# Build a Vue.js application from scratch

PixelsCamp - 21 March 2019

## Who I am? Why Vue.js?

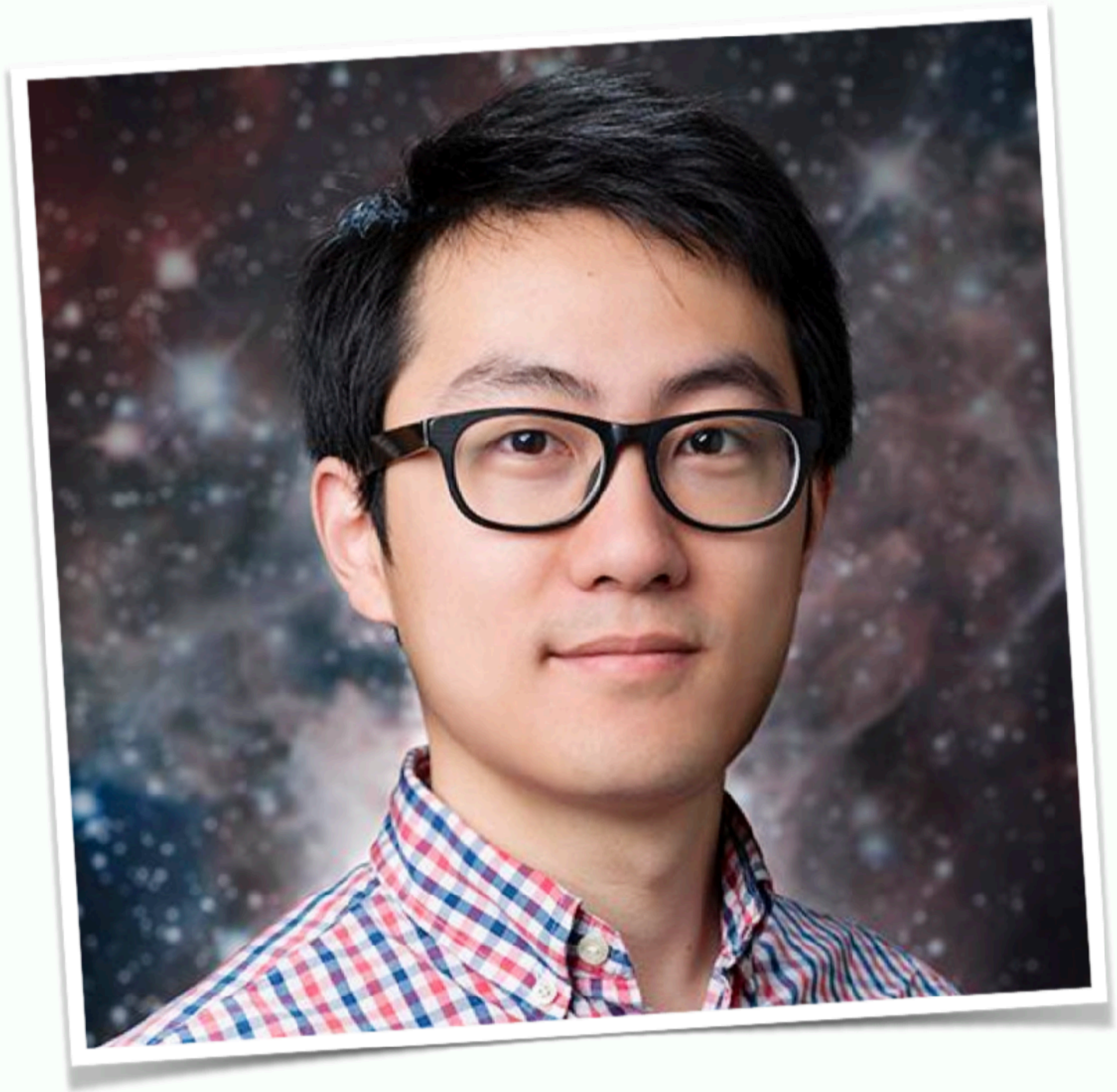
Pixelmatters

- **Tiago Coelho**, Co-CTO at Pixelmatters
- Vue.js v1 in personal projects
- Vue.js v2 in profissional projects

