

VUEDC @ CAPITALONE (JUNE 2019)

---

# VUE.JS 101

## WIFI

Network: -

Password: -

## GETTING SETUP

Repo: <https://github.com/bencodezen/vuejs-101-workshop>

Survey: <https://bencodezen.typeform.com/to/xTRp6x>

A little bit about me...





# BEN HONG

Senior Frontend Engineer

Meltano in GitLab

[@bencodezen](https://twitter.com/bencodezen)



# BEN HONG

Senior Frontend Engineer

Meltano in GitLab

[@bencodezen](https://twitter.com/bencodezen)

- Vue.js Community Partner
- Google Developer Expert
- Mozilla TechSpeaker

A little bit about you...

# Survey Results

Before we get started...

# FORMAT

1. Learn
2. Question
3. Apply

# FORMAT

1. Learn

2. Question

3. Apply

Explanations  
Examples

Clarifications  
What-ifs

Exercises  
Experiment

# PARTICIPATION TIPS

Raise your hand for **questions** at **any** time!

*There are no wrong questions here.*

# PARTICIPATION TIPS

All slides and examples are **public**.

*No need to hurry and copy notes before we switch slides.*

# PARTICIPATION TIPS

Please **no recording.**

*Out of respect for you and your participants' privacy*

# PARTICIPATION TIPS

These are **guidelines**. Not rules.

*Feel free to question and/or disagree.*

*Your opinion and experience matter too.*

*Choose what works best for you and your team.*

# QUESTIONS?

LET'S GET STARTED!

---

VUE.JS INTRO

# WHAT ARE CLIENT-SIDE FRAMEWORKS?

## WHAT ARE CLIENT-SIDE FRAMEWORKS?

---

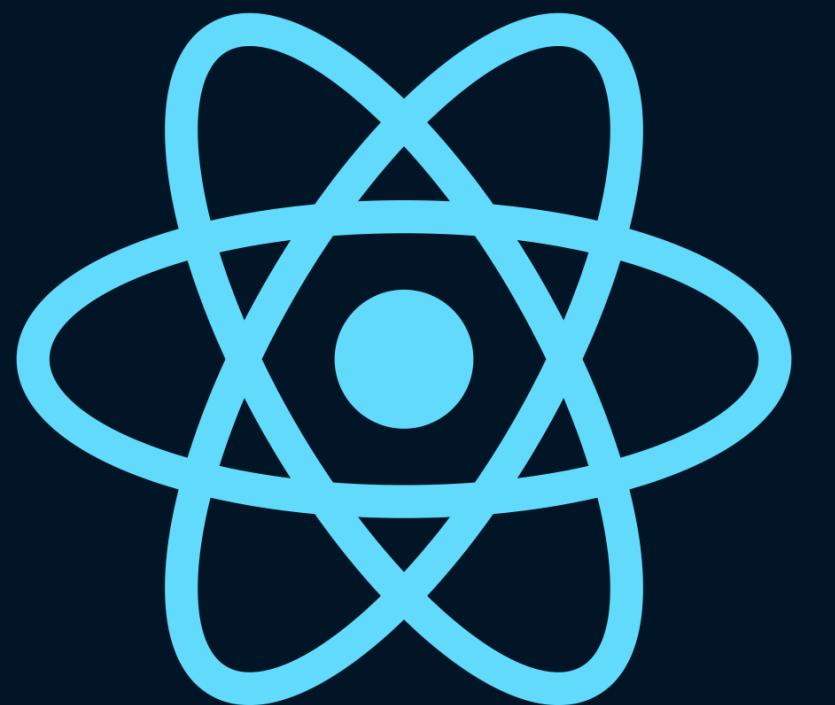
Allows frontend developers to take granular control of a site or application's user experience.

