



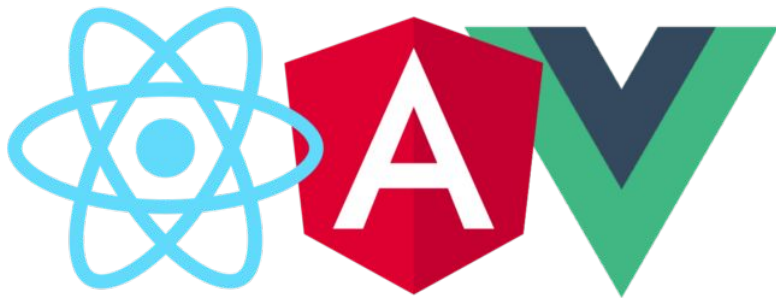
# Vue.js

Paul Bele - DevStaff Meetup 13.02

# Introduction to Vue.js

- Why do we need a Javascript framework?
- What's special about Vue?
- Basic architecture with code examples
- Vue ecosystem
- Simple SPA example

Why do we need  
a (modern) JS Framework?



# Example

Type an email address and hit enter

HTML

```
1 <html>
2 <body>
3   <div id="addressList">
4     <form>
5       <input>
6       <p class="help">Type an email address and hit enter</p>
7     </ul>
8   </div>
9 </body>
10 </html>
```

```
50 removeAddress(id) {
51   // state logic
52   this.state = this.state.filter(item => item.id !== id)
53
54   // UI logic
55   this.updateHelp()
56   const li = this.items[id]
57   this.ul.removeChild(li)
58 }
59
60 // utility method
61 updateHelp() {
62   if (this.state.length > 0) {
63     this.help.classList.add('hidden')
64   } else {
65     this.help.classList.remove('hidden')
66   }
67 }
68
69
70 const root = document.getElementById('addressList')
71 new AddressList(root)
```

JS

```
1 class AddressList {
2   constructor(root) {
3     // state variables
4     this.state = []
5
6     // UI variables
7     this.root = root
8     this.form = root.querySelector('form')
9     this.input = this.form.querySelector('input')
10    this.help = this.form.querySelector('.help')
11    this.ul = root.querySelector('ul')
12    this.items = {} // id -> li element
13
14    // event handlers
15    this.form.addEventListener('submit', e => {
16      e.preventDefault()
17      const address = this.input.value
18      this.input.value = ''
19      this.addAddress(address)
20    })
21
22    this.ul.addEventListener('click', e => {
23      const id = e.target.getAttribute('data-delete-id')
24      if (!id) return // user clicked in something else
25      this.removeAddress(id)
26    })
27  }
28
29   addAddress(address) {
30     // state logic
31     const id = String(Date.now())
32     this.state = this.state.concat({ address, id })
33
34     // UI logic
35     this.updateHelp()
36
37     const li = document.createElement('li')
38     const span = document.createElement('span')
39     const del = document.createElement('a')
40     span.innerText = address
41     del.innerText = 'delete'
42     del.setAttribute('data-delete-id', id)
43
44     this.ul.appendChild(li)
45     li.appendChild(del)
46     li.appendChild(span)
47     this.items[id] = li
48   }
49
50   removeAddress(id) {
51     // state logic
52     this.state = this.state.filter(item => item.id !== id)
53   }
```

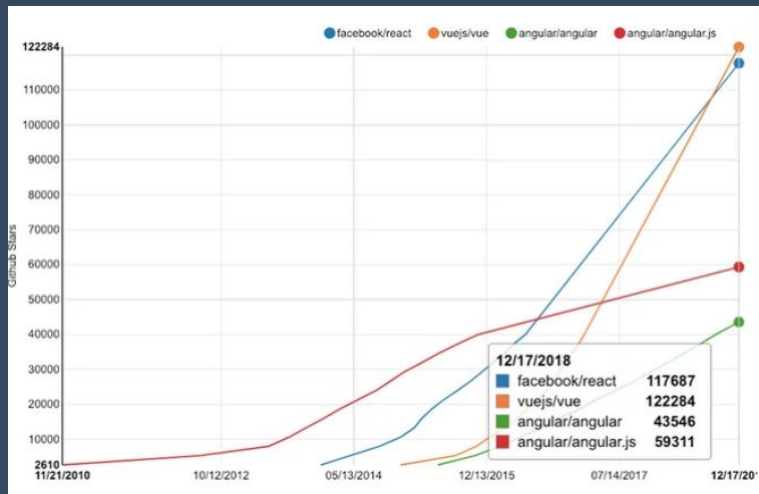
# The frameworks that are popular today have a few core commonalities:

