VUE 3

SMALLER, FASTER & STRONGER

Dan Vega

CodeMash 2020

Hello,

I'm Dan.

A Software Developer with 20 years experience.

"Live as if you were to die tomorrow.
Learn as if you were to live forever."
- Mahatma Gandhi

https://www.danvega.dev


Cleveland, Ohio

HUSBAND & FATHER

ABOUT TECH ELEVATOR

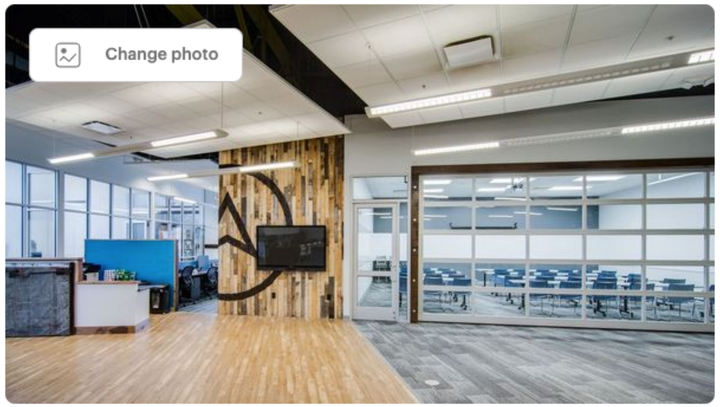
6 LOCATIONS
64 EMPLOYEES
1,000+ ALUMNI

1. Cleveland
2. THE CITY OF COLUMBUS
3. city of CINCINNATI
4. CITY OF PITTSBURGH
5. CITY OF Detroit
6. City of Philadelphia

meetup

Start a new group
30% OFF

Explore Messages Notifications



VueCLE

Cleveland, OH
208 members · Public group
Organized by **Tech Elevator** and **2 others**

Share:

About Events Members Photos Discussions More

Manage group Create event

What we're about

VueJS is the most popular JavaScript Framework in the world (counting by the number of stars on Github). VueCLE is the only meetup group in the Clevela...

Read more

Past events (8)

See all

TUE, DEC 17, 6:30 PM

NEO Web Holiday Hoopla

Masthead Brewing Co.

4 attendees Manage Went

THU, OCT 3, 5:30 PM

Server Side Rendering with Nuxt.js

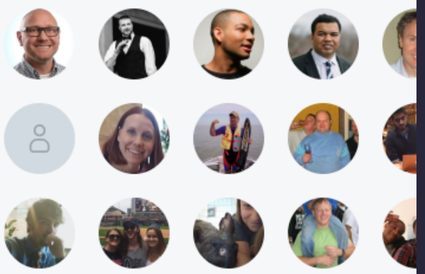
Tech Elevator

25 attendees Manage Went

Organizers

Tech Elevator and **2 others**
Message

Members (208)



VUE CLE MEETUP

package.json	release: v3.0.0-alpha.1	3 hours ago
rollup.config.js	build: warn invalid format in build script	11 days ago
tsconfig.json	chore: Use dynamic paths in tsconfig.json (#548)	15 days ago
yarn.lock	build: changelog generation	3 hours ago

README.md

vue-next PASSED

Status: Alpha.

The current codebase has basic feature parity with v2.x, together with the changes proposed in merged RFCs. There is a simple webpack-based setup with Single-File Component support available here.

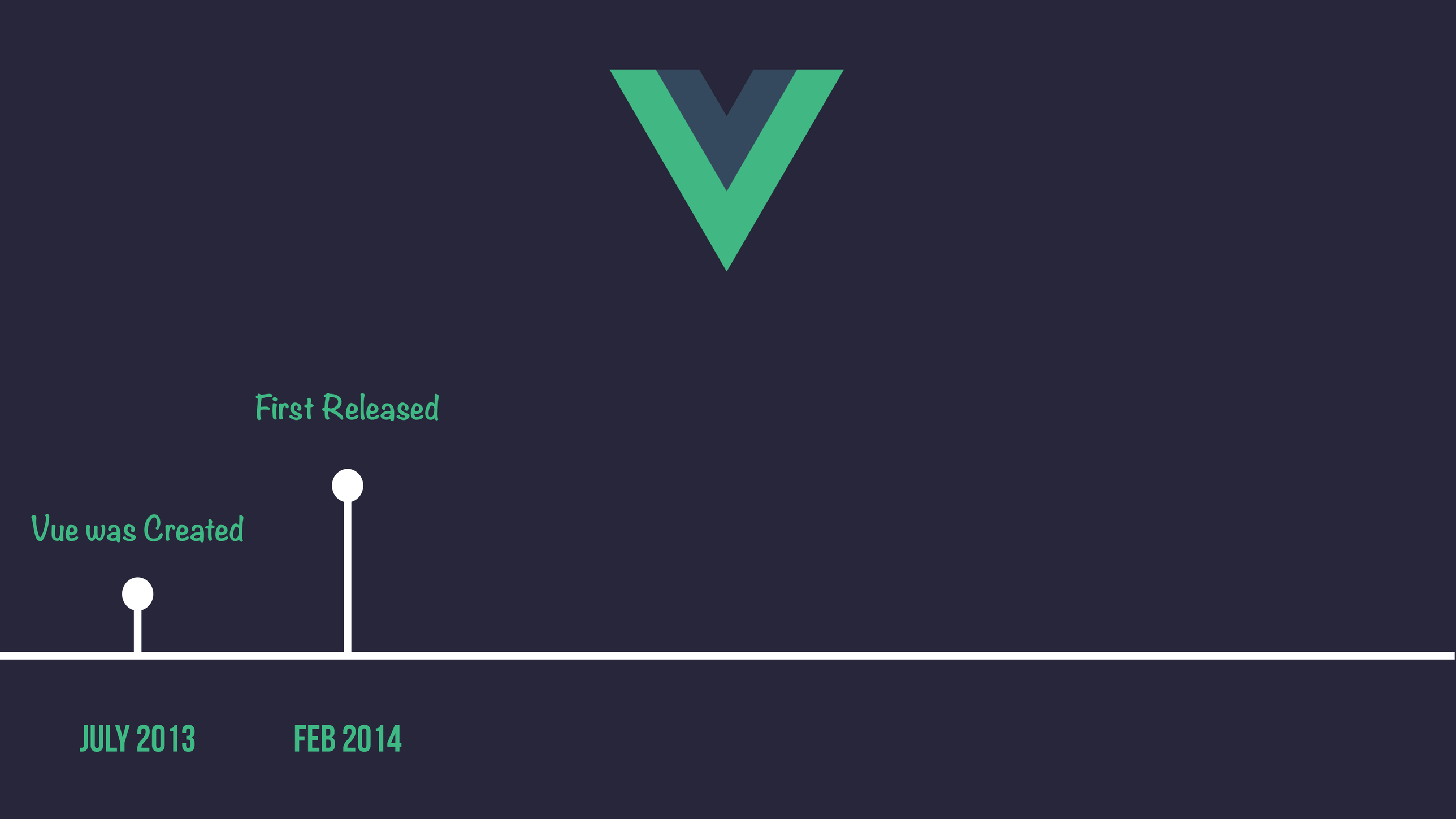
Contribution

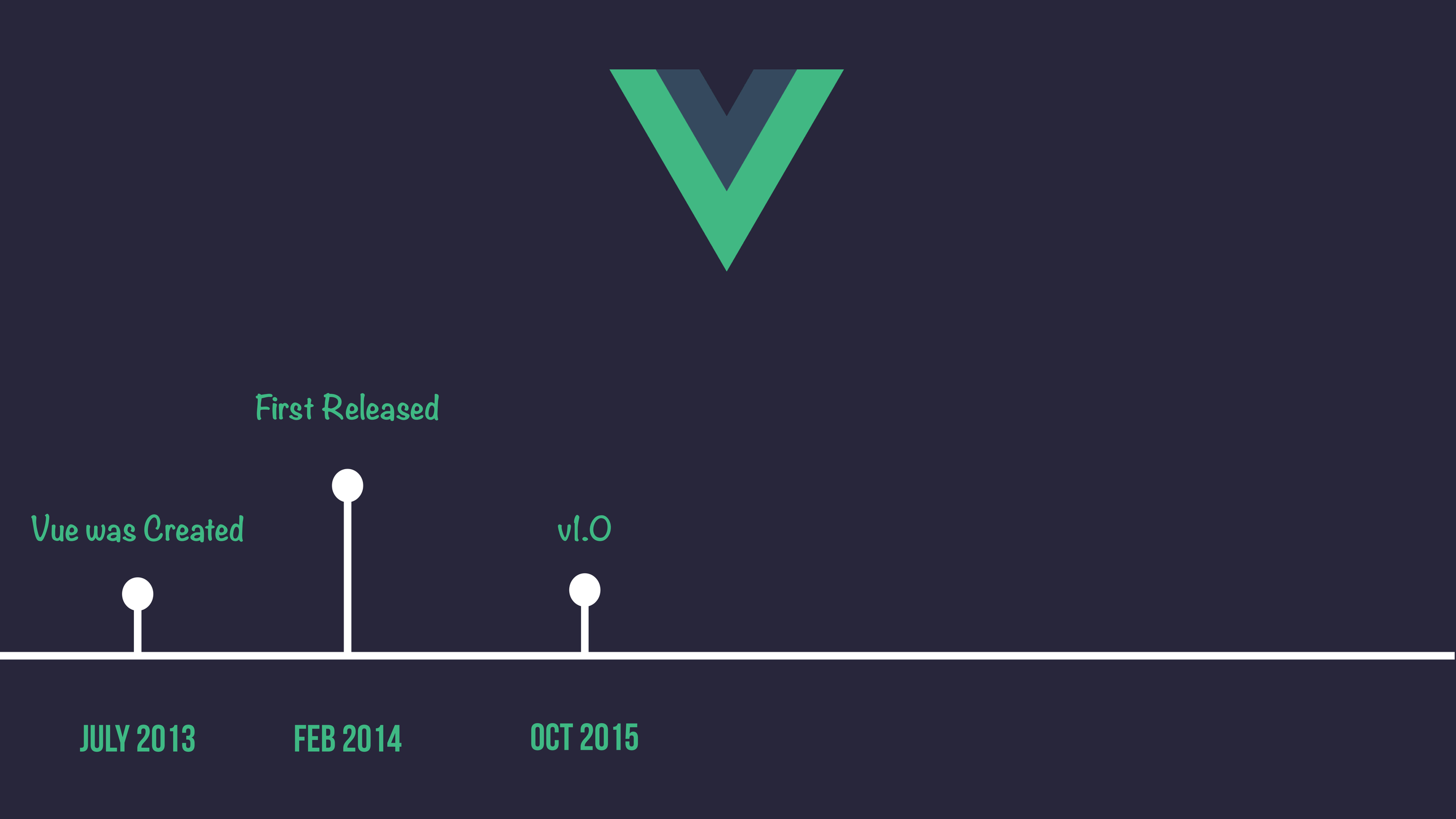
See Contributing Guide.

