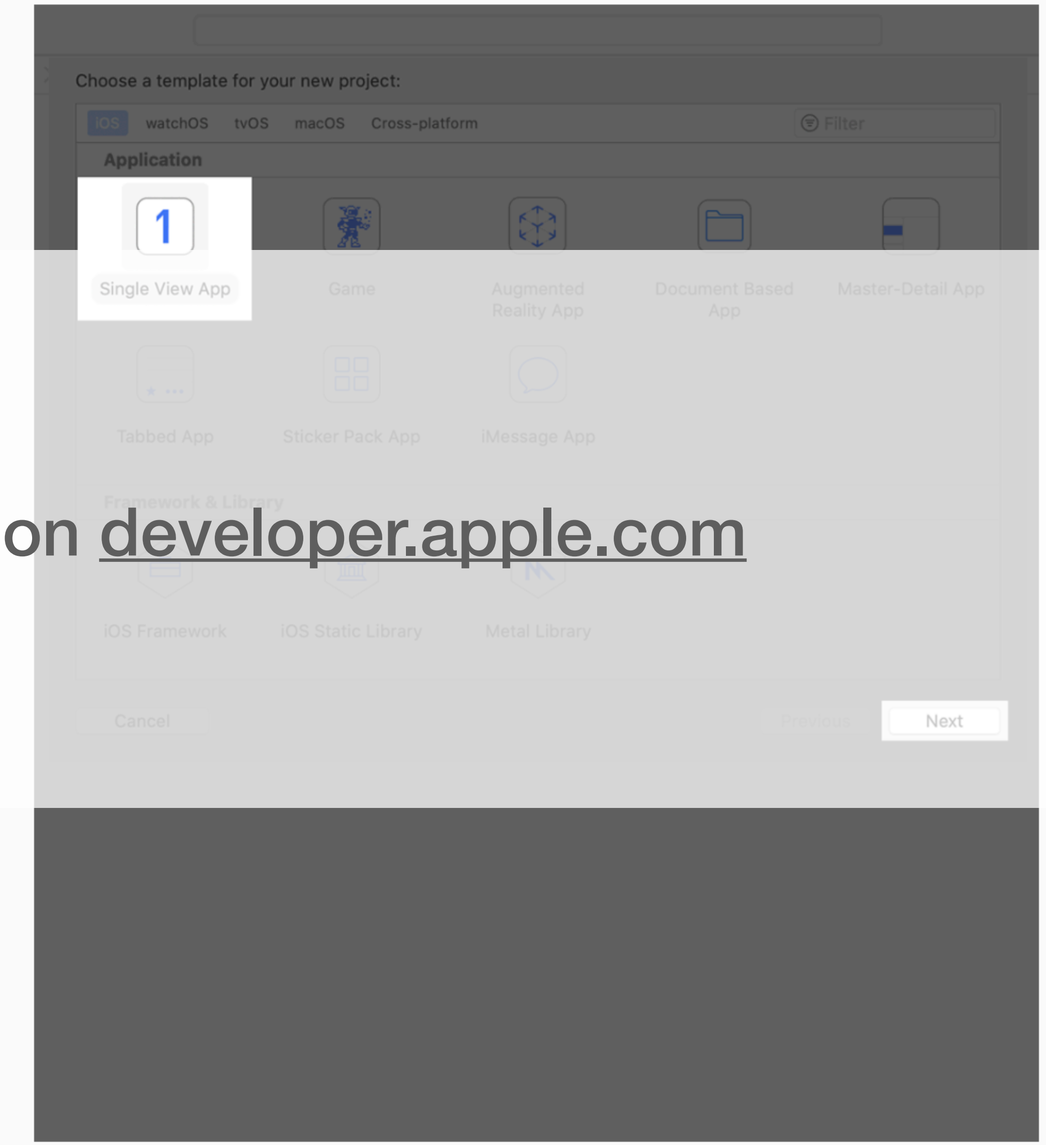
choose **File > New > Project**.

Step 2

In the template selector, select **iOS** as the platform, select the **Single View App** tem click **Next**.

Step 3

Enter “Landmarks” as the Product Name, select the **Use SwiftUI** checkbox, and click **Next**. Choose a location to save the Landmarks project on your Mac.



Swift UI Guides on developer.apple.com

https://developer.apple.com/tutorials/swiftui/creating-and-combining-views

Wohnzimmer

News & Trends | Sofas | Sessel | Hocker | Stühle | Tische | Schränke | Regale | TV-Möbel