- Synchronization of state and view
- Routing
- A template system
- Reusable components

# What is Vue?

# Introduction to Vue.js

- Github stars
  - Vue.js 151k
  - React 141k
  - Angular 59 k
- Google Trends
  - Passed React in August 2017
- Enterprise support
  - Alibaba, Baidu, Adobe, IBM, etc (<https://github.com/vuejs/awesome-vue#enterprise-usage>)



# Advantages

- Very Small Size
  - Angular: 500+ KB
  - React: 100 KB
  - Vue: 80 KB
- Performant (Virtual DOM)
- Uses same space with React and Angular
- Simple (One way data flow)
- Full-Featured API
- Powerful DevTools
- Active community
- Amazing documentation
- Easy to integrate into existing code base



# Getting started

- Super easy way is to use JSFiddle Hello World example

(<https://jsfiddle.net/chrisvfritz/50wL7mdz/>)

- `<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>`
- Vue-cli is not recommended for beginners, especially if you are not familiar with node.js tools

# Hello Vue.js

```
01_helloworld.html x
1 <!DOCTYPE html>
2 <html lang="en">
3 <body>
4 <script src="https://unpkg.com/vue"></script>
5
6 <div id="app">
7   <p>{{ message }}</p>
8 </div>
9 <script>
10   new Vue({
11     el: '#app',
12     data: {
13       message: 'Hello Vue.js!'
14     }
15   })
16 </script>
17 </body>
18 </html>
```

Hello Vue.js!

Elements Console Sources Network Performance Memory Application Security Au

Ready. Detected Vue 2.6.11. Components Vuex Events

Filter components Select <Root> Filter inspected data

<Root> = \$vm0

data

message: "Hello Vue.js!"

# The VUE instance

- Ground Zero for a Vuejs app
- Accepts an object containing the Vue instance definition
- Most definitions are shared between **instances** and **components**
- Has lifecycle hooks which are functions that are called at specific points in the app/component lifecycle

```
new Vue({  
  el: "#app",  
  data: {...},  
  computed: {...},  
  methods: {...},  
  template: {...}  
});
```

```
export default {  
  name: 'component',  
  data () { return {...} },  
  methods: {...}  
}
```

# Templates

- HTML based Templating syntax
- “Mustache” `{{}}` syntax for string interpolation and dynamic variables
- Rendered via a virtual DOM, for efficient updates

# Other Examples

Directives, Conditionals, Loops

# Directives

<!--

*A Vue.js directive can only appear in the form of  
a prefixed HTML attribute that takes the following format:*

<element

  prefix-directiveId="[argument:] expression [! filters...]">

</element>

-->

<div id="app-2">

  <div v-text="'hello ' + firstName + ' ' + lastName"></div>

  <span v-bind:title="message">

    Hover your mouse over me for a few seconds

    to see my dynamically bound title!

  </span>

</div>

<script>

```
new Vue({
  el: '#app-2',
  data: {
    firstName: 'Dev',
    lastName: 'Staff',
    message: 'You loaded this page on ' + new Date().toLocaleString()
  }
})
```

</script>

hello Dev Staff

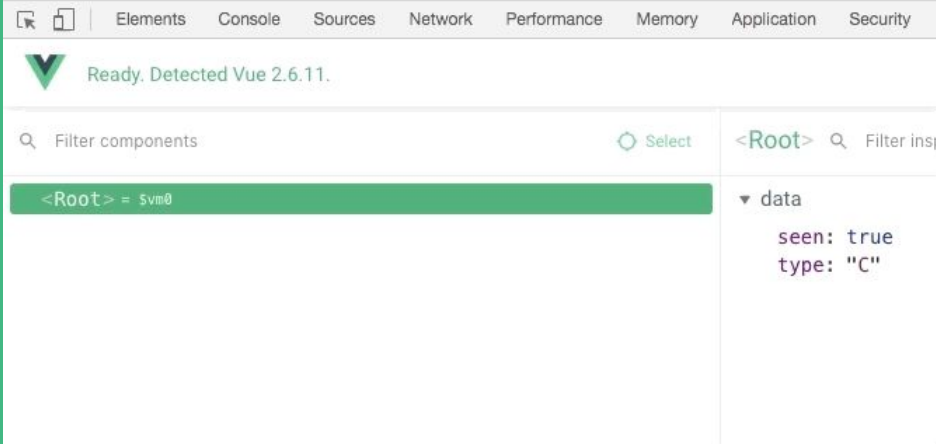
Hover your mouse over me for a few seconds to see my dynamically bound title!