# NUMBER OF DAYS SINCE

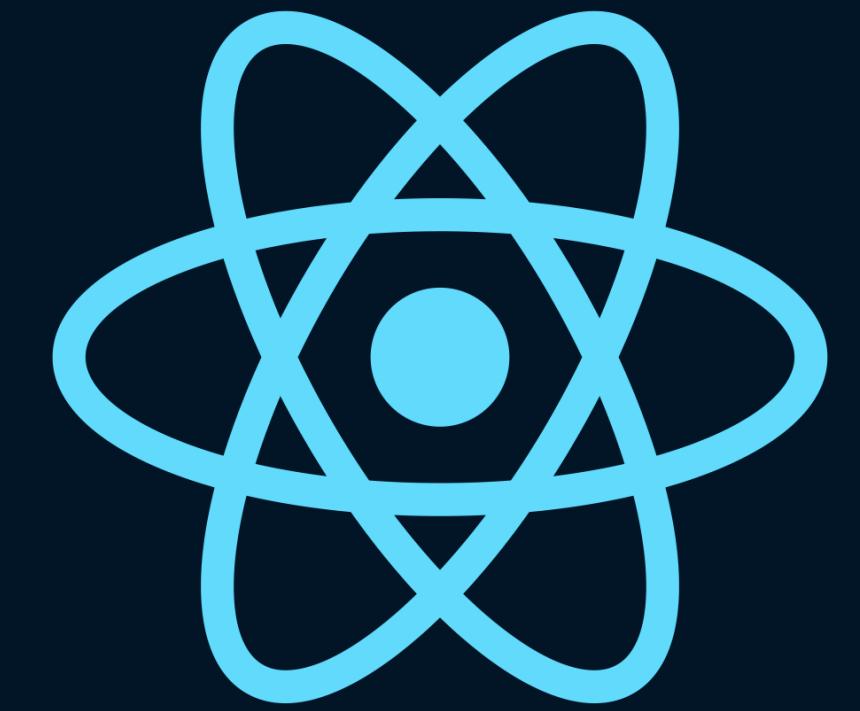
THIS DEPARTMENT  
HAS WORKED  
DAYS  
WITHOUT  
A LOST TIME  
ACCIDENT  
AVOID ACCIDENTS

# LAST NEW JS FRAMEWORK

# MOST POPULAR FRAMEWORKS



# MOST POPULAR FRAMEWORKS

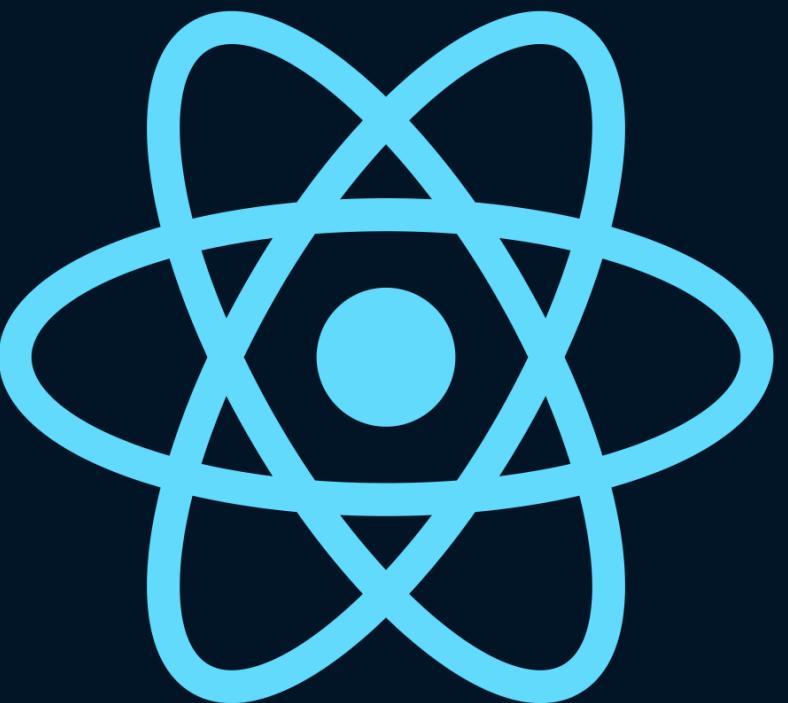


# ANGULAR OVERVIEW



- + Developer and maintained by Google
- + Easy to get something up and running pretty quickly
- Most of what was happening seemed like magic
- Opinionated on how you should built your app