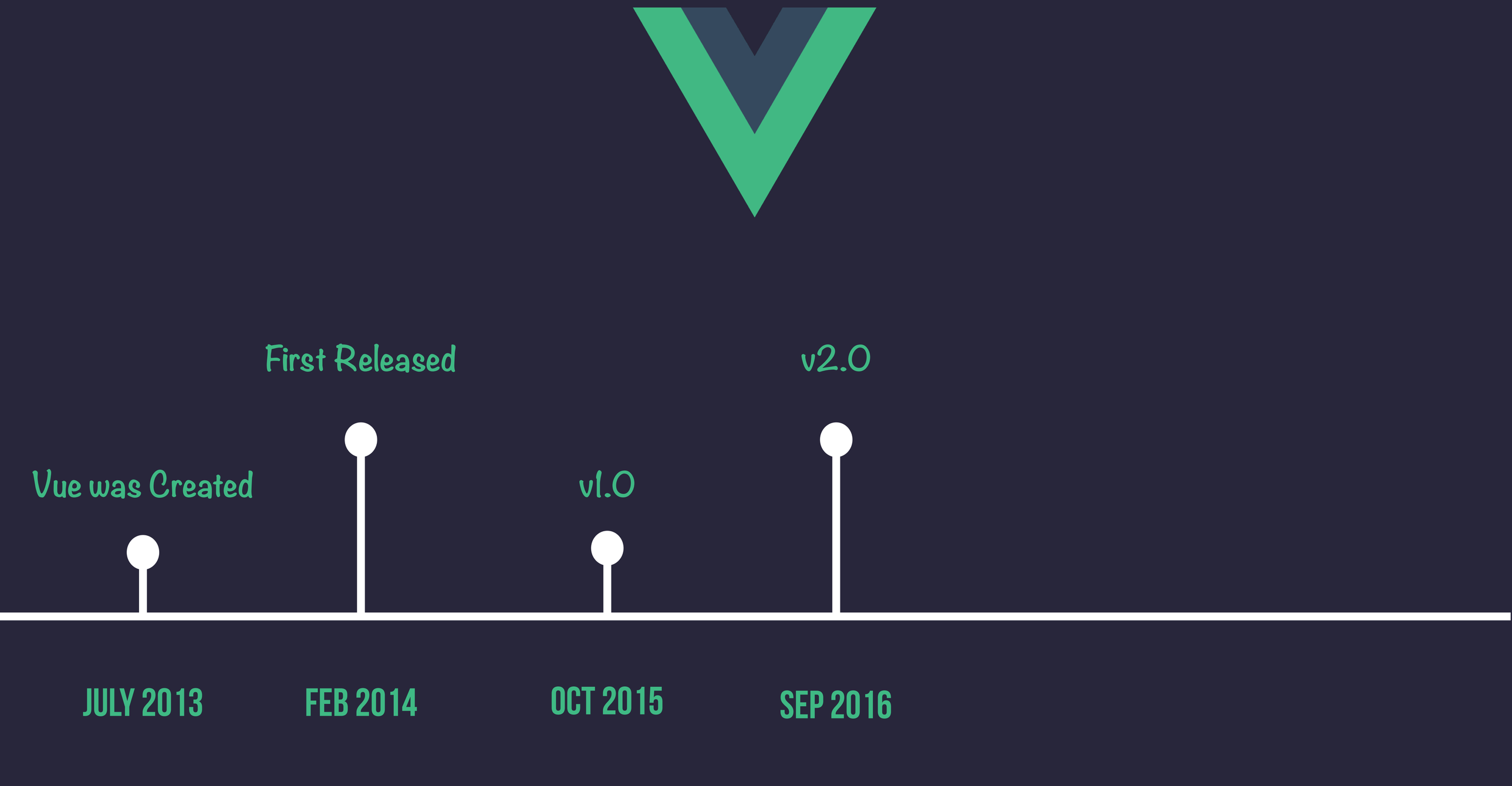
VUE 3
HOW DID WE GET HERE?

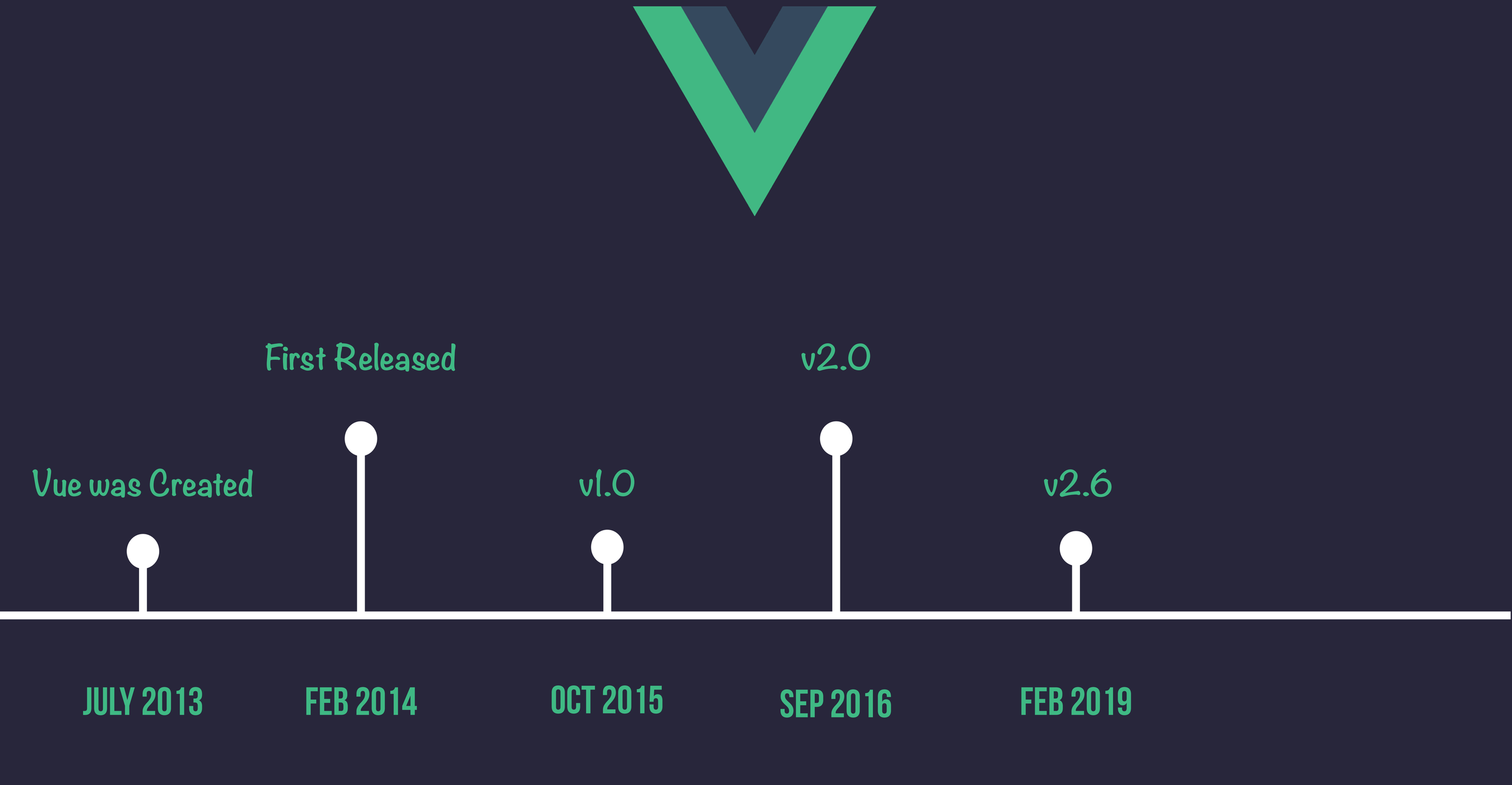
Vue was Created

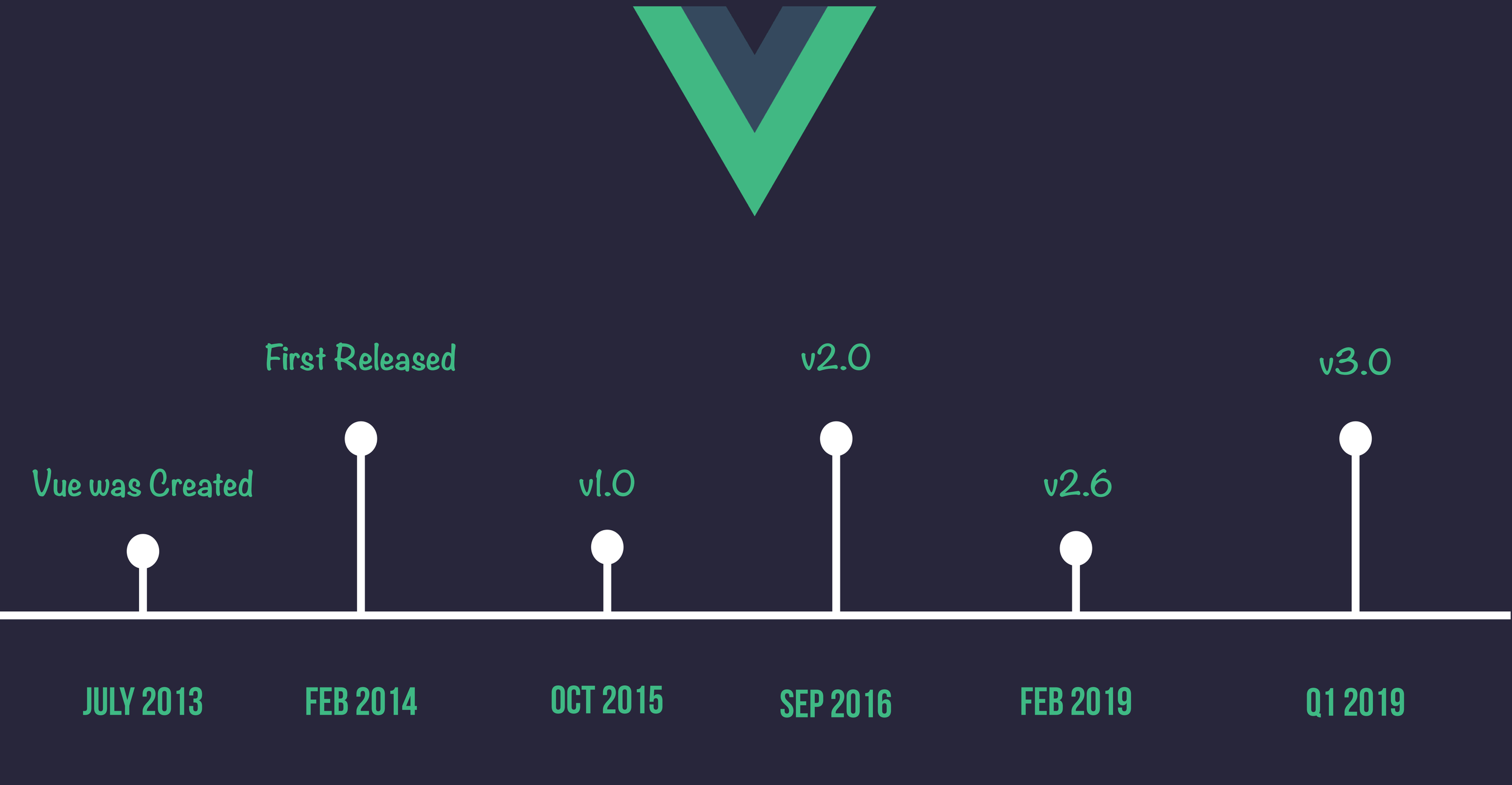
JULY 2013


First Released
Vue was Created
JULY 2013
FEB 2014


First Released
Vue was Created
v1.0
JULY 2013
FEB 2014
OCT 2015


First Released
v2.0
Vue was Created
v1.0
JULY 2013
FEB 2014
OCT 2015
SEP 2016


Vue was Created
First Released
v1.0
v2.0
v2.6
JULY 2013
FEB 2014
OCT 2015
SEP 2016
FEB 2019


Vue was Created
JULY 2013
First Released
FEB 2014
v1.0
OCT 2015
v2.0
SEP 2016
v2.6
FEB 2019
v3.0
Q1 2019


VUE 3 RELEASE DATE?