# Conditional directives

```
<div id="app-3">
  <span v-if="seen">Now you see me</span>

  <div v-if="type === 'A'">
    A
  </div>
  <div v-else-if="type === 'B'">
    B
  </div>
  <div v-else-if="type === 'C'">
    C
  </div>
  <div v-else>
    Not A/B/C
  </div>
</div>
<script>
  new Vue({
    el: '#app-3',
    data: {
      seen: true,
      type: 'C'
    }
  })
</script>
```

Now you see me  
C



Elements Console Sources Network Performance Memory Application Security

Ready. Detected Vue 2.6.11.

Filter components Select <Root> Filter ins

<Root> = svm0

▼ data

- seen: true
- type: "C"

# Loops

```
<div id="app-4">
  <ol>
    <li v-for="todo in todos">
      {{ todo.text }}
    </li>
  </ol>
</div>
<script>
  new Vue({
    el: '#app-4',
    data: {
      todos: [
        { text: 'Learn JavaScript' },
        { text: 'Learn Vue' },
        { text: 'Build something awesome' }
      ]
    }
  })
</script>
```

1. Learn JavaScript
2. Learn Vue
3. Build something awesome



# Methods & Event Handling

- Methods
  - Arbitrary functions
  - Can access/manipulate data
  - Can be called from templates, other methods

# Methods & Event Handling

```
<div id="app-5">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">Reverse Message</button>
</div>
<script>
  new Vue({
    el: '#app-5',
    data: {
      message: 'Hello Vue.js!'
    },
    methods: {
      reverseMessage: function () {
        this.message = this.message.split('').reverse().join('')
      }
    }
  })
</script>
```

Hello Vue.js!

Reverse Message

# V-model directive

```
<div id="example">
  <input type="text" v-model="message" placeholder="edit me"/>
  <p>Message is: {{ message }}</p>
  <!--
    The same as:

    <input
      v-bind:value="something"
      v-on:input="something = $event.target.value"
    >
    -->
  </div>

  <input type="checkbox" id="checkbox" v-model="checked" />
  <label for="checkbox">{{ checked }}</label>

  <input type="radio" id="one" value="One" v-model="picked" />
  <label for="one">One</label>
  <input type="radio" id="two" value="Two" v-model="picked" />
  <label for="two">Two</label>
  <span>Picked: {{ picked }}</span>

  <select v-model="selected">
    <option disabled value="">Please select one</option>
    <option>A</option>
    <option>B</option>
    <option>C</option>
  </select>
  <span>Selected: {{ selected }}</span>
</div>
<script>
  new Vue({
    el: 'example',
    data: {
      message: null,
      checked: false,
      picked: null,
      selected: null
    }
  })
</script>
```

Text:

Message is:

Checkbox

☐ false

Radio

☐ One

☐ Two

Picked:

Select

Selected:

# Filters

```
<div id="app-5">
  <p>Hi {{ name | fallback }}!</p>
</div>
<script>
  new Vue({
    el: '#app-5',
    data: {
      name: 'DevStaff'
    },
    filters: {
      fallback: function(name) {
        return name ? name : 'there'
      }
    }
  })
</script>
```

Hi DevStaff!