Pixelmatters

## The Founder - Evan You

- Worked at Google and Meteor
- Since 2016 is working full time on Vue.js

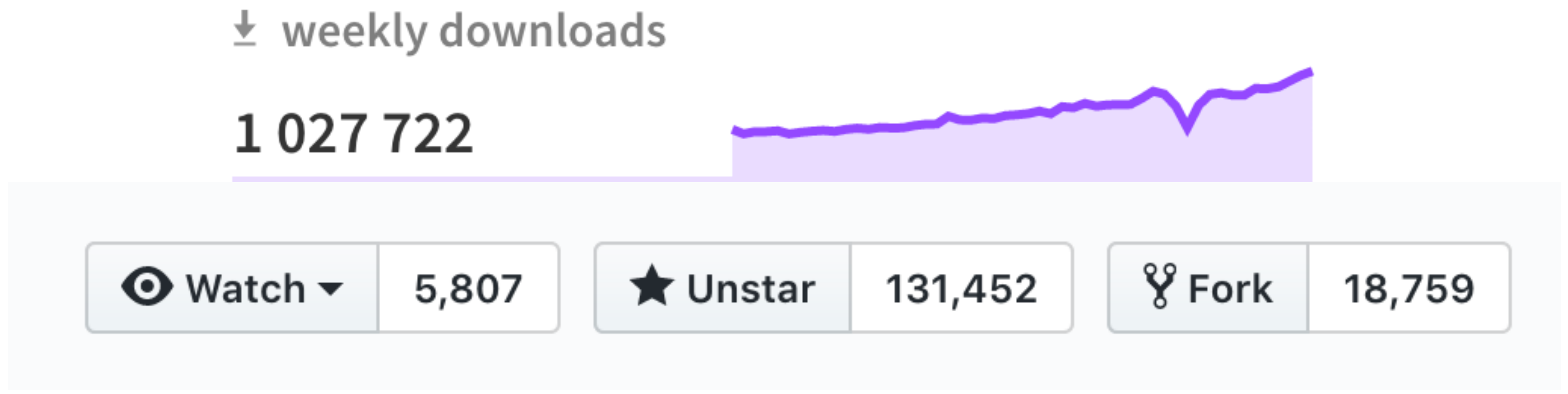


## Vue.js

- Progressive JS framework
- High-performance and small size
- Versatile
- Great community
- First release February 2014
- Latest release this week (v2.6.10)

## Vue.js

- Progressive JS framework
- High-performance and small size
- Versatile
- Great community
- First release February 2014
- Latest release this week (v2.6.10)