HOW ARE NEW FEATURES ADDED TO VUE?

VUE HAS AN EXTREMELY DIVERSE AUDIENCE

- Beginners just progressing from HTML, CSS & JavaScript
- Professionals moving on from jQuery
- Veterans migrating from another framework
- Backend engineers looking for lightweight frontend solution
- Architects choosing the foundation for entire organization

RFCs
Request for Comments

https://github.com/vuejs/rfcs/tree/master/active-rfcs

vuejs / rfcs

Watch 298 | Star 1.8k | Fork 159

Code | Issues 16 | Pull requests 11 | Actions | Security | Insights

Branch: master ▾ rfcs / active-rfcs /

Create new file | Upload files | Find file | History


yyx990803 Remove inline-template (#98) ···

Latest commit 9f17365 13 days ago

..

0001-new-slot-syntax.md	Typo fix (#12)	11 months ago
0002-slot-syntax-shorthand.md	fix: use YYYY-MM-DD date format (#41)	7 months ago
0003-dynamic-directive-argument...	Fix variable in template string to be consistent with js expression e...	11 months ago
0004-global-api-treeshaking.md	Rename 0000-global-api-treeshaking.md to 0004-global-api-treeshaking.md	7 months ago
0005-replace-v-bind-sync-with-v-...	Replace v-bind's .sync with a v-model argument (#8)	last month
0006-slots-unification.md	update(slots-unification): fix example	last month
0007-functional-async-api-change....	Functional and async components API change (#27)	last month
0008-render-function-api-change....	Render function API change (#28)	last month
0009-global-api-change.md	Global mounting/configuration API change (#29)	last month
0010-optional-props-declaration.md	Optional props declaration (#25)	last month
0011-v-model-api-change.md	Component v-model API change (#31)	last month
0012-custom-directive-api-change....	Custom Directive API Change (#32)	last month
0013-composition-api.md	update the referenced issue (#103)	last month
0014-drop-keycode-support.md	Remove keyCode support in v-on (#95)	13 days ago
0015-remove-filters.md	Remove filters (#97)	13 days ago
0016-remove-inline-templates.md	Remove inline-template (#98)	13 days ago

mesqueeb update the referenced issue (#103) fb0463e on Nov 17

2 contributors

875 lines (614 sloc) | 40.4 KB | Raw | Blame | History

- Start Date: 2019-07-10
- Target Major Version: 2.x / 3.x
- Reference Issues: #78
- Implementation PR: N/A

> Since this RFC is long, it is deployed in a more readable format here. There is also an accompanying API Reference.

Summary

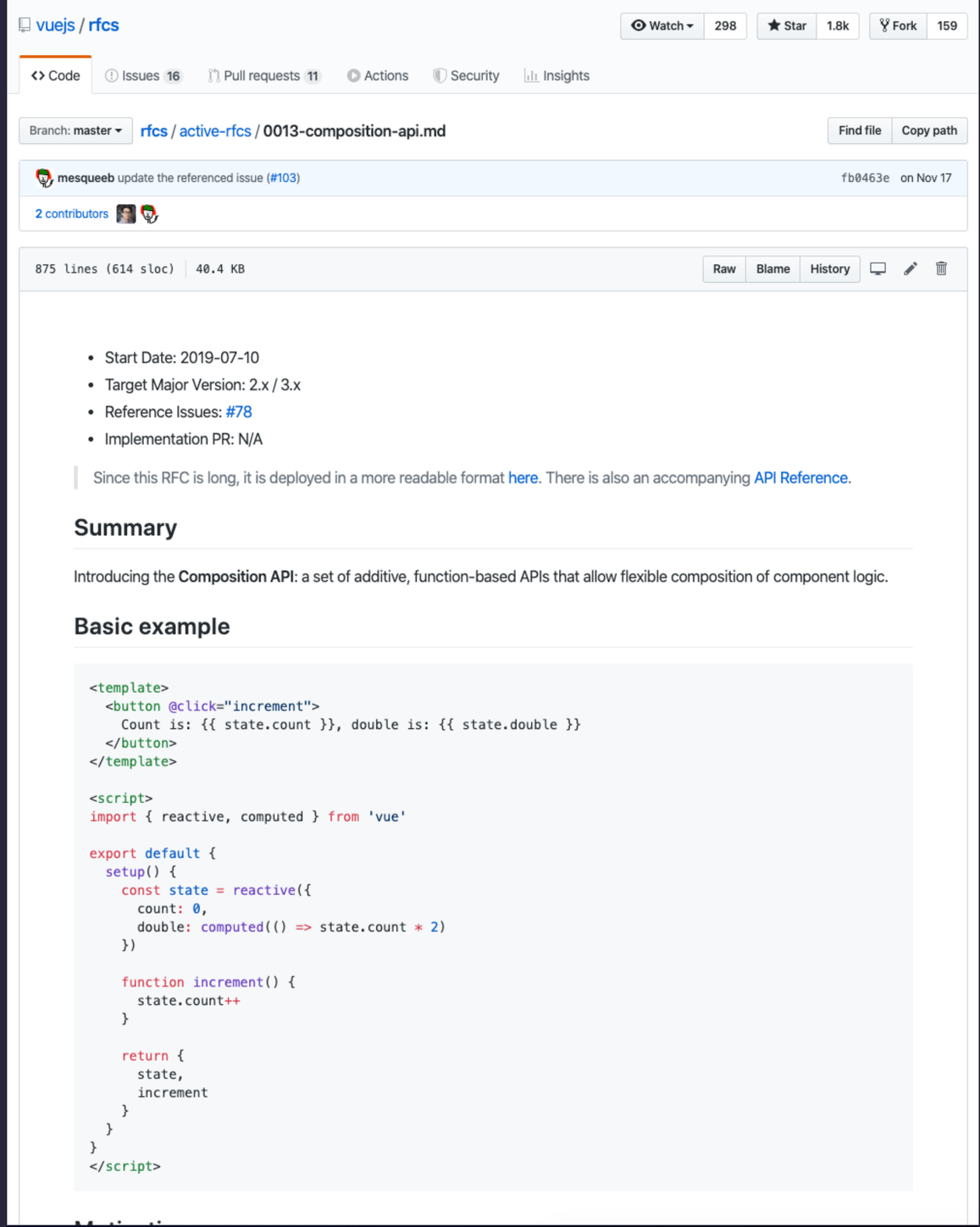
Introducing the **Composition API**: a set of additive, function-based APIs that allow flexible composition of component logic.

Basic example

```
<template>
  <button @click="increment">
    Count is: {{ state.count }}, double is: {{ state.double }}
  </button>
</template>

<script>
import { reactive, computed } from 'vue'

export default {
  setup() {
    const state = reactive({
      count: 0,
      double: computed(() => state.count * 2)
    })

    function increment() {
      state.count++
    }

    return {
      state,
      increment
    }
  }
}
</script>
```

RFC Template

- Summary
- Basic Example
- Motivation
- Detailed Design
- Drawbacks
- Alternatives
- Adoption Strategy



vuejs / rfcs

Watch 298 | Star 1.8k | Fork 159

Code | Issues 16 | Pull requests 11 | Actions | Security | Insights

Branch: master ▾ rfcs / active-rfcs / 0013-composition-api.md — Find file | Copy path

mesqueeb update the referenced issue (#103) — fb0463e on Nov 17

2 contributors

875 lines (614 sloc) | 40.4 KB — Raw | Blame | History

- Start Date: 2019-07-10
- Target Major Version: 2.x / 3.x
- Reference Issues: #78
- Implementation PR: N/A

> Since this RFC is long, it is deployed in a more readable format here. There is also an accompanying API Reference.

Summary

Introducing the **Composition API**: a set of additive, function-based APIs that allow flexible composition of component logic.

Basic example

```
<template>
  <button @click="increment">
    Count is: {{ state.count }}, double is: {{ state.double }}
  </button>
</template>

<script>
import { reactive, computed } from 'vue'

export default {
  setup() {
    const state = reactive({
      count: 0,
      double: computed(() => state.count * 2)
    })

    function increment() {
      state.count++
    }

    return {
      state,
      increment
    }
  }
}
</script>
```

VUE 3

SMALLER, FASTER & STRONGER

SMALLER

FASTER

STRONGER

- ➡Global API Change
- ➡Treeshaking
- ➡TypeScript
- ➡Fragments

SMALLER

- ➡Global API Change
- ➡Treeshaking
- ➡TypeScript
- ➡Fragments

FASTER

- ➡Virtual DOM
- ➡Reactivity System

STRONGER

SMALLER

- ➡Global API Change
- ➡Treeshaking
- ➡TypeScript
- ➡Fragments

FASTER

- ➡Virtual DOM
- ➡Reactivity System

STRONGER

- ➡Composition API

VUE 3: SMALLER

GLOBAL API CHANGE

RE-DESIGN APP BOOTSTRAPPING & GLOBAL API

- Global APIs that globally mutate Vue's behavior are now moved to app instances created using the new `createApp()` method, and their effects are now scoped to that app instance only.
- Global APIs that are do not mutate Vue's behavior (e.g. `nextTick`) are now named exports.