# Computed properties

```
<div id="example">
  <p>Reversed message 1 using method: "{{ reverseMessage }}"</p>
  <p>Reversed message 1 using method: "{{ reverseMessage }}"</p>

  <p>Reversed message 1 using computed properties: "{{ reverseMessageComputed }}"</p>
  <p>Reversed message 2 using computed properties: "{{ reverseMessageComputed }}"</p>

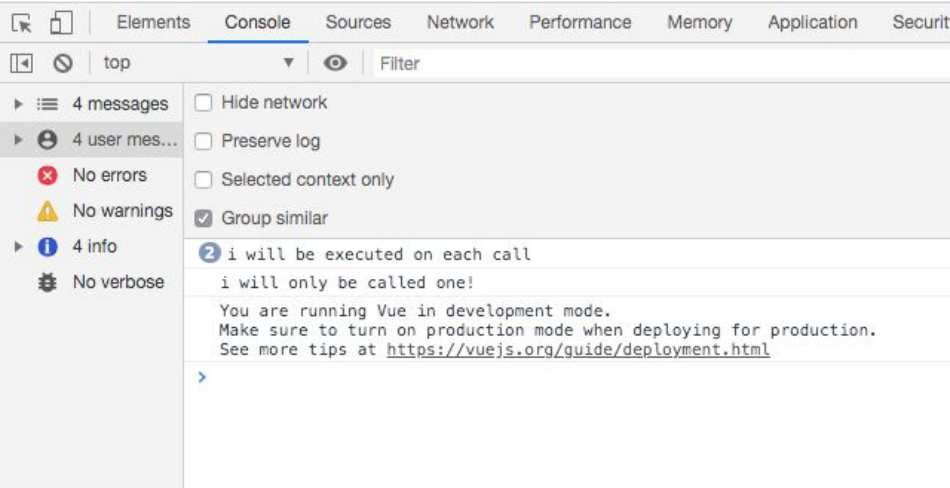
</div>
<script>
  new Vue({
    el: '#example',
    data: {
      name: 'Vue.js',
      message: 'This message will be reversed'
    },
    computed: {
      reverseMessageComputed: function() {
        console.log('i will only be called one!');
        return this.message.split('').reverse().join('')
      }
    },
    methods: {
      reverseMessage: function () {
        console.log('i will be executed on each call');
        return this.message.split('').reverse().join('')
      }
    }
  })
</script>
```

Reversed message 1 using method: "desrever eb lliw egassem sihT"

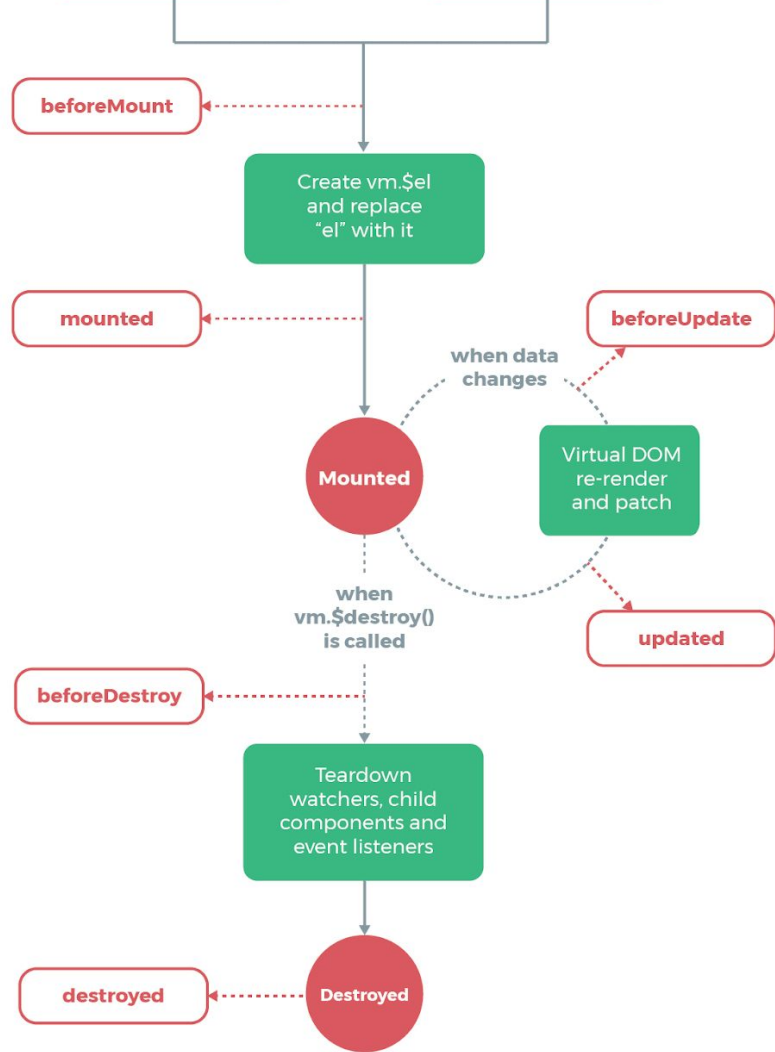
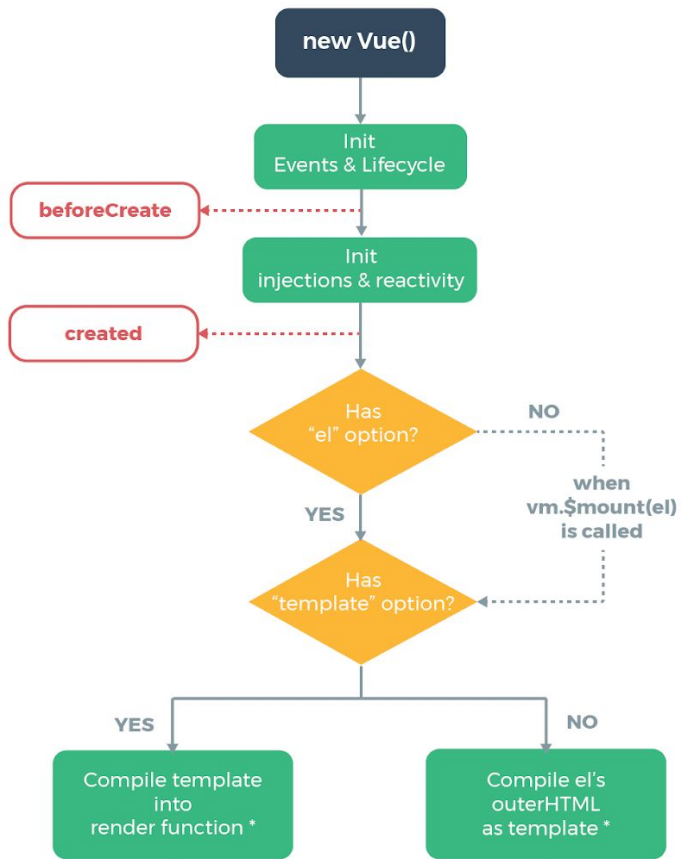
Reversed message 1 using method: "desrever eb lliw egassem sihT"