# REACT OVERVIEW



Developed by Facebook and has an MIT license



Great performance due to the use of the virtual DOM



You got a lot better at vanilla JavaScript really quickly



Just getting started had a high learning curve



Unfriendly to non-JavaScript developers



High pace of development resulting in unexpected bugs

# JSX EXAMPLE

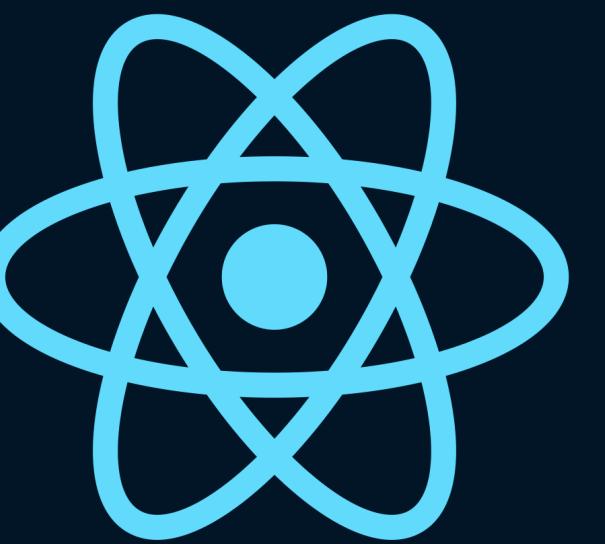
```
import React from 'react';
import { render } from 'react-dom';
import HelloWorld from './HelloWorld';

const styles = {
  fontFamily: 'Avenir, Helvetica, Arial, sans-serif',
  textAlign: 'center',
  color: '#2c3e50',
  marginTop: '60px'
};

const App = () => {
  <div style={styles} className="app">
    <h1>This is a React.js component file!</h1>
  </div>
};

export default App;
```

# COMPARING FRAMEWORKS



# COMPARING FRAMEWORKS



\*As of April 11th, 2018

# VUE OVERVIEW



- + Open-source framework with no corporate influence
- + Takes the best of both worlds and bring them together
- + It does not alienate non-JavaScript developers
- + Great performance that is on par if not better than React.js
- + Flexible and accommodating to how you prefer to build apps
- No corporate backing and has a “smaller” community compared to React / Angular

Creating and Combining Views

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

Apple Developer Discover Design Develop Distribute Support Account

SwiftUI Tutorials Creating and Combining Vie...

SwiftUI Essentials

# Creating and Combining Views

This tutorial guides you through building *Landmarks* — an iOS app for discovering and sharing the places you love. You'll start by building the view that shows a landmark's details.

To lay out the views, Landmarks uses *stacks* to combine and layer the image and text view components. To add a map to the view, you'll include a standard MapKit component. As you refine the view's design, Xcode provides real-time feedback so you can see how those changes translate into code.

Download the project files to begin building this project, and follow the steps below.

40min Estimated Time | Project files | Xcode 11 beta >

Elements Console Sources Network Performance Memory Application Security Audits Vue

Ready. Detected Vue 2.5.17.

Filter components

<Root>

<App>

<ProjectDetail> router-view: /tutorials/:id/\*

<Project key='topic://com.apple.SwiftUI/creating-and-combining-views'>

<NavigationBar>

<ProjectSection key=0>

<Hero>

<ProjectSection key=1>

<TaskList>

<ProjectSection key=2>

<ProjectSection key=3>

Select a component instance to inspect.

Console What's New

top Filter Default levels

vue-devtools Detected Vue v2.5.17

backend.js:1585

**WHICH FRAMEWORK  
SHOULD YOU  
CHOOSE?**

**IT DEPENDS**



CHOOSE VUE

**NO WAY!**

**TELL ME MORE!**

**QUESTIONS?**

AN OVERVIEW OF

---

# VUE.JS BASICS

# AN INSTANCE OF VUE.JS

```
const app = new Vue({  
  el: '#app',  
  data: {  
    firstName: 'Ben',  
    lastName: 'Hong',  
    event: 'Intro to Vue.js'  
  }  
})
```

# AN INSTANCE OF VUE.JS

```
const app = new Vue({  
  el: '#app', ←  
  data: {  
    firstName: 'Ben',  
    lastName: 'Hong',  
    event: 'Intro to Vue.js'  
  }  
})
```

Uses a CSS selector to determine where this Vue instance will live.

# AN INSTANCE OF VUE.JS

```
const app = new Vue({  
  el: '#app',  
  data: {  
    firstName: 'Ben',  
    lastName: 'Hong',  
    event: 'Intro to Vue.js'  
  }  
})
```

A data store for the Vue  
to access in the app.