Teppiche | Textilien | Beleuchtung | Dekoration

Produkte | Inspirationen | Produktplaner

IKEA

Raumwelten auf ikea.de

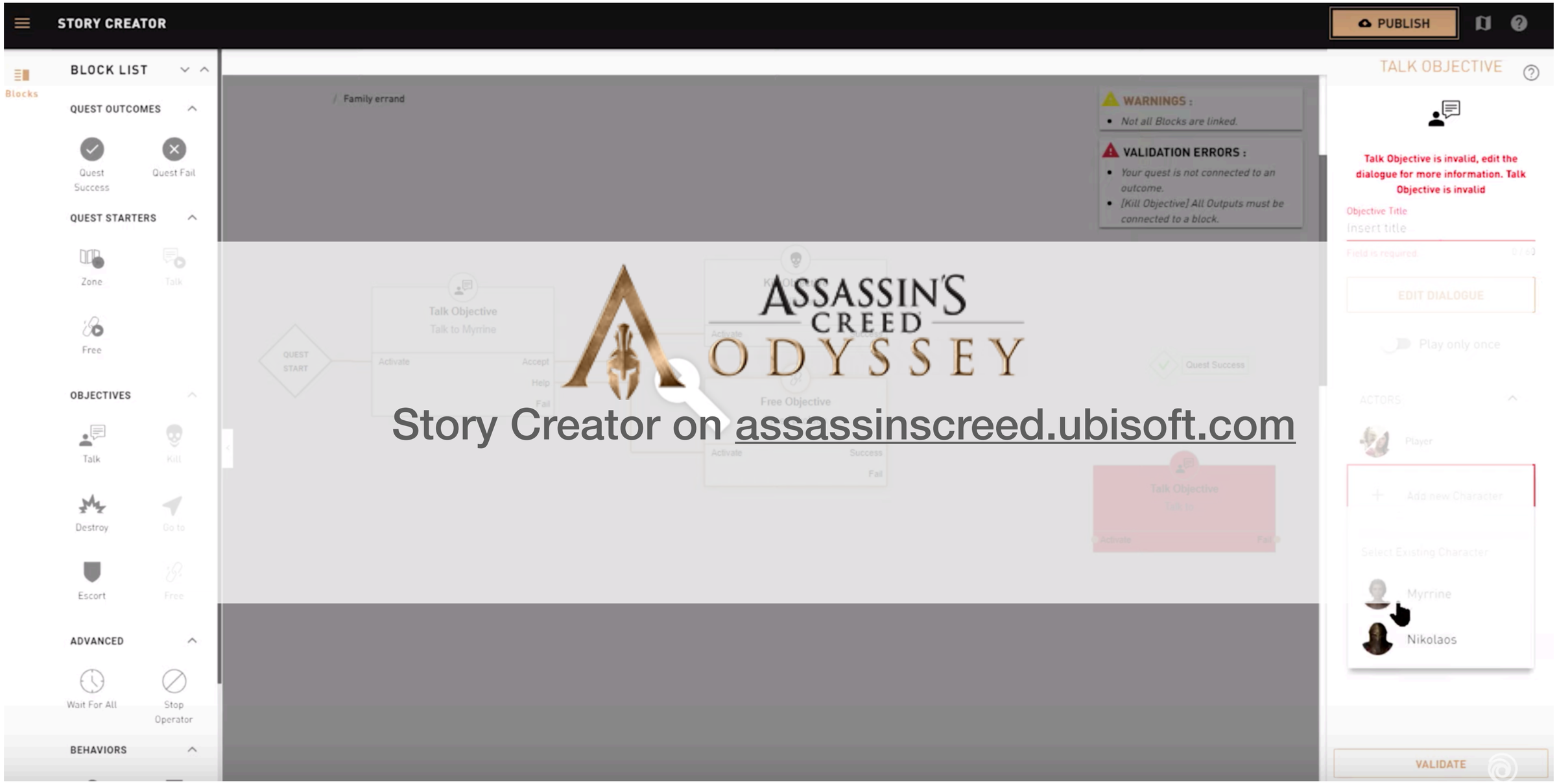
Alle Stühle | Esszimmerstühle | Polsterstühle | Klappstühle | Drehstühle | Kinderstühle

Preis | Farbe | Serien

406 Ergebnisse

Stühle

Relevanz



https://assassinscreed.ubisoft.com/story-creator-mode/en-us/

- User-facing web app
- Internal dashboards
- PWAs for service people on the streets

…and a few others

Ressources

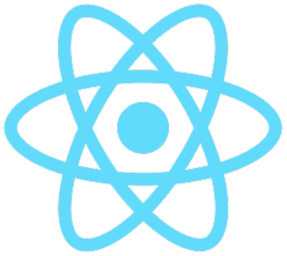
DEV

- Read the official guide
- Or this intro by Sarah Drasner
- Plenty of articles to find online
- Great Video courses as well

Building User Interfaces with Vue.js

What is Vue.js, exactly?

- Library for building UI
- Declarative Rendering
- Components-based
- Reactive Data (*not* RxJS)
- small core, great performance
- progressive extensibility through official modules and tooling



Minimal Example

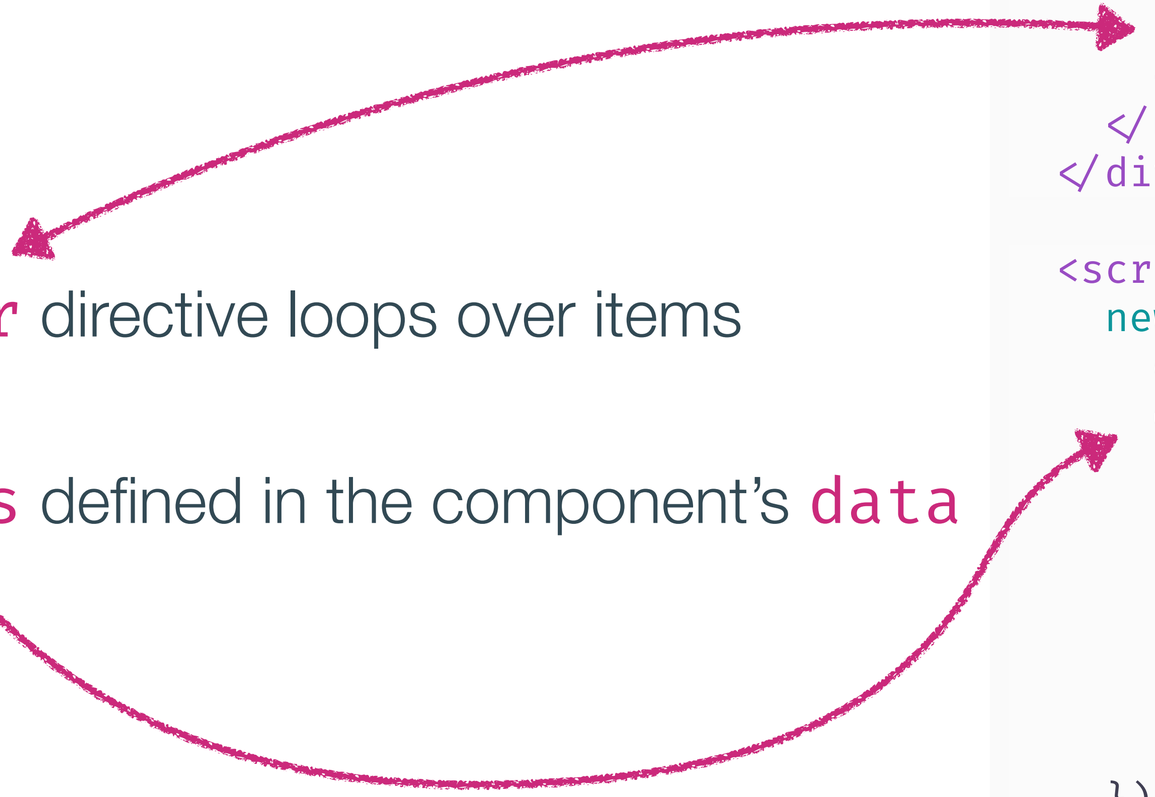
- Create the App’s mounting element
  - Content will be used as the template
- Create new Vue instance
  - mounts to that element
- Instance’ `data` option defines data for template

```
<body>
  <div id="app">
    {{ text }} Nice to meet Vue.
  </div>

  <script>
  new Vue({
    el: '#app',
    data: {
      text: 'Hello Class!',
    },
  })
  </script>

<script
  src=„https://unpkg.com/vue/dist/vue.js“
>
</script>
</body>
```