VUE 2 BEFORE

```
import Vue from 'vue'
import App from './App.vue'

Vue.config.ignoredElements = [/^app-/]
Vue.use(/* ... */)
Vue.mixin(/* ... */)
Vue.component(/* ... */)
Vue.directive(/* ... */)

new Vue({
  render: h => h(App)
}).$mount('#app')
```

PROBLEM #1

- Global configuration makes it easy to accidentally pollute other test cases during testing.
- vue-test-utils has to implement a special API createLocalVue to deal with this

PROBLEM #2

```
// this affects both root instances
Vue.mixin({ /* ... */ })

const app1 = new Vue({ el: '#app-1' })
const app2 = new Vue({ el: '#app-2' })
```

NEW GLOBAL API: CREATE APP

```
import { createApp } from 'vue'

const app = createApp()
```

MOUNTING

```
import { createApp } from 'vue'

const app = createApp()
const rootInstance = app.mount(App, '#app')
```

VUE 3 AFTER

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

const app = createApp()

app.config.ignoredElements = [/^app-/]
app.use(/* ... */)
app.mixin(/* ... */)
app.component(/* ... */)
app.directive(/* ... */)

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

GLOBAL API TREESHAKING

THE DILEMMA OF VUE 2

VUE 3: NAMED IMPORTS ONLY

v2

```
import Vue from 'vue'

Vue.nextTick(() => {})

const obj = Vue.observable({})
```

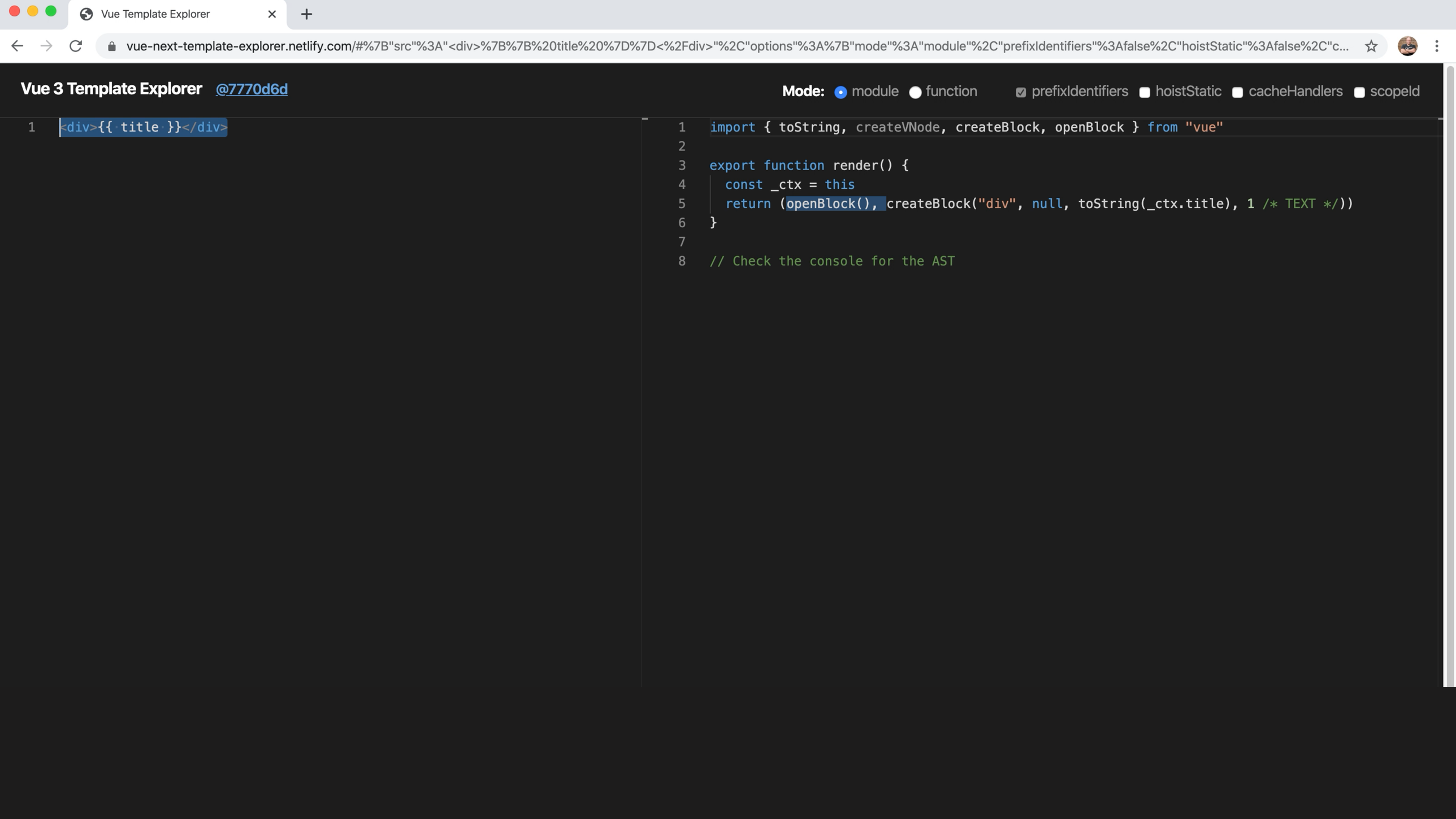
v3

```
import Vue, { nextTick, observable } from 'vue'

Vue.nextTick // undefined

nextTick(() => {})

const obj = observable({})
```

IN VUE 3:

- CDN will include all features
- Most Global and internal APIs are provided as ES Modules named exports (tree-shakable)
- The Compiler generates tree-shakable code for your templates as well

"The new baseline is about 10kb gzipped, which is less than half of the current runtime"


Vue Template Explorer
vue-next-template-explorer.netlify.com/#%7B"src"%3A" %7B%7B%20title%20%7D%7D<%2Fdiv>"%2C"options"%3A%7B"mode"%3A"module"%2C"prefixIdentifiers"%3Afalse%2C"hoistStatic"%3Afalse%2C"c...
Vue 3 Template Explorer @7770d6d
Mode: module function prefixIdentifiers hoistStatic cacheHandlers scopeId
1 {{ title }}
1 import { toString, createVNode, createBlock, openBlock } from "vue"
2
3 export function render() {
4 const _ctx = this
5 return (openBlock(), createBlock("div", null, toString(_ctx.title), 1 /* TEXT */))
6 }
7
8 // Check the console for the AST


TS

Did you know that Vue 3 was completely rewritten from the ground up using TypeScript?

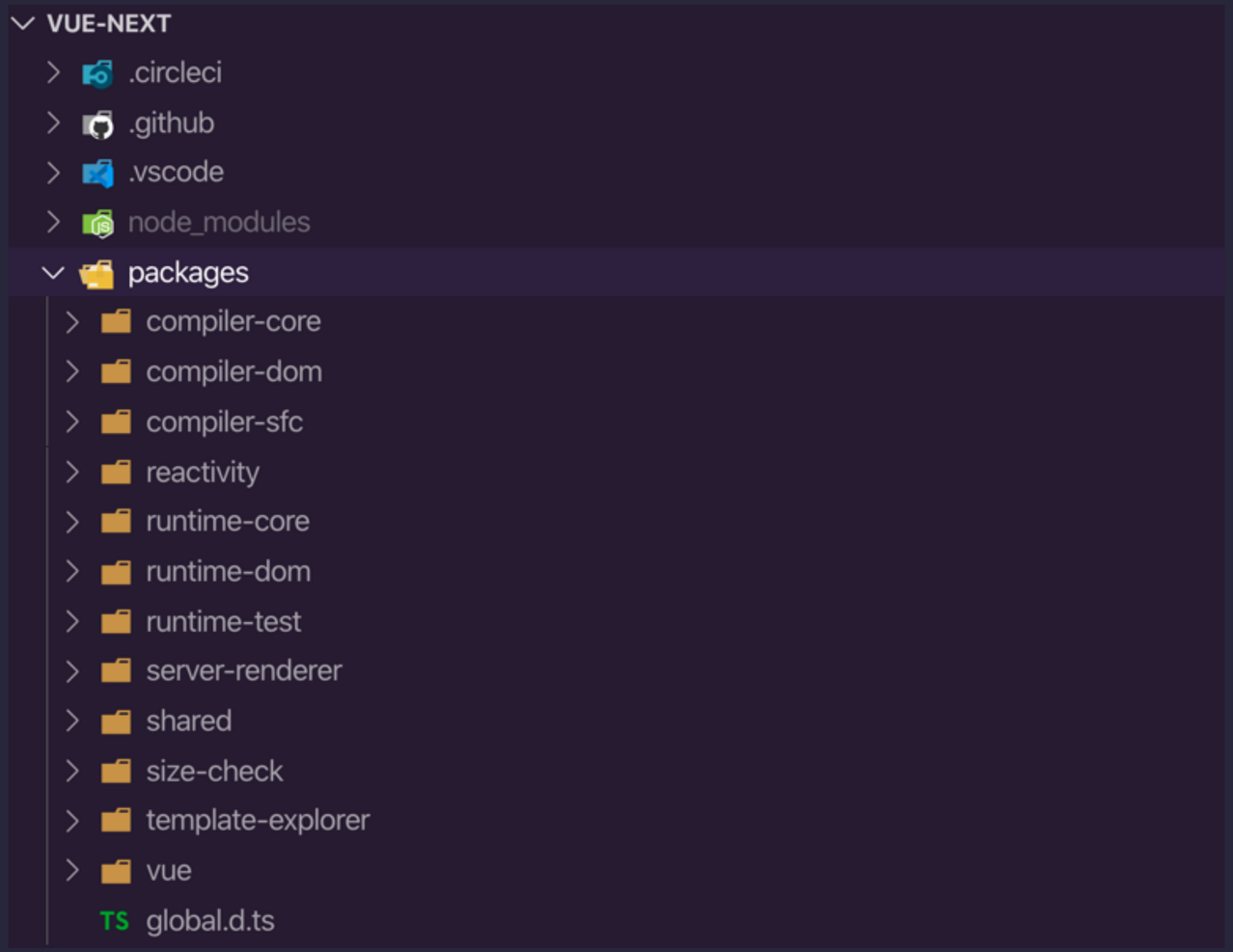
Are you telling me I need to learn TypeScript to use Vue 3 Dan?

No! You can continue to write Vue apps using JavaScript

SO WHY REWRITE VUE IN TYPESCRIPT?