weekly downloads

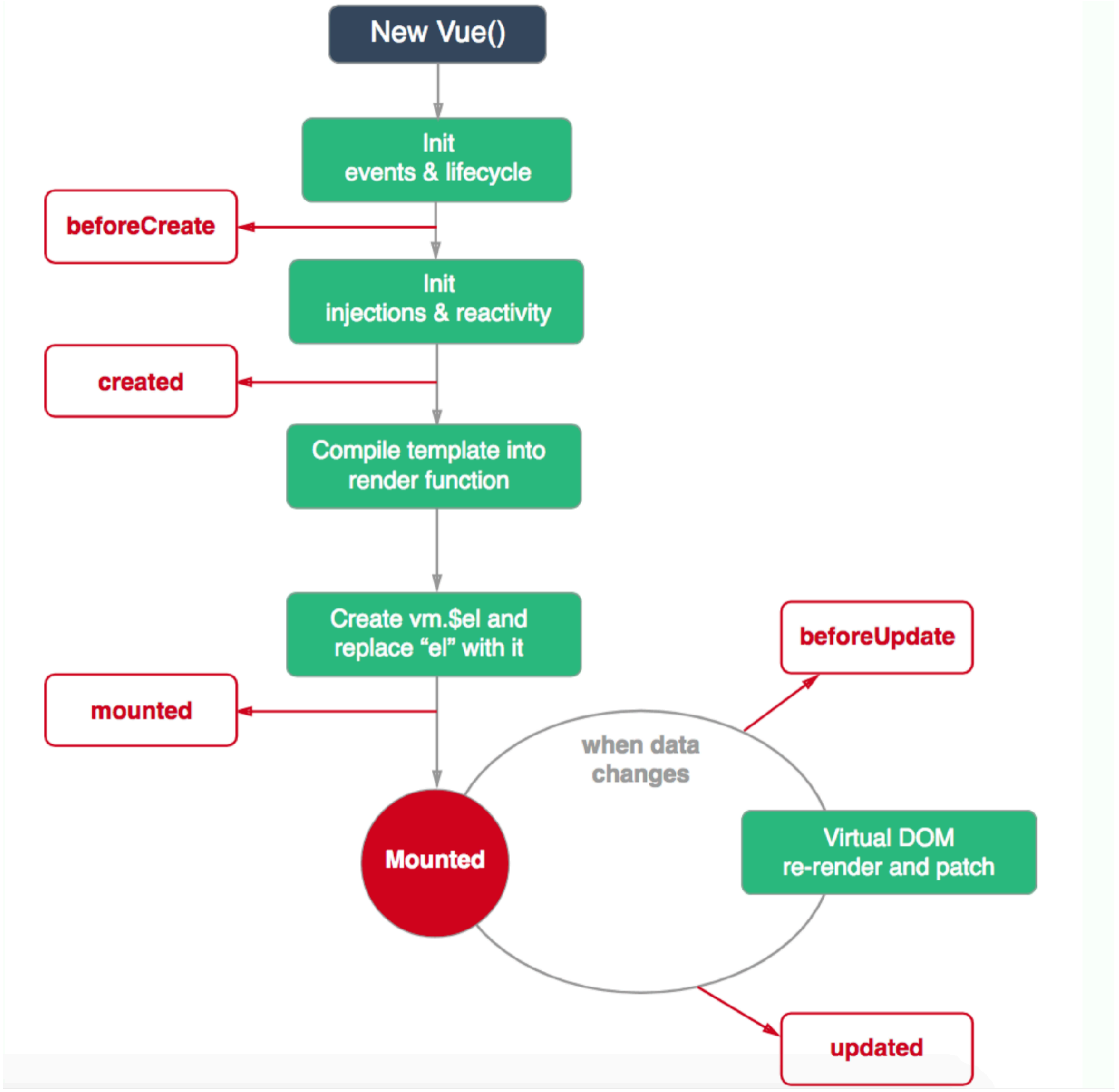
1 027 722

## The Vue instance

- Vue constructor function
- Properties and methods
- Instance lifecycle hooks

```
new Vue({
  // Options
})
```

Pixelmatters


New Vue()
Init events & lifecycle
beforeCreate
Init injections & reactivity
created
Compile template into render function
Create vm.$el and replace "el" with it
mounted
beforeUpdate
when data changes
Mounted
Virtual DOM re-render and patch
updated

Pixelmatters

## Get to the real stuff

Code Examples

## Lifecycle hooks

Pixelmatters

```
<!—— HTML ——>
<div id="app">
</div>
```

```
// Script
new Vue({
  el: '#app',
  created() {},
  mounted() {},
  …
})
```

## Conditional rendering

```html
<!-- HTML -->
<div id="app">
  <span v-if="seen">Now you see me</span>
  <span v-else>Now you don't</span>
</div>
```

```javascript
// Script
new Vue({
  el: '#app',
  data: {
    seen: true
  }
})
```

## Loops

Pixelmatters

```html
<!-- HTML -->
<div id="app">
  <ol>
    <li v-for="todo in todos">
      {{ todo.text }}
    </li>
  </ol>
</div>
```

```javascript
// Script
new Vue({
  el: '#app',
  data: {
    todos: [
      { text: 'Learn JavaScript' },
      { text: 'Learn Vue' },
      { text: 'Build something awesome' }
    ]
  }
})
```

## Directives: v-bind

```html
<!-- HTML -->
<div id="app">

  <!-- full syntax -->
  <img v-bind:src="url" />

  <!-- shorhand -->
  <img :src="url" />
</div>
```

```js
// Script
new Vue({
  el: '#app',
  data: {
    url: 'https://via.placeholder.com/150'
  }
})
```

# Bind HTML classes

Pixelmatters

```html
<!-- HTML -->
<div id="app">

  <div :class="{
      'active': isActive,
      'has-error': hasError
    }">
    Content goes here
  </div>

</div>
```

```javascript
// Script
new Vue({
  el: '#app',
  data: {
    active: true,
    hasError: false
  }
})
```

## Directives: v-on

```html
<!-- HTML -->
<div id="app">

  <!-- full syntax -->
  <img v-on:click="log()" />

  <!-- shorhand -->
  <img @click="log()" />
</div>
```

```js
// Script
new Vue({
  el: '#app',
  methods: {
    log() {
      console.log('Clicked');
    }
  }
})
```