Integrate directly form CDN

Rendering Lists

- `v-for` directive loops over items
- `items` defined in the component's `data`



```
<body>
  <div id="app">
    <ul>
      <li v-for="item in items">
        {{ item }}
      </li>
    </ul>
  </div>

  <script>
    new Vue({
      el: '#app',
      data: {
        items: [
          'thingie',
          'another thingie',
          'lots of stuff',
          'yadda yadda'
        ]
      }
    });
  </script>
</body>
```

Declarative Rendering

```
const items = [
  'thingie',
  'another thingie',
  'lots of stuff',
  'yadda yadda'
];

function listOfStuff() {
  let full_list = '';
  for (let i = 0; i < items.length; i++) {
      full_list = full_list + `<li> $
{items[i]} </li>`
  }
  const contain =
    document.querySelector('#container');
  contain.innerHTML =
    `<ul> ${full_list} </ul>`;
}

listOfStuff();
```

```
<body>
  <div id="app">
    <ul>
      <li v-for="item in items">
        {{ item }}
      </li>
    </ul>
  </div>

  <script>
    new Vue({
      el: '#app',
      data: {
        items: [
          'thingie',
          'another thingie',
          'lots of stuff',
          'yadda yadda'
        ]
      }
    });
  </script>
```

Form Inputs

- v-model allows two-way bindings
- modifiers like `.trim` provide great DX

```
<body>
  <div id="app">
    <h3>Type here:</h3>
    <textarea
      v-model.trim=„message"
      class="message"
      rows="5"
      maxlength="72"
    ></textarea
    ><br />
    <p class="booktext">{{ message }}</p>
  </div>

  <script>
  new Vue({
    el: '#app',
    data() {
      return {
        message: 'This is great'
      }
    }
  });
  </script>
</body>
```