- A More Maintainable Codebase
- Provide better type definitions for those using TypeScript
  - TypeScript definitions benefit JavaScript developers just as well
- Component code using TypeScript and JavaScript will look largely the same

DECOUPLED PACKAGES



FRAGMENTS

SINGLE ROOT NODE IN VUE 2.X

```
<template>
  <div class="title" v-html="title"></div>
</template>

<script>
export default {
  data() {
    return {
      title: "<h1>This is a title</h1>",
    }
  }
}
</script>
```

SINGLE ROOT NODE IN VUE 2.X

```
<template>
  <div class="title" v-html="title"></div>
  <div class="description" v-html="description"></div>
</template>

<script>
export default {
  data() {
    return {
      title: "<h1>This is a title</h1>",
      description: "<p>This is a description</p>",
    }
  }
}
</script>
```

SINGLE ROOT NODE IN VUE 2.X

```
<template>
  <div class="title" v-html="title"></div>
  <div class="description" v-html="description"></div>
</template>
```

```
Component template should contain exactly one root element. If
you are using v-if on multiple elements, use v-else-if to
chain them instead.

1  |
2  |    <div class="title" v-html="title"></div>
3  |    <div class="description" v-html="description"></div>
   |    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
4  |

(found in <Root>)
```

SINGLE ROOT NODE IN VUE 2.X

```
<template>
  <div class="page">
    <div class="title" v-html="title"></div>
    <div class="description" v-html="description"></div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      title: "<h1>This is a title</h1>",
      description: "<p>This is a description</p>",
    }
  }
}
</script>
```

MULTIPLE ROOT NODES IN VUE 3

```
<template>
  <div class="title" v-html="title"></div>
  <div class="description" v-html="description"></div>
</template>

<script>
export default {
  data() {
    return {
      title: "<h1>This is a title</h1>",
      description: "<p>This is a description</p>",
    }
  }
}
</script>
```

MULTIPLE ROOT NODES IN VUE 3

```
<div class="title" v-html="title"></div>
<div class="description" v-html="description"></div>

<script>
import { createVNode, createBlock, Fragment, openBlock } from "vue"

export function render() {
  const _ctx = this
  return (openBlock(), createBlock(Fragment, null, [
    createVNode("div", {
      class: "title",
      innerHTML: _ctx.title
    }, null, 8 /* PROPS */, ["innerHTML"]),
    createVNode("div", {
      class: "description",
      innerHTML: _ctx.description
    }, null, 8 /* PROPS */, ["innerHTML"])
  ], 64 /* STABLE_FRAGMENT */))
}
</script>
```

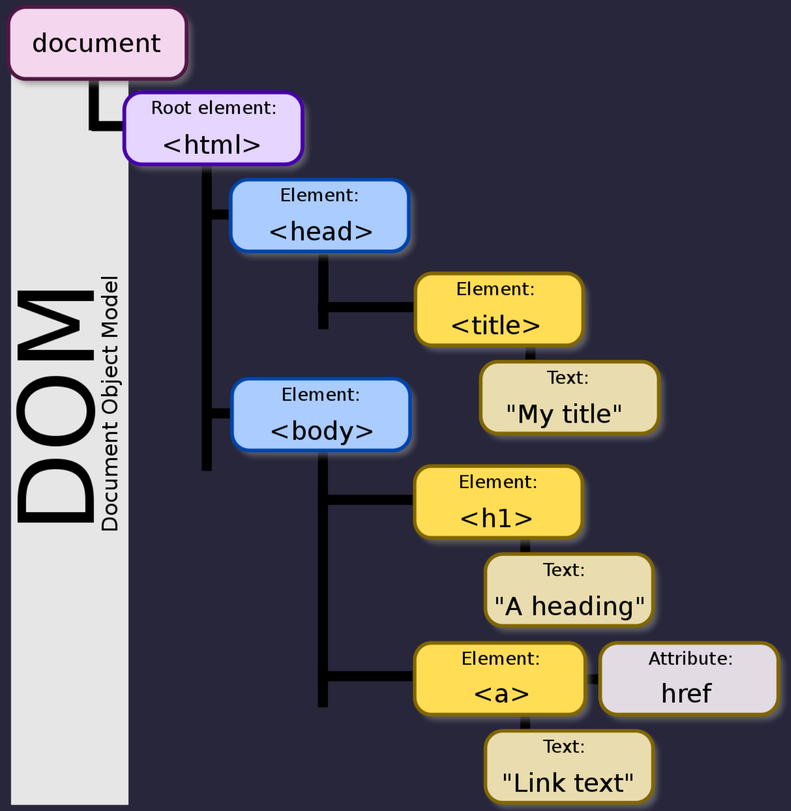
VUE 3: FASTER

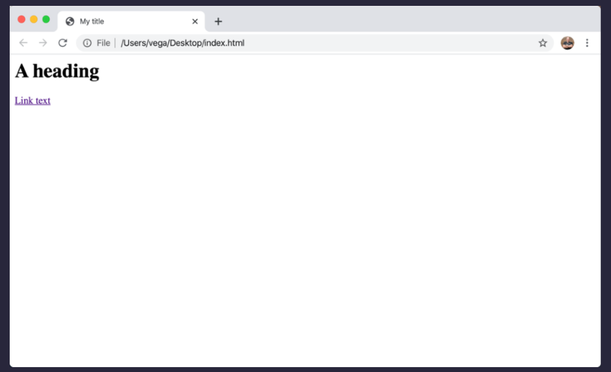
VIRTUAL DOM

DOM (Document Object Model) is an abstraction of a structured text, where HTML Tags become nodes in the DOM. While HTML is a text, the DOM is an in-memory representation of this text in form of nodes. The DOM provides an interface (API) to traverse and modify the nodes, and It contains methods like getElementById or removeChild.

DOCUMENT OBJECT MODEL

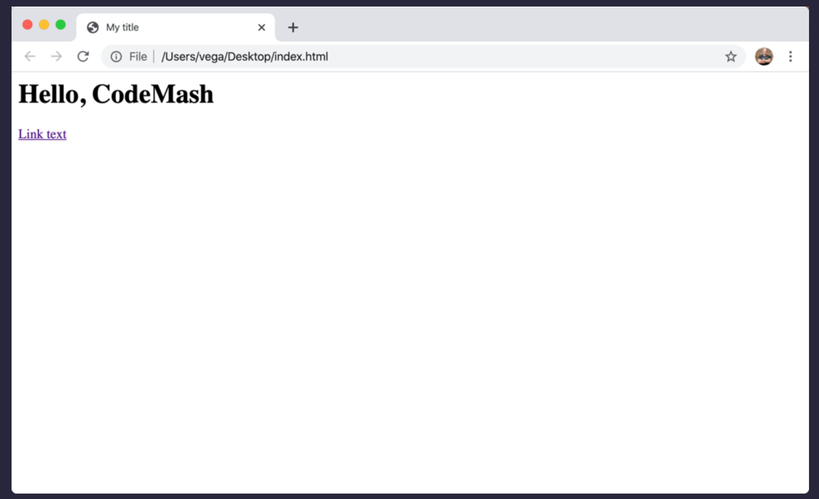
```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="">Link text</a>
  </body>
</html>
```





DOM API

```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="">Link text</a>
  </body>
</html>

<script>
  const heading = document.getElementsByTagName('h1')[0]
  heading.textContent = "Hello, CodeMash"
</script>
```



WHAT IS THE VIRTUAL DOM?

- A way of representing the actual DOM with JavaScript Objects (vNode)
- A virtual DOM is simply a component tree of all the virtual nodes
- A way to make efficient updates to the actual DOM
- The Virtual DOM is an abstraction of the DOM. It is lightweight and detached from the browser-specific implementation details. Since the DOM itself was already an abstraction, the virtual DOM is, in fact, an abstraction of an abstraction. 🤯

SO THE VIRTUAL DOM IS FAST RIGHT?

SO WHY DO WE NEED VIRTUAL DOM?

RENDER FUNCTIONS

```
<template>
  <div>Hello, World!</div>
</template>
```

RENDER FUNCTIONS

```
<template>
  <div>Hello, World!</div>
</template>
```

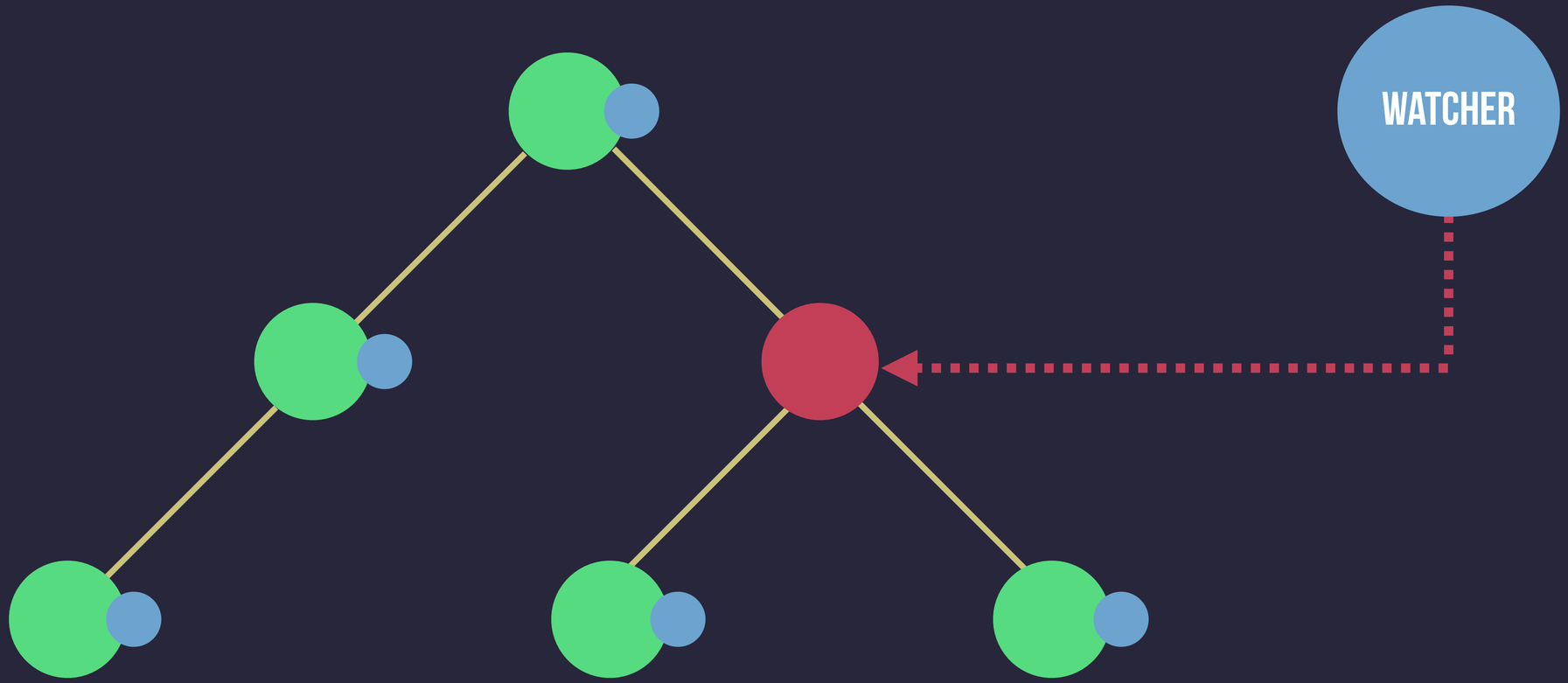
```
<script>
render(h) {
  return h('div', ‘Hello, World!’)
}
</script>
```

RENDER FUNCTIONS

- Give you full access to JavaScript
- Where you can do arbitrary logic
- Build Functional Wrapper Components
- Build pure JavaScript Higher Order Components

WHAT IS CHANGING IN VUE 3?