## Modifiers

```
<!—— HTML ——>
<div id="app">

  <!—— the click event's propagation will be stopped ——>
  <a v-on:click.stop="doThis"></a>

  <!—— the submit event will no longer reload the page ——>
  <form v-on:submit.prevent="onSubmit"></form>

  <!—— modifiers can be chained ——>
  <a v-on:click.stop.prevent="doThat"></a>

  <!—— only call `submit()` method when the `key` is `Enter` ——>
  <!—— other modifiers are available as .tab, .esc, .space, etc ——>
  <input v-on:keyup.enter="submit">

  <!—— Alt + C, nested multipliers —>
  <input @keyup.alt.67="clear">

</div>
```

## Single File Components (.vue files)

```
<template>
  <div class="component">

  </div>
</template>

<script>
  export default {}
</script>

<style>
  .component { }
</style>
```

## Single File Components (.vue files)

Pixelmatters

```
<template>
  <div class="component1">

  </div>
</template>

<script>
  export default {}
</script>

<style>
  .component1 { }
</style>
```

```
<template>
  <div class="component2">
    <component1/>
  </div>
</template>

<script>
  import component1 from './component1'

  export default {
    components: {
      component1
    }
  }
</script>

<style>
  .component2 { }
</style>
```

## Props & Events

```
<template>
  <div
    class="component1"
    @click="$emit('custom-event')">
    {{ content }}
  </div>
</template>

<script>
  export default {
    props: {
      content: {
        type: String,
        required: true
      }
    }
  }
</script>

<style>
  .component1 { }
</style>
```

```
<template>
  <div class="component2">
    <component1
      content="Hello from component 2"
      @custom-event="executeMethod()"/>
  </div>
</template>

<script>
  import component1 from './component1'

  export default {
    components: {
      component1
    },

    methods: {
      executeMethod() {}
    }
  }
</script>
```

## Vue-router

Official router for Vue.js

## Routes

Pixelmatters

```html
<!-- HTML -->
<div id="app">
  <h1>Hello App!</h1>
  <p>
    <!--
      `<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>

  <!--
    component matched by the route
    will render here
  -->
  <router-view></router-view>
</div>
```

```js
// Script
const routes = [
  { path: '/foo', component: Foo },
  { path: '/bar', component: Bar }
]

const router = new VueRouter({
  routes
})

new Vue({
  el: '#app',
  router
})
```

# $route and $router

Pixelmatters

```html
<!-- HTML -->
<div id="app">
  <h1>Hello App!</h1>
  <p>
    <!--
      `<router-link>` will be rendered as
      an `<a>` tag by default
    -->
    <router-link to="/foo/1">
      Go to Foo
    </router-link>
    <div @click="goToBar()">
      Go to bar
    </div>
   </p>

  <!--
    component matched by the route
    will render here
  -->
  <router-view></router-view>
</div>
```

```js
// Script
const routes = [
  { path: '/foo/:id', component: Foo },
  { path: '/bar/:id', component: Bar }
]

const router = new VueRouter({
  routes
})

new Vue({
  el: '#app',
  router,
  methods: {
    goToBar () {
      // get info from current route
      console.log(this.$route.params.id)

      this.$router.push('/bar', {
        params: { id: 1 }
      })
    }
  }
})
```