Reversed message 1 using computed properties: "desrever eb lliw egassem sihT"

Reversed message 2 using computed properties: "desrever eb lliw egassem sihT"



# Lifecycle



# Components

- Nested components hierarchy
- Each component renders either a template or returns `createElement()` calls
- Components are typically defined in single files with `<template>`, `<script>` and `<style>` sections

# Components

- Re-usable. Can be packaged in libraries
- Leverages pre-processor (webpack, rollup) to support “all the formats”
- HTML templates, are heavily borrowed from Angular (directives)
- CSS
  - Stylus, SCSS, PostCSS, etc
- JS
  - ES6, ES7, Typescript, etc



# Components

```
<template>
  <div class="hello">
    <h1>{{ msg }}</h1>
    <label for="user">Check a user:</label>
    <input type="text" name="about-user" v-model="user"/>
    <router-link :to="'/about-user/'+this.user">View the user</router-link>

    <div id="app">
      <router-link :to="{ name: 'HelloWorld' }">Home</router-link>
      <router-link to="/about">About</router-link>
    </div>
  </div>
</template>

<script>
export default {
  name: 'About',
  data () {
    return {
      msg: 'Here we talk about the about page',
      user: ''
    }
  }
}
</script>
|

<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped>
h1, h2 {
  font-weight: normal;
}
ul {
  list-style-type: none;
  padding: 0;
}
li {
  display: inline-block;
  margin: 0 10px;
}
a {
  color: #42b983;
}
</style>
```

# Vue.js - Batteries included

- Vuex - Official app management system
  - Atomic and centralized, similar to Redux
  - Integrates with Vue Dev Tools
- Vue-router - Official SPA router
- Ecosystem
  - Because Vue sits in the same “seams” as React, ideas are quickly pulled from react world into Vue world
    - 10k+ packages on npm with “vue” in the name
  - Editor support: Atom, Visual Studio Code, Sublime, IntelliJ, etc