THE BOTTLENECK OF TRADITIONAL VIRTUAL DOM



THE BOTTLENECK OF TRADITIONAL VIRTUAL DOM

```
<template>
  <div id="content">
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">{{ message }}</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
  </div>
</template>
```

- Diff
  - Diff props of
  - Diff children of
    - Diff <p>
      - Diff props of <p>
      - Diff children of <p>
    - Repeat n times

The performance of traditional VDOM is determined by the total size of the template rather than the amount of dynamic content in it.

AFTER VUE 3 IMPROVEMENTS

```
<template>
  <div id="content">
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">{{ message }}</p>
    <p class="text">Lorem ipsum</p>
    <p class="text">Lorem ipsum</p>
  </div>
</template>
```

- Diff <p> textContent

With the new strategy, update performance is determined by the amount of dynamic content instead of the total template size.

UPDATE PERFORMANCE BENCHMARK

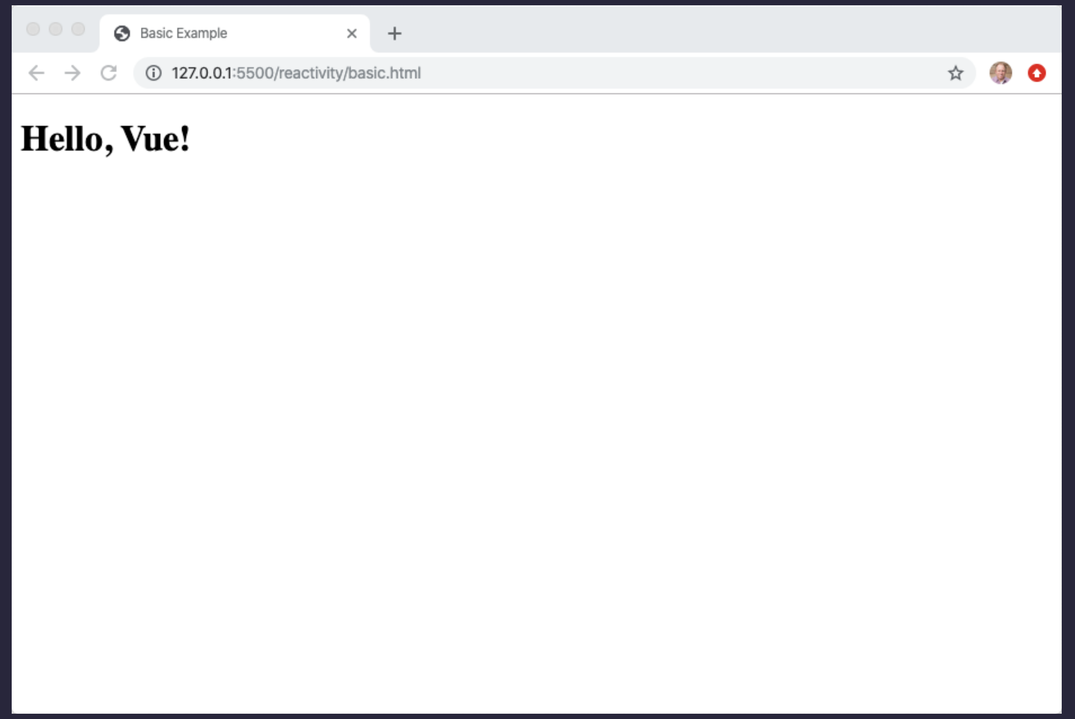
- v-for with 1000 iterations
- Inside each iteration:
  - 12 DOM elements nested 3 levels deep
  - 2 dynamic class bindings
  - 1 dynamic text interpolation
  - 1 dynamic id attribute binding
- Update all dynamic bindings, take average of 100 runs

120-130% PERFORMANCE IMPROVEMENTS

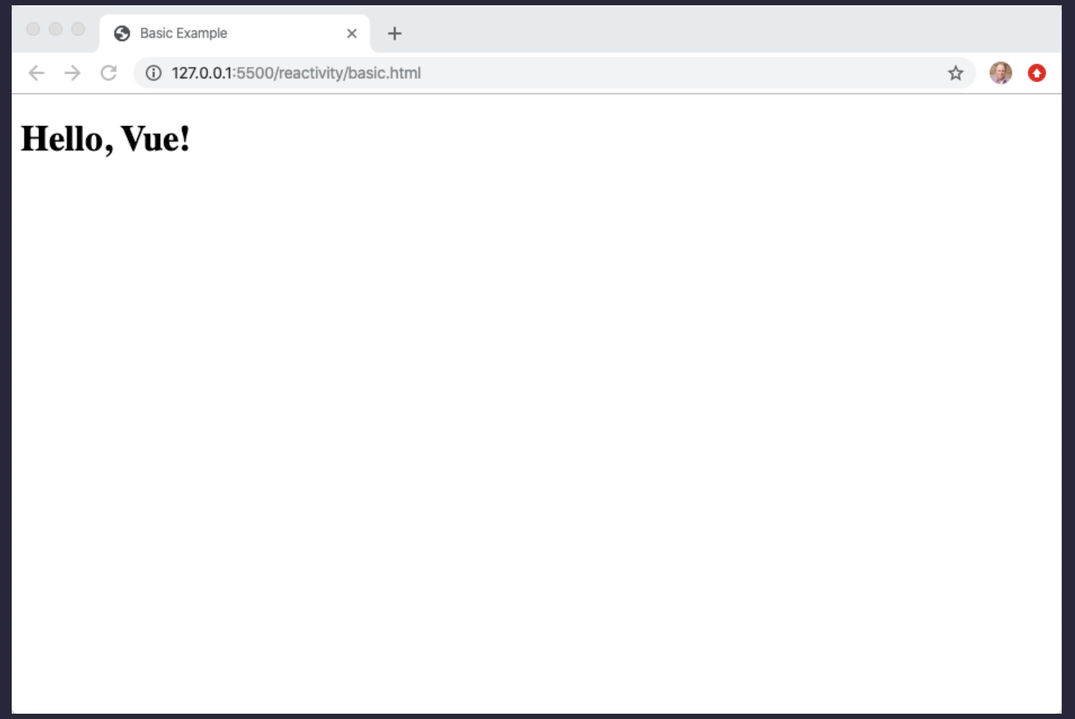
REACTIVITY SYSTEM

```
<div id="app">
  <h1>{{ title }}</h1>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        title: 'Hello, Vue!'
      }
    }
  });
</script>
```

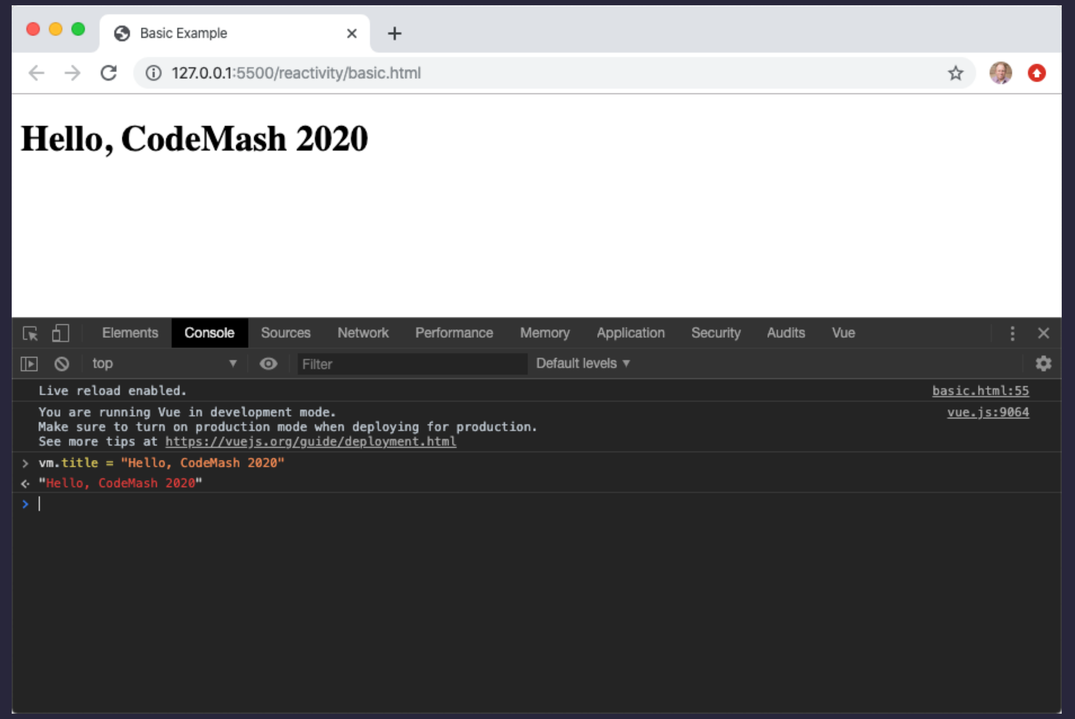
```
<div id="app">
  <h1>{{ title }}</h1>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        title: 'Hello, Vue!'
      }
    }
  });
</script>
```



```
<div id="app">
  <h1>{{ title }}</h1>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        title: 'Hello, Vue!'
      }
    }
  });
</script>
```

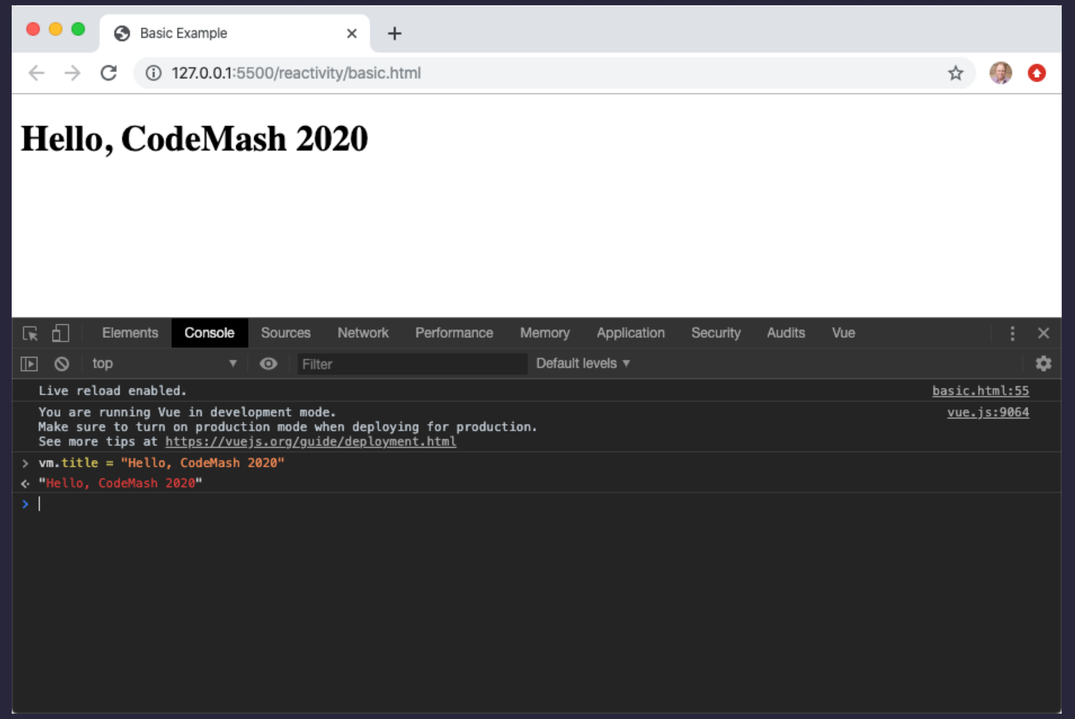


```
<div id="app">
  <h1>{{ title }}</h1>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        title: 'Hello, Vue!'
      }
    }
  });
</script>
```




```
vm.title = "Hello, CodeMash 2020"
```

```
<div id="app">
  <h1>{{ title }}</h1>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        title: 'Hello, Vue!'
      }
    }
  });
</script>
```




```
vm.title = "Hello, CodeMash 2020"
```