# EXPRESSIONS IN VUE.JS

```
const app = new Vue({  
  el: '#app',  
  data: {  
    firstName: 'Ben',  
    lastName: 'Hong',  
    event: 'Intro to Vue.js'  
  }  
})
```

```
<!-- Using data in your template is easy! -->  
<div id="app">  
  <p>{{ firstName }} {{ lastName }} is at {{ event }}!</p>  
</div>
```

# EXPRESSIONS IN VUE.JS

```
const app = new Vue({  
  el: '#app',  
  data: {  
    firstName: 'Ben',  
    lastName: 'Hong',  
    event: 'Intro to Vue.js'  
  }  
})
```

```
<!-- This is how it renders! -->  
<div id="app">  
  <p>Ben Hong is at Intro to Vue.js!</p>  
</div>
```

# EXPRESSIONS IN VUE.JS

*<!-- Using data in your template is easy! -->*

```
<p>{{ firstName }} {{ lastName }} is at {{ conference }}!</p>
```

*<!-- You can even do math! -->*

```
<p>Let's perform calculations! {{ 12 * 12 / 4 }}</p>
```

*<!-- Any simple expression just works -->*

```
<p>{{ firstName.toUpperCase() }} IS EXCITED!</p>
```

*<!-- Basic logic works too! -->*

```
<p>{{ onSale ? price * 0.8 : price }}</p>
```

# COMPUTED PROPERTIES

```
<p>{{ onSale ? price * 0.8 : price }}</p>
```

```
<script>  COMPUTED PROPERTIES
const app = new Vue({
  el: '#app',
  data: {
    price: 100,
    onSale: false
  },
  computed: {
    salePrice() {
      if (this.onSale) {
        return this.price * 0.8
      } else {
        return this.price
      }
    }
  }
})
</script>
```

<p>{{ onSale ? price \* 0.8 : price }}</p>

# METHODS IN VUEJS

```
const app = new Vue({  
  el: '#app',  
  data: {  
    potion: 300,  
    antidote: 100,  
    revive: 1500,  
    onSale: true  
  },  
  methods: {  
    generatePrice(itemPrice) {  
      if (this.onSale) {  
        return itemPrice * 0.8  
      } else {  
        return itemPrice  
      }  
    }  
  }  
)
```

```
<div id="app">  
  <ul>  
    <li>{{ generatePrice(potion) }}</li>  
    <li>{{ generatePrice(antidote) }}</li>  
    <li>{{ generatePrice(revive) }}</li>  
  </ul>  
</div>
```

**QUESTIONS?**

# EXERCISE #1: CAPTAIN MARVEL

## INSTRUCTIONS

1. Create a new Vue instance that will live in the <main> element.
2. Create a data store that will have:
  - ▶ Captain Marvel's first name
  - ▶ Captain Marvel's last name
  - ▶ Captain Marvel's hero name
3. Create a computed property for her full name
4. Create a method to calculate how many years has passed
5. Refactor static data on site using Vue

**LET'S TALK ABOUT  
DIRECTIVES**



Directives are the part of  
Vue.js that are a bit **magical**...

# WHAT ARE DIRECTIVES?

They are **Vue specific syntax** that allow you to accomplish common goals without worrying about how it is implemented

- ▶ v-if
- ▶ v-for
- ▶ v-else-if
- ▶ v-bind
- ▶ v-else
- ▶ v-on
- ▶ v-show
- ▶ v-model

# V-IF / V-ELSE-IF / V-ELSE

Allows you to determine whether an element on the page should **exist**

```
<article>
  <h1>Dashboard</h1>
  <section v-if="userPermission === 'admin'">
    ...
  </section>
  <section v-else-if="userPermission === 'user'">
    ...
  </section>
  <section v-else>
    <p>Please login.</p>
  </section>
</article>
```

# V-SHOW

Allows you to determine whether an element on the page should be **visible**

```
<article>
```

```
  <h1>Dashboard</h1>
```

```
  <button>Trigger Modal</button>
```

```
  <aside v-show="showModal">
```

```
    <p>I toggle on and off quite a bit.</p>
```

```
  </aside>
```

```
</article>
```

### V-SHOW

Allows you to determine whether an element on the page should be **visible**

VS

### V-IF / V-ELSE-IF / V-ELSE

Allows you to determine whether an element on the page should **exist**

**QUESTIONS?**

# V-ON

Allows us to attach methods to HTML elements  
when a certain event occurs

*<!-- The long version -->*  
**<button v-on:click="alert('Alohomora')">Unlock</button>**

*<!-- The more common shorthand -->*  
**<button @click="alert('Alohomora')">Unlock</button>**