# Vue-router

- First-class Router for Vue.js applications
- Similar conventions to React-Router < v4
- (or that's what i heard)
- Modes for hash(default) and browser history
- Redirects, URL params, wildcard, full package!

```
import Vue from 'vue'
import Router from 'vue-router'
import HelloWorld from '@components/HelloWorld'

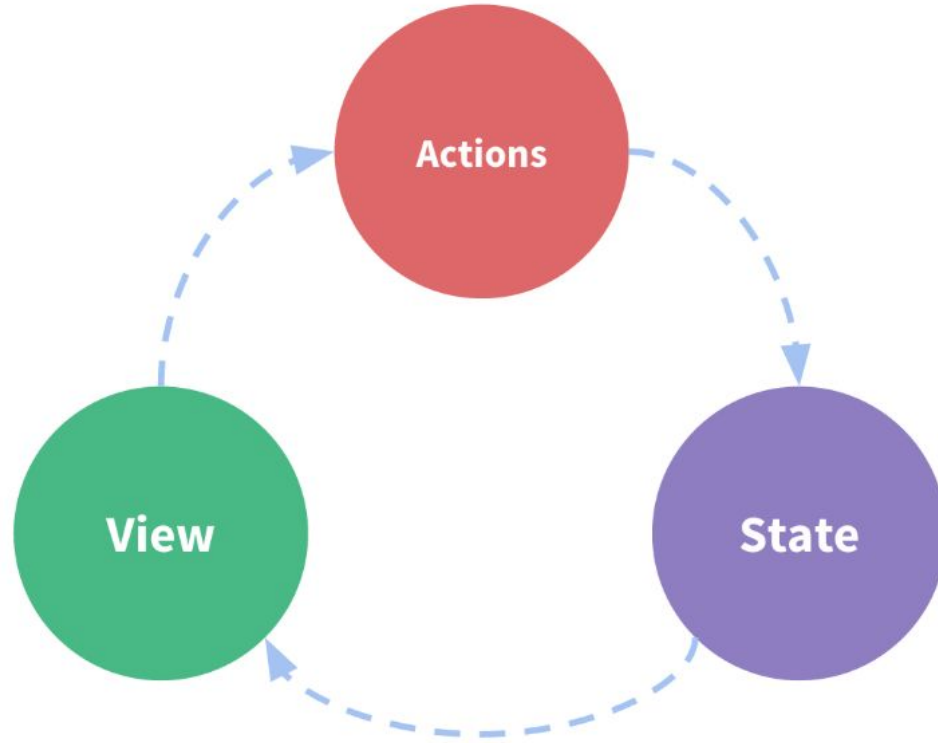
Vue.use(Router)

export default new Router({
  routes: [
    {
      path: '/',
      name: 'HelloWorld',
      component: HelloWorld
    }
  ]
})
```