This works really well


Until it doesn't

HOW CHANGES ARE TRACKED IN 2.X

- Vue will walk through all the properties of your data object and convert them to getters and setters using `Object.defineProperty()`
- This is an ES5 only and un-shimmable feature and why vue doesn't support IE8 below

REACTIVE DATA

```
<div id="app">
  <p>{{ a }}</p>
  <p>{{ b }}</p>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        a: "a",
        b: "b"
      }
    }
  });
</script>
```

REACTIVE DATA

```
<div id="app">
  <p>{{ a }}</p>
  <p>{{ b }}</p>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        a: "a",
        b: "b"
      }
    }
  });
</script>
```

All properties declared in the return object are reactive

REACTIVE DATA

```
<div id="app">
  <p>{{ a }}</p>
  <p>{{ b }}</p>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        a: "a",
        b: "b"
      }
    }
  });
</script>
```

All properties declared in the return object are reactive

```
vm.a = “abc123"
```

CREATED

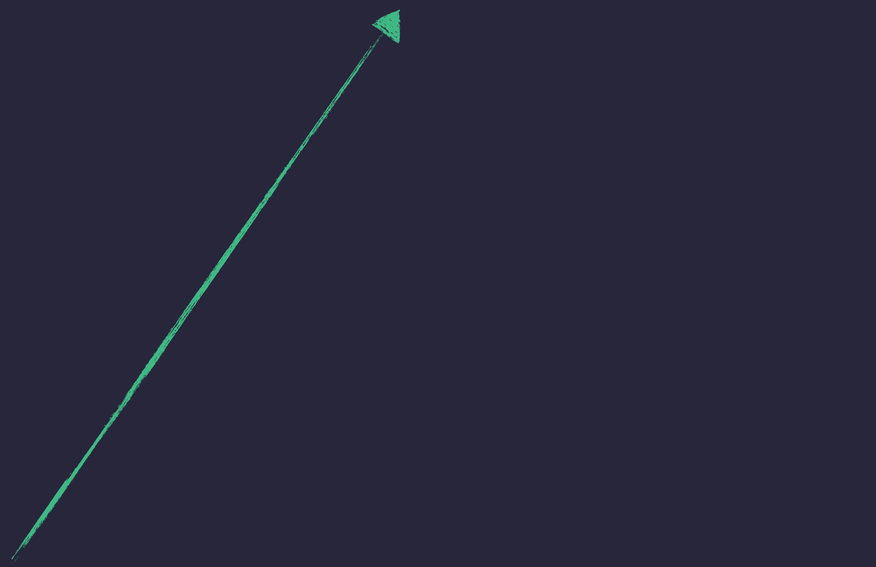
```
<div id="app">
  <p>{{ a }}</p>
  <p>{{ b }}</p>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        a: "a"
      }
    },
    created() {
      this.b = "b"
    }
  });
</script>
```

CREATED

```
<div id="app">
  <p>{{ a }}</p>
  <p>{{ b }}</p>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        a: "a" // reactive
      }
    },
    created() {
      this.b = "b" // not reactive
    }
  });
</script>
```

CREATED

```
<div id="app">
  <p>{{ a }}</p>
  <p>{{ b }}</p>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        a: "a" // reactive
      }
    },
    created() {
      this.b = "b" // not reactive
    }
  });
</script>
```



SOLUTION

```
<div id="app">
  <p>{{ a }}</p>
  <p>{{ b }}</p>
</div>

<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        a: "a",
        b: ""
      }
    },
    created() {
      this.b = "b"
    }
  });
</script>
```

Initialize to an empty string

CHANGE DETECTION CAVEATS

REACTIVITY CAVEAT #1

Adding & Removing Properties to an object

OBJECT PROPERTIES

```
<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        book: {
          name: 'My Awesome Book',
          qty: 1,
          price: 12.00
        }
      }
    },
    computed: {
      total() {
        return this.book.qty * this.book.price;
      }
    }
  });
</script>
```

```
<div id="app">
  <h1>{{ book.name }} </h1>
  <p>Qty: {{ book.qty }}</p>
  <p>Price: {{ book.price }}</p>
  <p>Total: {{ total }}</p>
</div>
```

OBJECT PROPERTIES: NOT REACTIVE

```
<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        book: {
          name: 'My Awesome Book',
          qty: 1
        }
      }
    },
    computed: {
      total() {
        return this.book.qty * this.book.price;
      }
    },
    created() {
      this.book.price = 12.00
    }
  });
</script>
```

```
<div id="app">
  <h1>{{ book.name }} </h1>
  <p>Qty: {{ book.qty }}</p>
  <p>Price: {{ book.price }}</p>
  <p>Total: {{ total }}</p>
</div>
```

OBJECT PROPERTIES: A SOLUTION

```
<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        book: {
          name: 'My Awesome Book',
          qty: 1,
          price: 0
        }
      }
    },
    computed: {
      total() {
        return this.book.qty * this.book.price;
      }
    },
    created() {
      this.book.price = 12.00
    }
  });
</script>
```

```
<div id="app">
  <h1>{{ book.name }} </h1>
  <p>Qty: {{ book.qty }}</p>
  <p>Price: {{ book.price }}</p>
  <p>Total: {{ total }}</p>
</div>
```

VUE SET & DELETE API

```
Vue.set(target, propertyName, value)

Vue.delete(target, propertyName)
```

SOLUTION USING VUE.SET

```
<script>
  const vm = new Vue({
    el: "#app",
    data() {
      return {
        book: {
          name: 'My Awesome Book',
          qty: 1
        }
      }
    },
    computed: {
      total() {
        return this.book.qty * this.book.price;
      }
    },
    created() {
      Vue.set(this.book, "price", 12.00);
    }
  });
</script>
```

```
<div id="app">
  <h1>{{ book.name }} </h1>
  <p>Qty: {{ book.qty }}</p>
  <p>Price: {{ book.price }}</p>
  <p>Total: {{ total }}</p>
</div>
```

REACTIVITY CAVEAT #2

Modifying Array Indices

ADDING TO AN ARRAY

```
<script>
  new Vue({
    el: '#app',
    data() {
      return {
        newBook: '',
        books: [
          'Atomic Habits',
          'Superfans',
          'The Unicorn Project'
        ]
      }
    },
    methods: {
      addBook() {
        this.books.push(this.newBook);
        this.newBook = '';
      }
    }
  });
</script>
```

```
<div id="app">
  <input type="text" v-model="newBook" />
  <button @click="addBook()">Add Book</button>
  <ul>
    <li v-for="book in books">{{ book }}</li>
  </ul>
</div>
```

UPDATING USING ARRAY INDICES

```
<script>
  new Vue({
    el: '#app',
    data() {
      return {
        newBook: '',
        books: [
          'Atomic Habits',
          'Superfans',
          'The Unicorn Project'
        ]
      }
    },
    methods: {
      updateBook() {
        this.books[0] = 'NEW RELEASE'
      }
    }
  });
</script>
```

```
<div id="app">
  <button @click="updateBook()">Update Book</button>
  <ul>
    <li v-for="book in books">{{ book }}</li>
  </ul>
</div>
```

Wait just a minute there Dan…
Didn't you use Array.push()
And everything worked just fine?
Why is that?

MUTATION METHODS

- `push()`
- `pop()`
- `shift()`
- `unshift()`
- `splice()`
- `sort()`
- `reverse()`

- `arr[i] = value`
- `arr.length = value`

UPDATING AN ARRAY USING VUE.SET

```
<script>
  new Vue({
    el: '#app',
    data() {
      return {
        newBook: '',
        books: [
          'Atomic Habits',
          'Superfans',
          'The Unicorn Project'
        ]
      }
    },
    methods: {
      updateBook() {
        Vue.set(this.books, 0, 'NEW RELEASE')
      }
    }
  });
</script>
```

```
<div id="app">
  <button @click="updateBook()">Update Book</button>
  <ul>
    <li v-for="book in books">{{ book }}</li>
  </ul>
</div>
```

👋 Object.defineProperty()

- Moving to a Proxy-based change detection
- You don't have to do anything to get this
- 100% faster & 1/2 the memory size
- There will be a compatibility build for IE 11

VUE 3: GOODBYE REACTIVITY CAVEATS

```
// Setting an array item by index
this.myArray[0] = newValue

// Adding a new property to an object
this.myObject[key] = value

// Deleting a property from an object
delete this.myObject[key]
```

VUE 3: STRONGER

COMPOSITION API

The first thing you need to know is that the Composition API is purely additive and can be used alongside the Options API. This is an alternative, advanced API that solves some of the limitations with the existing Options API.