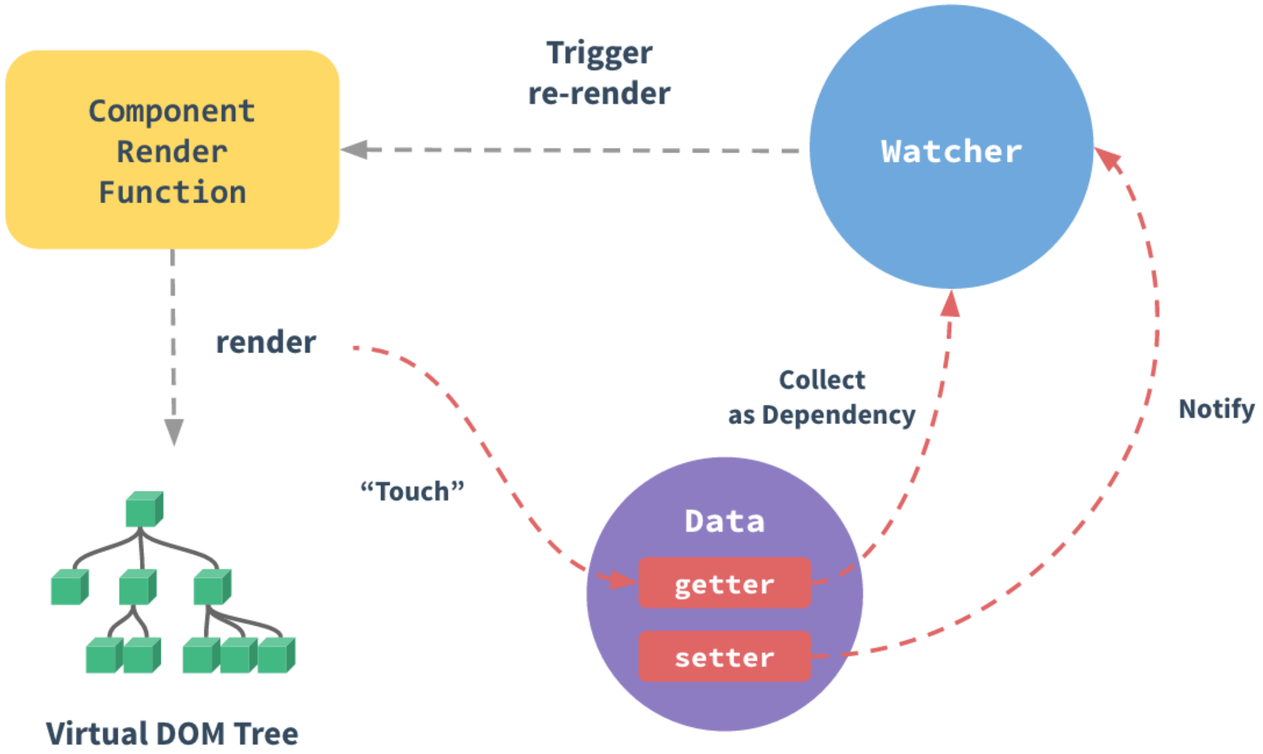
Event handling

`v-on` & methods

- Event listener can be added with `v-on`
- Attribute value can contain:
  - Name of method to use as handler
  - Inline expression (`count++`)

```
<body>
  <div id="app">
    <p><button v-on:click="increment">+</
button> {{ counter }}</p>
  </div>

  <script>
  new Vue({
    el: '#app',
    data() {
      return {
        counter: 0,
      }
    },
    methods: {
      increment() {
        this.counter++
      },
    },
  })
  </script>
</body>
```

Reactivity



- Vue tracks changes to data that's used by the component
- When changes occur, Vue updates the component by re-rendering
- Necessary changes to the DOM are applied by diffing the Virtual DOM

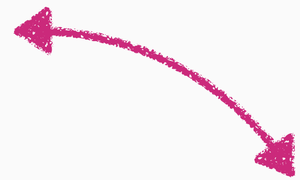
Reactivity

Computed Properties

- pure functions
- lazily evaluated when accessed
- return value is cached
- When reactive dependencies change, computed property is re-evaluated

Good for computing deprived values

```
<body>
  <div id="app">
    <button @click=„counter++">+</button>
    <p>
      {{ counter }}<br>
      {{ plusOne }}
    </p>
  </div>
</body>

  <script>
  new Vue({
    el: '#app',
    data() {
      return {
        counter: 0,
      }
    },
    computed: {
      plusOne() { return this.counter +1 }
    }
  }
  </script>
</body>
```



Reactivity

Watchers

- Function name defines the dependency to watch
- Changes to watched data trigger Watcher
- Handler function is run, receiving new and previous value

Good for running side effects

```
<body>
  <div id="app">
    <button @click="counter++">+</button>
    <p>
      {{ counter }}<br>
      {{ plusOne }}
    </p>
  </div>

  <script>
  new Vue({
    el: '#app',
    data() {
      return {
        counter: 0,
      }
    },
    watch: {
      counter(newValue) {
        console.log(`count: ${newValue}`)
      }
    }
  </script>
</body>
```

Benefits

- Declarative Rendering
- Approachable Template Syntax
- Data Reactivity provides easy update mechanism

- Small (23kb min+gzip)
- Easily integratable from CDN

„That’s cool. But what if it gets more complicated?“

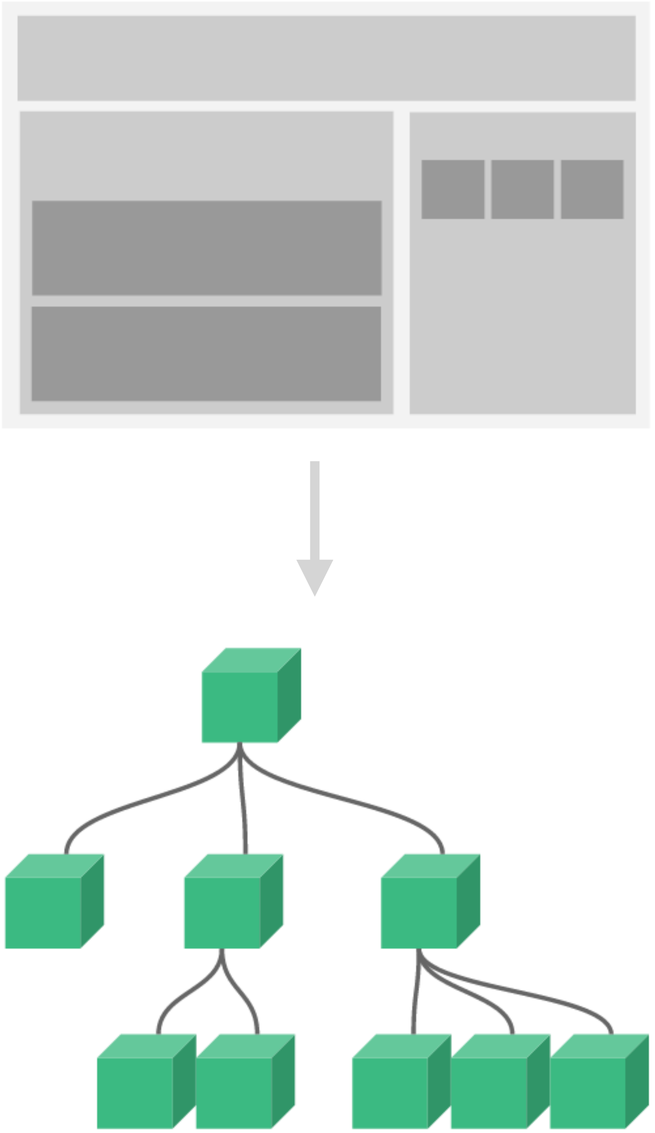
– (some of) you, probably

Components

Components

UI Building blocks

- Some good points
- With code snippet



Components

Definition

- Register components globally or locally
- Templates can be put in the component or extracted into <template> elements

```
<template id=„counter“>
  <button v-on:click="count++">
    You clicked me {{ count }} times.
  </button>
</template>

<script>
  Vue.component('button-counter', {
    template: '#counter'
    data: function() {
      return {
        count: 0,
      }
    },
  })
</script>
```