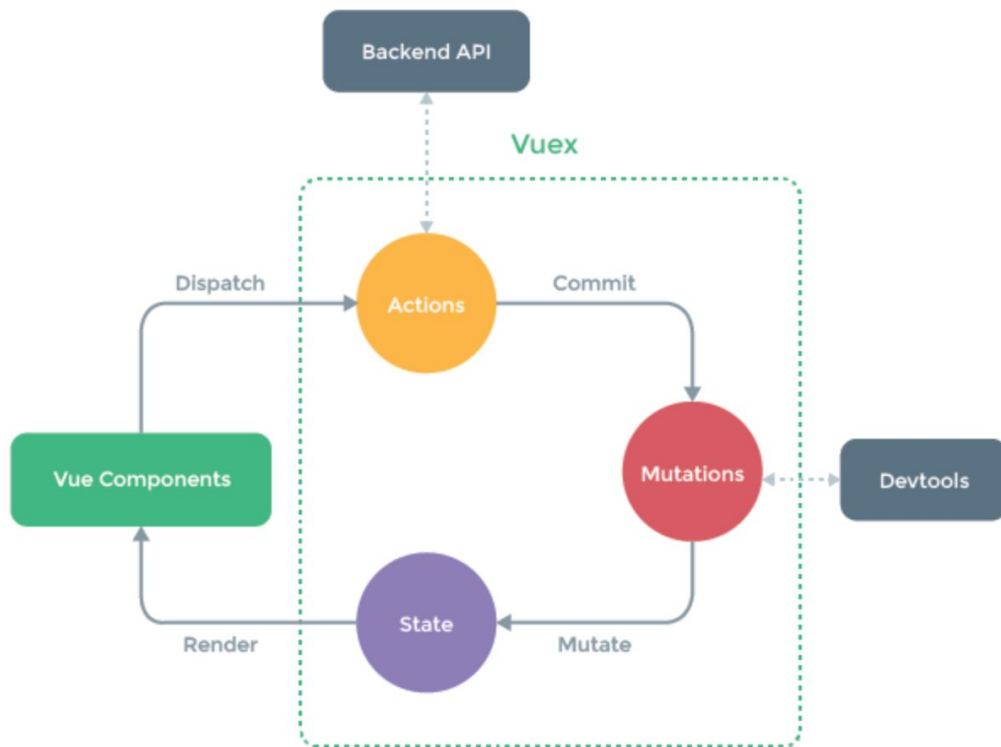
# Vuex

- State management library, a la Redux
- Provides conventions/constraints around application state
- Single store
- Data access via “getters”
- Data modification via “mutations”
- Asynchronous work done in “actions”

# Vuex - State management pattern



# Vuex



# Example using vue-cli

- A simple as possible SPA app using vue-cli project as a starting point
- Have 3 pages and be able to pass GET parameters from one to the other
- Leverage webpack dev server (setup by vue-cli)

# Project Init

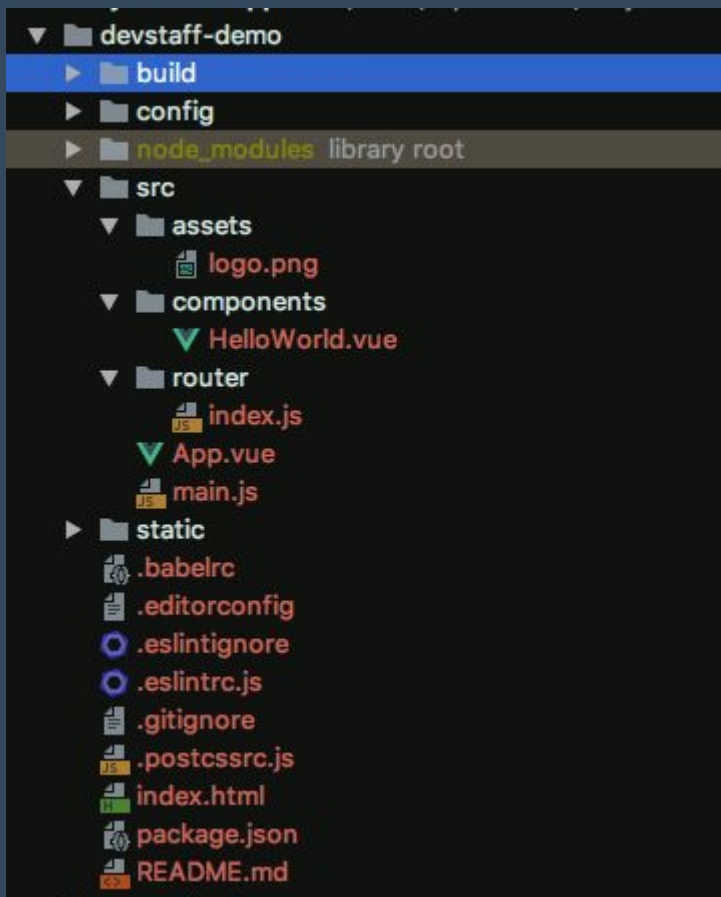


A terminal window with a dark background. The title bar at the top reads "...talk-snippets (zsh)". The prompt is "vuejs-talk-snippets:master (●) >>>". The command "vue init web" has been entered, and the cursor is at the end of the line.

```
...talk-snippets (zsh)  
vuejs-talk-snippets:master (●) >>> vue init web
```



# Dir structure



devstaff-demo:master (●) >>>

View code

# Summary

- Vue.js is not “just” another Javascript framework
- Aims for a balance between rich, developer-friendly features and clean, understandable code
- Community is vibrant and engaged, and shows every sign of being in this for the long haul
- Vue.js is definitely worth learning, either as a first-timer’s Javascript framework or developers looking for a fresh SPAs

Thank you!  
Any questions?

<https://github.com/skmetaly/vuejs-talk-snippets>

<https://www.slideshare.net/PaulBele/introduction-to-vuejs-devstaff-meetup-1302>