*<!-- Built in modifiers! -->*  
**<input type="text" @v-on.enter="submit" />**

**QUESTIONS?**

# V-MODEL

Allows us to use two-way binding

```
<div id="app">
  <h1>Live Message Updates!</h1>
  <p>{{ message }}</p>
  <input type="text" v-model="message">
</div>
```

```
<script>
const app = new Vue({
  el: '#app',
  data: {
    message: ''
  }
})
</script>
```

# CODE DEMO

CodePen - That Was Easy

<https://codepen.io/bencodezen/pen/GOarKO>

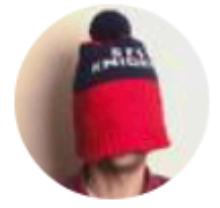
**QUESTIONS?**

# EXERCISE #2: COUNTER

## INSTRUCTIONS

1. Make the count a dynamic data property
2. Hide the text based on whether the count is even
3. Write a method to increment the count dynamically
4. Write a method to decrement the count dynamically
5. Use two way binding to change the amount that the counter can change by

# QUICK BREAK



Sam Saccone

@samccone

Following



99.7% of software development in one requirement

A user should be able to view a list of items.

10:36 AM - 1 Dec 2017

610 Retweets 1,834 Likes



26



610



1.8K



# V-FOR

Allows us to “render a list of items based on an array (or object).”

```
<script>
const app = new Vue({
  el: '#app',
  data: {
    Hogwarts: [
      'Ravenclaw',
      'Hufflepuff',
      'Slytherin',
      'Gryffindor'
    ]
  }
})
</script>
```

```
<!-- Given the data in the Vue instance -->
<div id="app">
  <ul>
    <li v-for="house in Hogwarts">
      {{ house}}
    </li>
  </ul>
</div>

<!-- Will be rendered as... -->
<ul>
  <li>Ravenclaw</li>
  <li>Hufflepuff</li>
  <li>Slytherin</li>
  <li>Gryffindor</li>
</ul>
</template>
```

**QUESTIONS?**

# V-BIND

Allows us to manipulate HTML attributes  
with dynamic data

*<!-- The long form -->*

```
<a v-bind:class="{ active: isActive }">...</a>
```

*<!-- The more common shorthand -->*

```
<a :class="{ active: isActive }">...</a>
```

*<!-- If isActive is true, it will render as... -->*

```
<a class="active">...</a>
```

**QUESTIONS?**

# EXERCISE #3: HP WORLD

## INSTRUCTIONS

1. Convert all "Static {property}" to data properties
2. Loop through the data property `wallet` to generate the same markup as the flat HTML version
3. Dynamically bind a class name that will be identical to the key as you looping through your data
4. Write methods to wire up buttons to do function commented above each button

LET'S TALK CLI

AS WE COME TO A CLOSE...

---

**FINAL THOUGHTS**



#1. ACCESSIBLE TO  
DEVELOPERS OF  
PRACTICALLY ALL  
ABILITIES

# JSX EXAMPLE

```
import React from 'react';
import { render } from 'react-dom';
import HelloWorld from './HelloWorld';

const styles = {
  fontFamily: 'Avenir, Helvetica, Arial, sans-serif',
  textAlign: 'center',
  color: '#2c3e50',
  marginTop: '60px'
};

const App = () => {
  <div style={styles} className="app">
    <h1>This is a React.js component file!</h1>
  </div>
};

export default App;
```