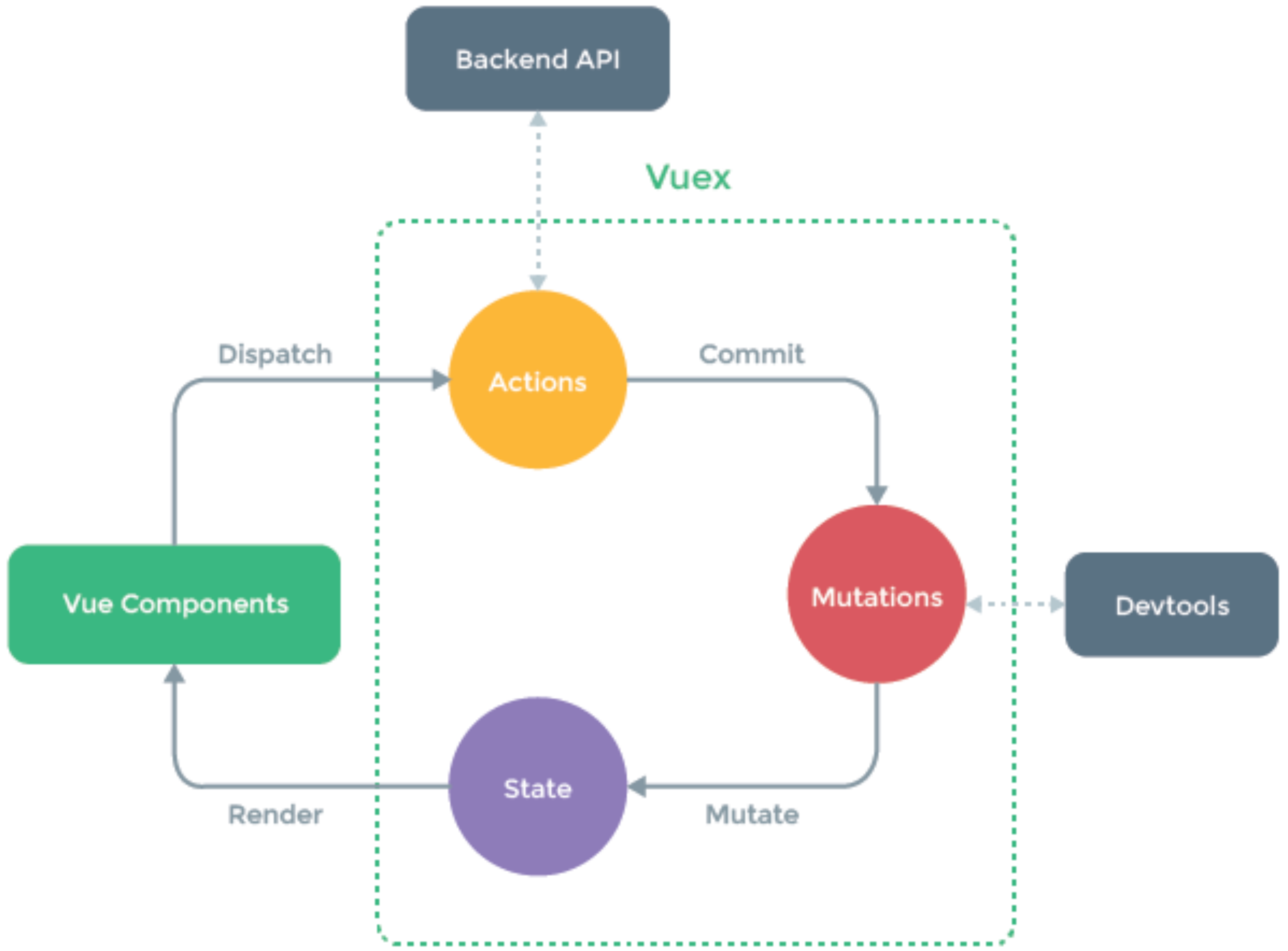
Pixelmatters

## Vuex

State management pattern + library for Vue.js

# Vuex Architecture

Pixelmatters



## Store

Pixelmatters

```
<template>
  <div class="component" @click="add()">
    {{ count }}
  </div>
</template>

<script>
  import { mapActions, mapState } from 'vuex'

  export default {
    computed: {
      ...mapState({
        count: state => state.count,
      }),
    },

    methods: {
      ...mapActions({
        // `this.add()` available in component
        add: 'increment'
      })
    }
  }
</script>
```

```
// Script
const store = new Vuex.Store({
  state: {
    count: 0
  },
  mutations: {
    increment (state) {
      state.count++
    }
  },
  actions: {
    increment (context) {
      context.commit('increment')
    }
  }
})


new Vue({
  el: '#app',
  store
})
```

Pixelmatters

## The time is now!

Build our Vue.js app with Vue CLI 3

## What we will create

- A Vue.js app with Vue CLI 3

- 2 components associated with 2 routes:
  **/posts** - list all faked posts
  **/posts/:id** - list information about a specific post

- Use Vuex - Add state, actions and mutations to handle fake blog posts from an external API. *We'll be using JSONPlaceholder which is a fake REST API*

## Requirements

- **Node.js** 8.11.0+ recommended (https://nodejs.org/)
  *Check which version do you have with “**node -v**” on the terminal*

- **Vue CLI 3** by running **“npm install -g @vue/cli@3.5.0”** on the terminal
  *Check which version do you have with “**vue -V**” on the terminal. If you have the previous* `vue-cli` *(1.x or 2.x) package installed globally, you need to uninstall it first with* **“`npm uninstall vue-cli -g`”**

- **Code Editor**

Pixelmatters

## Live coding 🤓

## We are hiring!

Get to know more about **Pixelmatters** at *www.pixelmatters.com*

# Thanks!

Pixelmatters

Twitter: @tiagofscoelho