Components

Usage

- Components can be used like elements
- Allows encapsulating pieces of UI and their logic

```
<body>
  <div id="app">
    <button-counter></button-counter>
  </div>

  <script>
    Vue.component('button-counter',/*  ...  */)

    new Vue({
      el: '#app',
    })
  </script>
</body>
```

Props & Events

Props

- We use Props to pass down state to a child component.
- We can pass down anything from primitive types to complex objects or functions

```
Vue.component('child', {
  props: ['text'],
  template: `<div>{{ text }}<div>`
});

new Vue({
  el: "#app",
  data() {
    return {
      message: 'hello mr. magoo'
    }
  }
});
```

Props & Events

Props

- We use Props to pass down state to a child component.
- We can pass down anything from primitive types to complex objects or functions

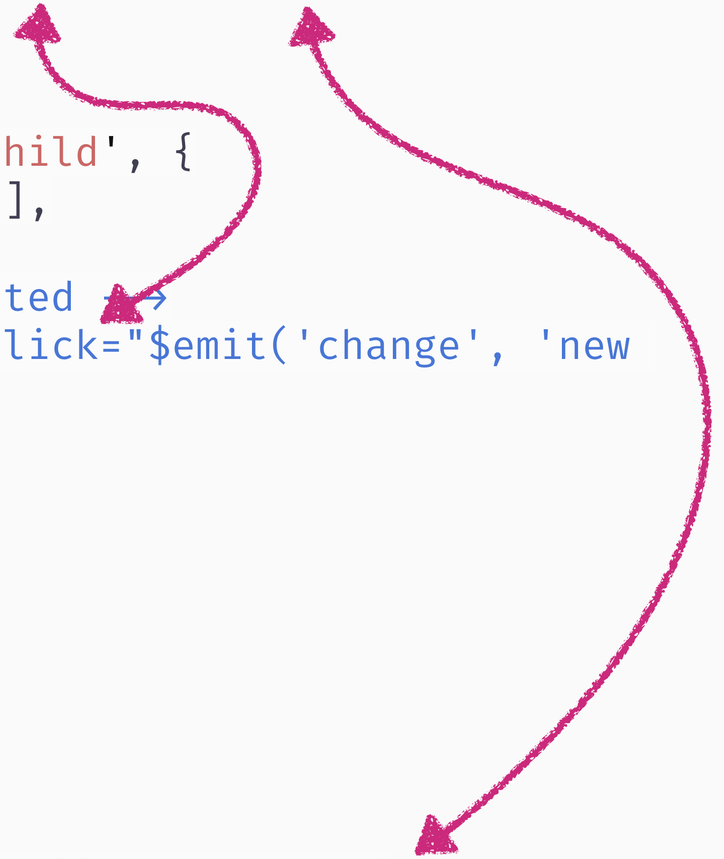
```
<template>
  <div id="app">
    <child :text="message"/>
  </div>
</template>

<script>
Vue.component('child', {
  props: ['text'],
  template: `<div>{{ text }}<div>`
});

new Vue({
  el: "#app",
  data() {
    return {
      message: 'hello mr. magoo'
    }
  }
});
</script>
```

Props & Events

Events

- We don’t recommend mutating a parent’s state in the child
- We use events to communicate changes to the parent component
- The `v-on` syntax is like for element events
- From the child, we use `this.$emit()` to emit events to the parent

```
<template>
  <div id="app">
    <child :text="message"
           v-on:change="message = $event" />
  </div>
</template>
<script>
Vue.component('child', {
  props: ['text'],
  template: `
  <!-- rest omitted -->
  <button v-on:click="$emit('change', 'new
Message')">
    Emit Event
  </button>
  `
});

new Vue({
  el: "#app",
  data() {
    return {
      message: 'hello mr. magoo'
    }
  }
});
</script>
```



Single File Components

- Custom Format for components
- Allows to collocate
  - Template
  - Javascript
  - Styles

- Tooling to convert to JS is provided

ButtonCounter.vue

```
<template>
  <button v-on:click="count++" class="counter-btn">
    You clicked me {{ count }} times.
  </button>
</template>

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

<style lang="scss" scoped>
  .counter-btn {
    border-radius: 3px;
  }
</style>
```

Tooling for SFCs

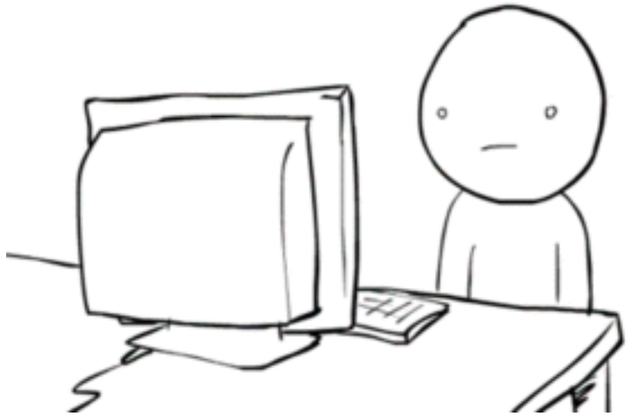
vue-compiler-utils

rollup-plugin-vue

vue-loader

vue-jest

Separation of Concerns?



Separation of ~~Technologies~~!