# SFC EXAMPLE

```
<template>
<div>
  <h1>This is a Vue Single File Component (SFC) file!</h1>
</div>
</template>

<script>
export default {
  name: 'App'
}
</script>

<style>
#app {
  font-family: 'Avenir, Helvetica, Arial, sans-serif';
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

**#2. THEIR DOCS  
ARE ONE OF THE  
BEST OUT THERE**



 Vue.js

Special Sponsors

  
 Share Code

Guide 2.x

Essentials

Installation

[Introduction](#)

What is Vue.js?

Getting Started

Declarative Rendering

Conditionals and Loops

Handling User Input

Composing with Components

Relation to Custom Elements

Ready for More?

The Vue Instance

Template Syntax

Computed Properties and Watchers

Class and Style Bindings

Conditional Rendering

List Rendering

Event Handling

Learn ▾ Ecosystem ▾ Team Support Vue ▾ Translations ▾

Patreon Sponsors


[Become a Sponsor](#)

  
Accelerate App Adoption  
ads via Carbon

## Introduction

### What is Vue.js?

Vue (pronounced /vju:/, like **v**iew) is a **progressive framework** for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only, and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with [modern tooling](#) and [supporting libraries](#).

If you'd like to learn more about Vue before diving in, we [created a video](#) walking through the core principles and a sample project.

If you are an experienced frontend developer and want to know how Vue compares to other libraries/frameworks, check out the [Comparison with Other Frameworks](#).

## Getting Started

 The official guide assumes intermediate level knowledge of HTML, CSS, and JavaScript. If you are totally new to frontend development, it might not be the best idea to jump right into a framework as your first step - grasp the basics then come back! Prior experience with other frameworks helps, but is not required.

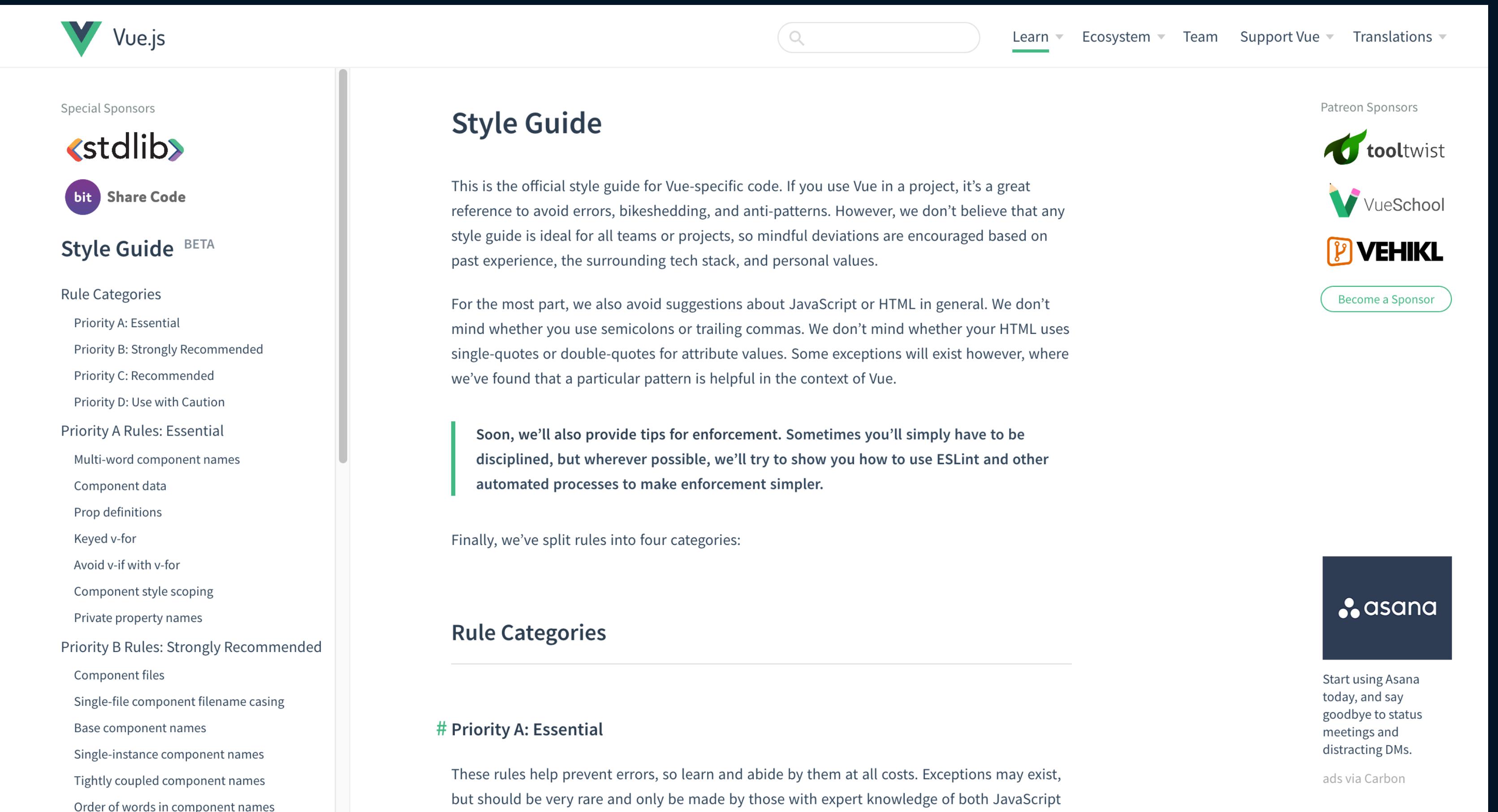
# VUE.JS DOCS

<https://vuejs.org/v2/guide/>



**#3. GIVES YOU  
WHAT YOU WANT  
**WHEN YOU NEED IT,**  
THEN IT GETS OUT  
OF YOUR WAY.**

Paraphrased from Sarah Drasner



The screenshot shows the official Vue.js Style Guide page. At the top, there's a navigation bar with the Vue.js logo, a search bar, and links for Learn, Ecosystem, Team, Support Vue, and Translations. Below the navigation, there are sections for Special Sponsors (stdlib, bit Share Code) and Patreon Sponsors (tooltwist, VueSchool, VEHIKL). A "Become a Sponsor" button is also present. The main content starts with a section titled "Style Guide" which includes a brief introduction about the guide's purpose and the avoidance of general JavaScript/HTML rules. It then discusses rule categories, specifically Priority A (Essential) and Priority B (Strongly Recommended), with examples like multi-word component names and single-instance component names. A sidebar on the left lists these categories and their sub-rules. A callout box on the right provides information about enforcement and tools like ESLint. The page ends with a "Rule Categories" section and a "Priority A: Essential" section.