Don't be this guy who went out and rewrote his entire application using the Composition API

VUE IS OPTIONS BASED

```
const vm = new Vue({})
```

VUE IS OPTIONS BASED

```
const vm = new Vue({
  el: '#app',
  data() {
    return {
      count: 0
    }
  },
  methods: {
    increment() {
      this.count++
    }
  },
  computed: {
    double() {
      return this.count * 2
    }
  }
})
```

VUE IS OPTIONS BASED

```
const vm = new Vue({
  el: '#app',
  data() {
    return {
      count: 0
    }
  },
  methods: {
    increment() {
      this.count++
    }
  },
  computed: {
    double() {
      return this.count * 2
    }
  }
})
```

- Components
- Props
- Data
- Computed Properties
- Methods
- Lifecycle Hooks

OPTIONS API LIMITATIONS

- Code Organization
- Logic Reuse
- Better TypeScript Support

CODE ORGANIZATION

CODE ORGANIZATION BY OPTIONS

```
const API_URL = "https://api.openbrewerydb.org/breweries?by_city=cleveland"
const vm = new Vue({
  el: '#app',
  data() {
    return {
      searchCity: 'Cleveland',
      breweries: [],
      orderBy: 'name'
    }
  },
  methods: {
    fetchBreweries() {
      fetch(API_URL)
        .then(response => {
          return response.json()
        })
        .then(data => {
          this.breweries = data;
        })
    },
    setOrderBy(key) {
      this.orderBy = key
    }
  },
  computed: {
    breweriesSorted() {
      if (this.orderBy != '') {
        // sorting logic
      }
      return this.breweries
    }
  },
  created() {
    this.fetchBreweries()
  }
})
```

CODE ORGANIZATION BY OPTIONS

```
const API_URL = "https://api.openbrewerydb.org/breweries?by_city=cleveland"
const vm = new Vue({
  el: '#app',
  data() {
    return {
      searchCity: 'Cleveland',
      breweries: [],
      orderBy: 'name'
    }
  },
  methods: {
    fetchBreweries() {
      fetch(API_URL)
        .then(response => {
          return response.json()
        })
        .then(data => {
          this.breweries = data;
        })
    },
    setOrderBy(key) {
      this.orderBy = key
    }
  },
  computed: {
    breweriesSorted() {
      if (this.orderBy != '') {
        // sorting logic
      }
      return this.breweries
    }
  },
  created() {
    this.fetchBreweries()
  }
})
```

Brewery DB API Feature

Sorting Feature

Options API

Composition API

LOGIC REUSE

LOGIC REUSE IN VUE 2

- Mixins
- Higher Order Components
- Renderless Components (via scoped slots)

USING THE COMPOSITION API

https://vue-composition-api-rfc.netlify.com/

API Reference

> Download the free Cheat Sheet from Vue Mastery or watch their Vue 3 Course.

`setup`

The `setup` function is a new component option. It serves as the entry point for using the Composition API inside components.

- **Invocation Timing**

  `setup` is called right after the initial props resolution when a component instance is created. Lifecycle-wise, it is called before the `beforeCreate` hook.

- **Usage with Templates**

  If `setup` returns an object, the properties on the object will be merged on to the render context for the component's template:

```html
<template>
  <div>{{ count }} {{ object.foo }}</div>
</template>

<script>
import { ref, reactive } from 'vue'

export default {
  setup() {
    const count = ref(0)
    const object = reactive({ foo: 'bar' })
```

THE SETUP FUNCTION

```
<template>
  <button @click="increment">
    Count is: {{ state.count }}, double is: {{ state.double }}
  </button>
</template>

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

export default {
  setup() {









  }
};
</script>
```

THE SETUP FUNCTION: INVOCATION TIMING

```
<template>
  <button @click="increment">
    Count is: {{ state.count }}, double is: {{ state.double }}
  </button>
</template>

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

export default {
  setup() {
  }
};
</script>
```

Called prior to beforeCreate()

Use in place of created()

THE SETUP FUNCTION: REACTIVE STATE

```
<template>
  <button @click="increment">
    Count is: {{ state.count }}, double is: {{ state.double }}
  </button>
</template>

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

export default {
  setup() {
    const state = reactive({
      count: 0,
      double: computed(() => state.count * 2)
    });

    function increment() {
      state.count++;
    }

    return { state, increment };
  }
};
</script>
```

THE SETUP FUNCTION: FUNCTIONS

```
<template>
  <button @click="increment">
    Count is: {{ state.count }}, double is: {{ state.double }}
  </button>
</template>

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

export default {
  setup() {
    const state = reactive({
      count: 0,
      double: computed(() => state.count * 2)
    });

    function increment() {
      state.count++;
    }

    return { state, increment };
  }
};
</script>
```

THE SETUP FUNCTION: LIFECYCLE HOOKS

```
import { ref, onBeforeMount, onMounted, onUpdated } from "vue";

export default {
  setup() {
    const count = ref(0);

    onBeforeMount(() => {
      console.log("onBeforeMount() called...");
    });
    onMounted(() => {
      console.log("onMounted() called...");
    });
    onUpdated(() => {
      console.log("onUpdated() called...");
    });

    setInterval(() => {
      count.value++;
    }, 1000);

    return { count };
  }
};
```

THE SETUP FUNCTION: LIFECYCLE HOOKS

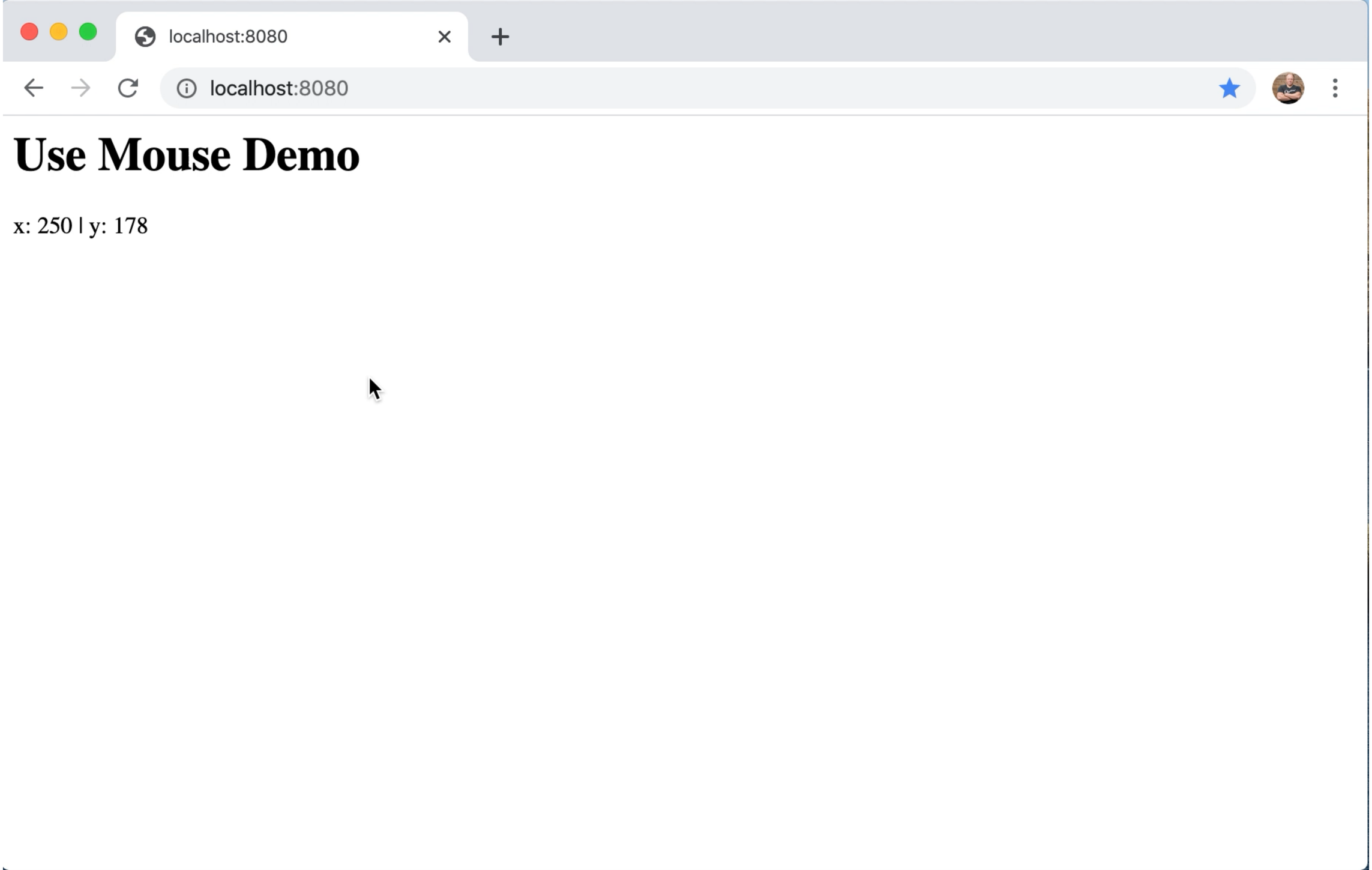
```
import { ref, onBeforeMount, onMounted, onUpdated } from "vue";

export default {
  setup() {
    const count = ref(0);

    onBeforeMount(() => {
      console.log("onBeforeMount() called...");
    });
    onMounted(() => {
      console.log("onMounted() called...");
    });
    onUpdated(() => {
      console.log("onUpdated() called...");
    });

    setInterval(() => {
      count.value++;
    }, 1000);

    return { count };
  }
};
```

THE SETUP FUNCTION: WATCH

```
import { ref, watch } from "vue";

export default {
  setup() {
    const count = ref(0);

    watch(() => {
      console.log(`watching count: ${count.value}`);
    });

    setInterval(() => {
      count.value++;
    }, 1000);

    return { count };
  }
};
```

THE SETUP FUNCTION: USAGE WITH TEMPLATES

```
<template>
  <button @click="increment">
    Count is: {{ state.count }}, double is: {{ state.double }}
  </button>
</template>

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

export default {
  setup() {
    const state = reactive({
      count: 0,
      double: computed(() => state.count * 2)
    });

    function increment() {
      state.count++;
    }

    return { state, increment };
  }
};
</script>
```


localhost:8080
localhost:8080
Use Mouse Demo
x: 250 | y: 178

```
import { ref } from 'vue'

export function useMousePosition() {
  const x = ref(0)
  const y = ref(0)
}
```

```
import { ref, onMounted, onUnMounted } from 'vue'

export function useMousePosition() {
  const x = ref(0)
  const y = ref(0)

  onMounted(() => {
    window.addEventListener('mousemove', update)
  })

  onUnmounted(() => {
    window.removeEventListener('mousemove', update)
  })
}
```

```
import { ref, onMounted, onUnMounted } from 'vue'

export function useMousePosition() {
  const x = ref(0)
  const y = ref(0)

  function update(e) {
    x.value = e.pageX
    y.value = e.pageY
  }

  onMounted(() => {
    window.addEventListener('mousemove', update)
  })

  onUnmounted(() => {
    window.removeEventListener('mousemove', update)
  })

  return { x, y }
}
```

```
<template>
  <h1>Use Mouse Demo</h1>
  <p>x: {{ x }} | y: {{ y }}</p>
</template>

<script>
import { useMousePosition } from "./use/useMousePosition";

export default {
  setup() {
    const { x, y } = useMousePosition();
    return { x, y };
  }
};
</script>
```

WHEN TO USE THE COMPOSITION API

- When you need to organize large components bye feature
- When you need to reuse code across other components
- When you want betterTypeScript Support
- Larger Teams that prefer this API

SMALLER

- ➡Global API Change
- ➡Treeshaking
- ➡TypeScript
- ➡Fragments

FASTER

- ➡Virtual DOM
- ➡Reactivity System

STRONGER

- ➡Composition API

OTHERS

- ➡Portals
- ➡Suspense
- ➡Simpler Render Functions
- ➡Optional Props Declaration
- ➡Functional Components
- ➡V-Model Updates
- ➡& So Much More

UPCOMING VUE 3 COURSE

https://www.danvega.dev/courses/vue3

Q&A