- HTMl, JS and CSS for one component are one concern
- SFCs bundle code by functionality, not technology

Typescript

…it’s complicated

vue-class-component

https://github.com/vuejs/vue-class-component

- TS Support with the default syntax is limited
- `vue-class-component` provides type safety
- …at a runtime cost

```
import { Component } from 'vue-class-component'

@Component({
  props: {
    propMessage: String
  }
})
class MyComponent extends Vue {
  // initial data
  msg = 123

  // use prop values for initial data
  helloMsg = 'Hello, ' + this.propMessage

  // computed
  get computedMsg () {
    return 'computed ' + this.msg
  }

  // method
  greet () {
    alert('greeting: ' + this.msg)
  }
}
```

…but it gets better in Vue 3

more about that later …

Building User Interfaces with Vue.js

Building Applications with Vue.js

The progressive framework

- Some good points
- With code snippet

VueRouter

Client-Side Routing

router.vuejs.org

VueRouter

Usage

```
<div id="app">
  <h1>Hello App!</h1>
  <p>
    <!-- use router-link component for navigation. -->
    <!-- specify the link by passing the `to` prop. -->
    <!-- `<router-link>` will be rendered as an `<a>` tag by
default -->
    <router-link to="/foo">Go to Foo</router-link>
    <router-link to="/bar">Go to Bar</router-link>
  </p>
  <!-- route outlet -->
  <!-- component matched by the route will render here -->
  <router-view></router-view>
</div>
```

VueRouter

Config

```
// 1. Define route components.
// These can be imported from other files
const Foo = { template: '<div>foo</div>' }
const Bar = { template: '<div>bar</div>' }
```

VueRouter

Config

```
// 1. Define route components.
// These can be imported from other files
const Foo = { template: '<div>foo</div>' }
const Bar = { template: '<div>bar</div>' }

// 2. Define some routes
// Each route should map to a component.
const routes = [
  { path: '/foo', component: Foo },
  { path: '/bar', component: Bar }
]
```

VueRouter

Config

```
// 1. Define route components.
// These can be imported from other files
const Foo = { template: '<div>foo</div>' }
const Bar = { template: '<div>bar</div>' }

// 2. Define some routes
// Each route should map to a component.
const routes = [
  { path: '/foo', component: Foo },
  { path: '/bar', component: Bar }
]

// 3. Create the router instance and pass the
`routes` option
const router = new VueRouter({
  routes // short for `routes: routes`
})
```

VueRouter

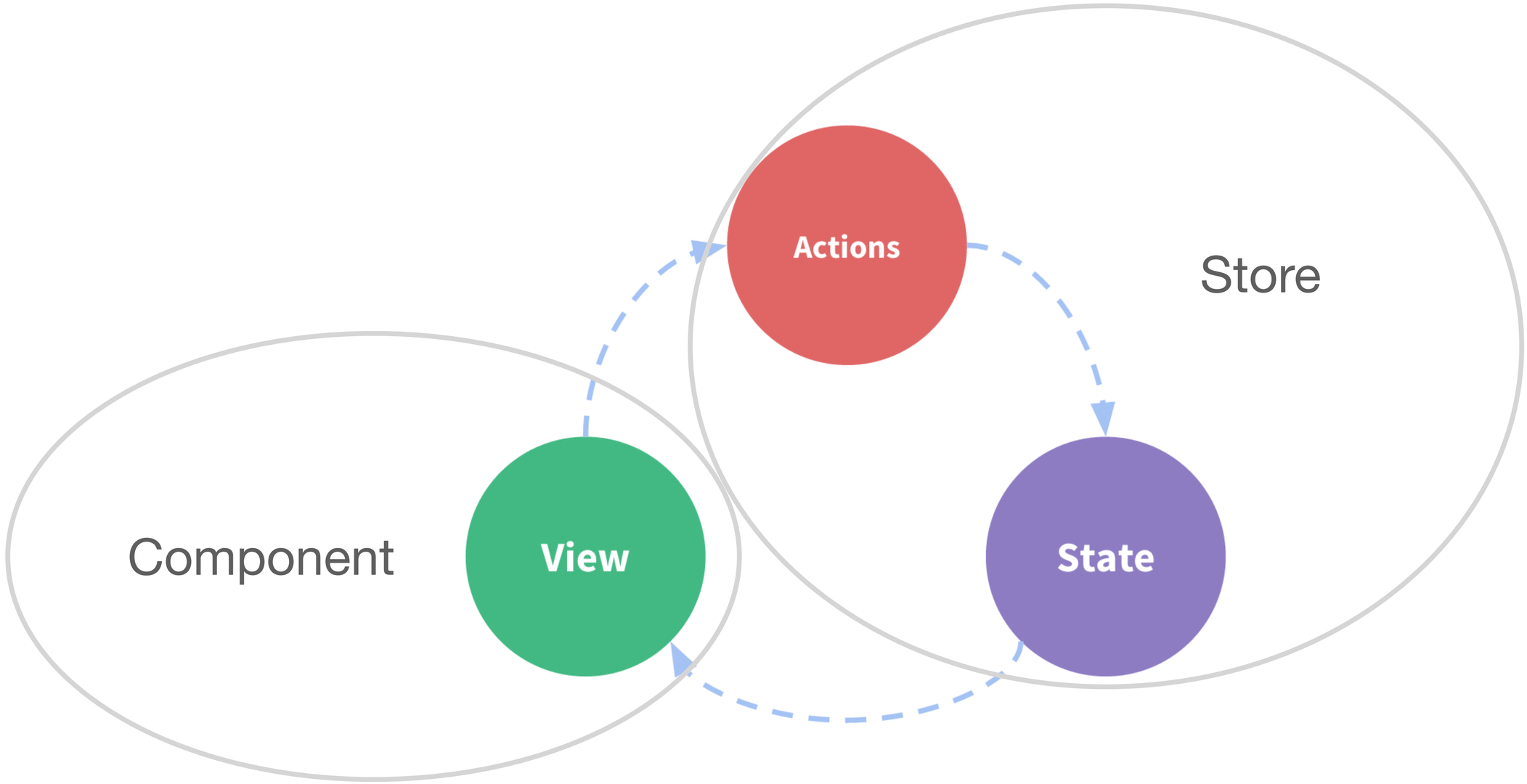
Config

```
// 1. Define route components.
// These can be imported from other files
const Foo = { template: '<div>foo</div>' }
const Bar = { template: '<div>bar</div>' }

// 2. Define some routes
// Each route should map to a component.
const routes = [
  { path: '/foo', component: Foo },
  { path: '/bar', component: Bar }
]

// 3. Create the router instance and pass the `routes`
option
const router = new VueRouter({
  routes // short for `routes: routes`
})

// 4. Create and mount the root instance.
const app = new Vue({
  router
}).$mount('#app')
```