**Style Guide**

This is the official style guide for Vue-specific code. If you use Vue in a project, it's a great reference to avoid errors, bikeshedding, and anti-patterns. However, we don't believe that any style guide is ideal for all teams or projects, so mindful deviations are encouraged based on past experience, the surrounding tech stack, and personal values.

For the most part, we also avoid suggestions about JavaScript or HTML in general. We don't mind whether you use semicolons or trailing commas. We don't mind whether your HTML uses single-quotes or double-quotes for attribute values. Some exceptions will exist however, where we've found that a particular pattern is helpful in the context of Vue.

Soon, we'll also provide tips for enforcement. Sometimes you'll simply have to be disciplined, but wherever possible, we'll try to show you how to use ESLint and other automated processes to make enforcement simpler.

Finally, we've split rules into four categories:

## Rule Categories

### # Priority A: Essential

These rules help prevent errors, so learn and abide by them at all costs. Exceptions may exist, but should be very rare and only be made by those with expert knowledge of both JavaScript

**Special Sponsors**

 stdlib

 bit Share Code

**Style Guide BETA**

**Rule Categories**

- Priority A: Essential
- Priority B: Strongly Recommended
- Priority C: Recommended
- Priority D: Use with Caution

Priority A Rules: Essential

- Multi-word component names
- Component data
- Prop definitions
- Keyed v-for
- Avoid v-if with v-for
- Component style scoping
- Private property names

Priority B Rules: Strongly Recommended

- Component files
- Single-file component filename casing
- Base component names
- Single-instance component names
- Tightly coupled component names
- Order of words in component names

**Patreon Sponsors**

 tooltwist

 VueSchool

 VEHIKL

[Become a Sponsor](#)



Start using Asana today, and say goodbye to status meetings and distracting DMs.

ads via Carbon

# VUE.JS STYLE GUIDE

<https://vuejs.org/v2/style-guide/>

**Vue.js is the most  
compassionate framework  
in the market right now.**

# SO WHAT ELSE IS THERE TO LEARN ABOUT?

- ▶ Props
- ▶ Slots
- ▶ Watch Property
- ▶ Custom Events
- ▶ Lifecycle Hooks
- ▶ Managing CSS in Vue.js
- ▶ State Management
- ▶ Animations
- ▶ Render Function
- ▶ Component Libraries
- ▶ Testing
- ▶ And much more!

# RECOMMENDED RESOURCES

## Paid

- ▶ [Vue Mastery](#)
- ▶ [Vue School](#)
- ▶ [Frontend Masters: Introduction to Vue.js](#)
- ▶ [Udemy: Vue.js 2 - The Complete Guide](#)

## Free

- ▶ [Official Vue.js Guide](#)
- ▶ [VueMastery: Intro to Vue.js](#)

# QUESTIONS?

# THANKS EVERYONE!

@BENCODEZEN

<https://bencodezen.typeform.com/to/IhZjnU>