Vuex

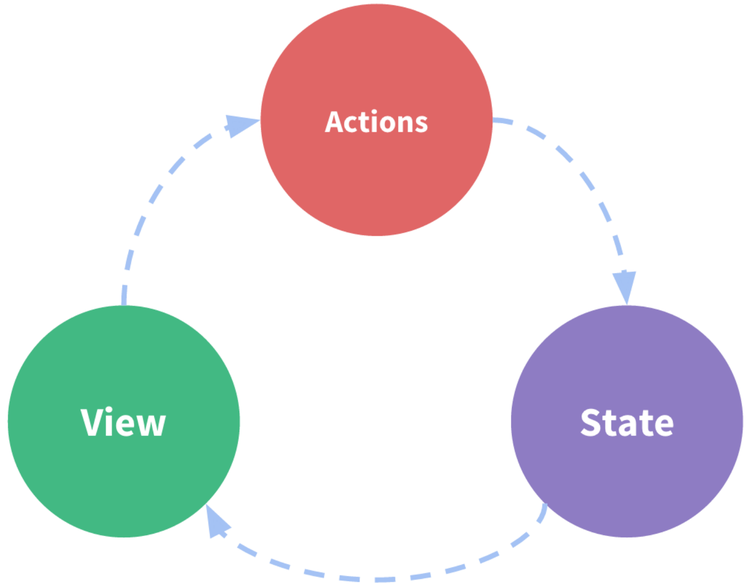
State Management

vuex.vuejs.org


Actions
Store
Component
View
State

Global State Management

- inspired by the „Flux“ Pattern
- Handling global state in a central placed
- Similiar solutions in other frameworks:
  - Redux
  - MobX



Vuex

Creating a store

```
const store = new Vuex.Store({
  state: {
    count: 0
  },
  mutations: {
    increment (state) {
      state.count++
    }
  }
})

store.commit('increment')

console.log(store.state.count) // → 1
```

Adding the store to to your app

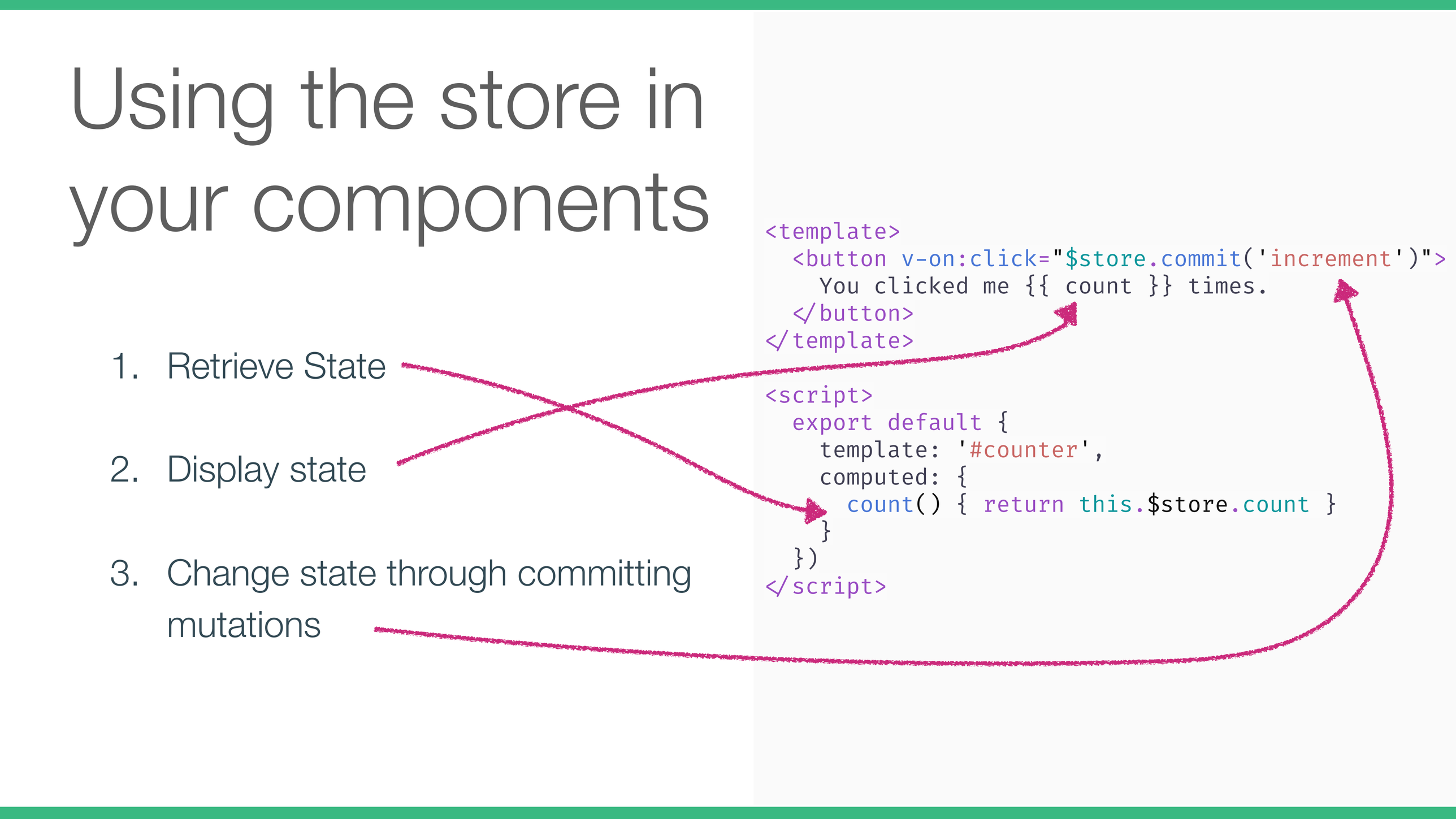
1. Install the plugin
2. Inject the store in the root instance

```
// Add Vuex as a Plugin to Vue
Vue.use(Vuex)

const app = new Vue({
  el: '#app',
  // add the store using the "store" option.
  // this will inject the store instance
  // to all child components.
  store,
  components: { Counter },
  template: `
    <div class="app">
      <counter></counter>
    </div>
  `
})
```

Using the store in your components

1. Retrieve State
2. Display state
3. Change state through committing mutations

```
<template>
  <button v-on:click="$store.commit('increment')">
    You clicked me {{ count }} times.
  </button>
</template>

<script>
  export default {
    template: '#counter',
    computed: {
      count() { return this.$store.count }
    }
  })
</script>
```



Vue CLI

Standard Tooling for Vue.js Development

cli.vuejs.org

Vue CLI

- Webpack build process abstracted away
  - but configurable
- Plugin-based, extendable architecture
- Setup projects and dynamically define tooling
  -


Tests — -bash — 137×25
Last login: Sun Feb 10 13:02:35 on console
Thorstens-MacBook-Pro:Tests thorstenlunborg$
New recording

Demo

vue-devtools

The Vue.js ecosystem

Frameworks

NUXTJS

- Opinionated Framework for ambitious applications
- supports 3 modes: SPA, SSR & static generation

Gridsome

- Static Site generator using Vue.js components and GraphQL, similar to Gatsby.js
- Plugin adapters for various data sources available

Vuepress

- Status site generator for documentation
- fast, easily themeable, extendable via plugins

Component Libraries

element.eleme.io

https://vuetifyjs.org

Ionic-vue

quasar.dev

Libraries

there’s a lot of 'em

- https://github.com/vue/awesome-vue
- Collection of ressources
  - Tutorials
  - Reference Websites
  - Demo projects
- And of course: libraries, sorted by type

- Components & Libraries
  - UI Components
    - Table
    - Notification
    - Loader
      - Progress Bar
    - Tooltip
    - Overlay
    - Parallax
    - Icons
    - Menu
    - Minus Plus Input
    - Carousel
    - Charts
    - Time
    - Calendar
    - Map
    - Form
      - Phone Number Input
      - Picker
      - Generator
        - Date Picker
      - Select
      - Slider
      - Drag and Drop
      - Autocomplete
      - Type Select
      - Color Picker
      - Switch
      - Masked Input
      - Rich Text Editing
      - Image Manipulation
      - Video Manipulation
      - File Upload
      - Context Menu
      - Miscellaneous
      - Wizard
      - CSV
      - Comment System

…what about Java+Vue?

spring-boot-vuejs

build passing | coverage 88% | license mit | springboot 2.1.6 RELEASE | jdk 8, 9, 11 | nodejs v11.14.0 | vue.js 2.6.10
vue CLI 3.8.4 | webpack 4.28.4 | axios 0.18.0 | jest 23.6.0 | nightwatch 0.9.21

> **If you´re a JavaMagazin / blog.codecentric.de / Softwerker reader**, consider switching to vue-cli-v2-webpack-v3

https://github.com/jonashackt/spring-boot-vuejs

https://spring-boot-vuejs.herokuapp.com/

So … Vue 3.0

Whats up with that?

- Source written in Typescript
- Complete type safety for components with new API
- Improvements across various metrics
- Little to no breaking changes for the typical user
  - Mostly auto-fixable with codemods we provide
- All changes discussed transparently as RFCs

github.com/vuejs/rfcs

Typescript

class API

https://github.com/vuejs/rfcs/pull/17

- Goal: 100% type safety
- without decorators
  - not standardized yet
- …we failed
-

```
export default class App extends Vue {
 count = 0

 created() {
   console.log(this.count)
 }

 get plusOne() {
   return this.count + 1
 }

 increment() {
   this.count++
 }
}
```

Function-based API

- New Proposal, not accepted yet
- based around composing behaviour through functions instead of an options object
- allows for better logic composition and reuse

```vue
<template>
  <div>
    Count is {{ count }}, count * 2 is
{{ double }}
    <button @click="increment">+</button>
  </div>
</template>

<script>
export default {
  data() {
    return {
      count: 0
    }
  },
  methods: {
    increment() {
      this.count++
    }
  },
  computed: {
    double() {
      return this.count * 2
    }
  }
}
</script>
```

```vue
<template>
  <div>
    Count is {{ count }}, count * 2 is
{{ double }}
    <button @click="increment">+</button>
  </div>
</template>

<script>
import { value, computed } from 'vue'
export default {
  setup() {
    const count = value(0)
    const double = computed(() => count.value * 2)
    const increment = () => { count.value++ }
    return {
      count,
      double,
      increment
    }
  }
}
</script>
```

Typescript

- Some good points
- With code snippet

```
import { createComponent } from 'vue'

const MyComponent = createComponent({
  // props declarations are used to infer prop
types
  props: {
    msg: String
  },
  setup(props) {
    props.msg // string | undefined

    // bindings returned from setup() can be used
    // for type inference in templates
    const count = value(0)
    return {
      count
    }
  }
})
```

this broke the internet 4 days ago

635 !!!!!

Function-based Component API 635

#42 opened 18 days ago by yyx990803 updated 1 hour ago From `function-apis`

3.x core

Thanks!