

Vue.js 101

Ben Hong

Senior UI Engineer at Politico

@bencodezen

Be sure to clone the repo if you want to follow
along and/or work on the exercises!

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

Slides: <https://slides.com/bencodezen/vuejs-101-tutorial/live>

Before we get started...

You can find the resources below:

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

Slides: <https://slides.com/bencodezen/vuejs-101-tutorial>

And for those posting on social media:

[#vuejs101](#)



Ben Hong

Senior UI Developer @ POLITICO

@bencodezen



Ben Hong

Senior UI Developer @ POLITICO

@bencodezen



VueDC

<https://www.vuedc.io>



Ben Hong

Senior UI Developer @ POLITICO

@bencodezen



VueDC

<https://www.vuedc.io>



VueMeetups

<https://www.vuemeetups.org>

A Little About You...

“The best way
to learn how to
write code is
to write code.”

— Kyle Simpson
Open Web Evangelist

“There is no
substitute for
hands-on
experience.”

— Kelsey Hightower
Staff Developer Advocate,
Google Cloud Platform, Google

O'REILLY
Velocity

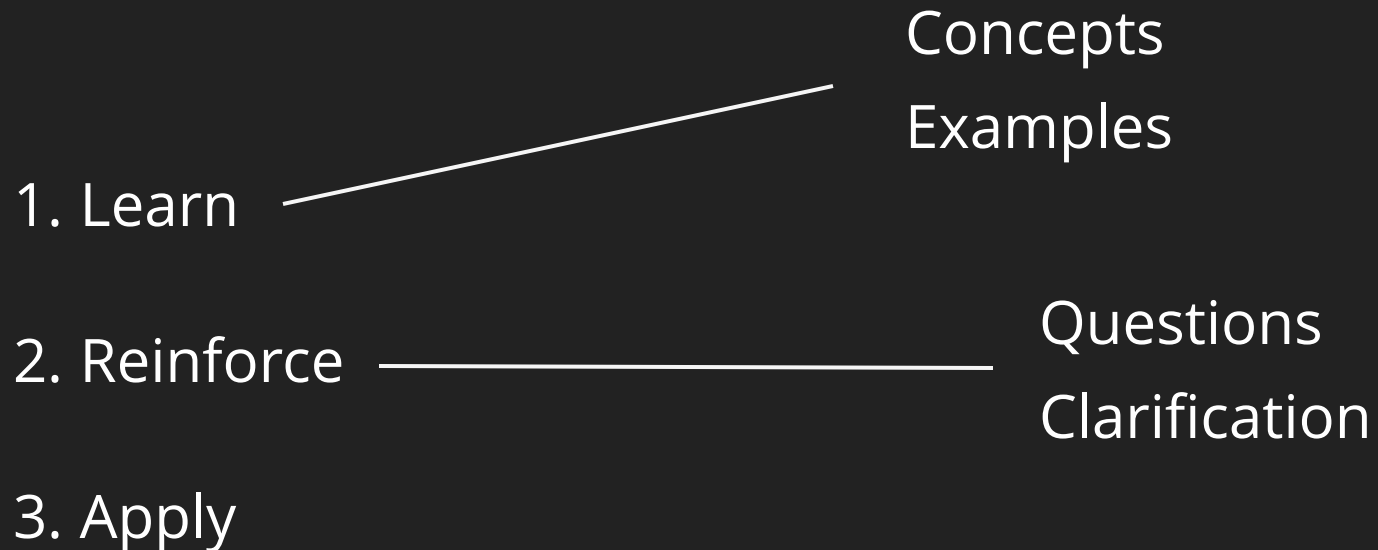
Learning Format

1. Learn
2. Reinforce
3. Apply

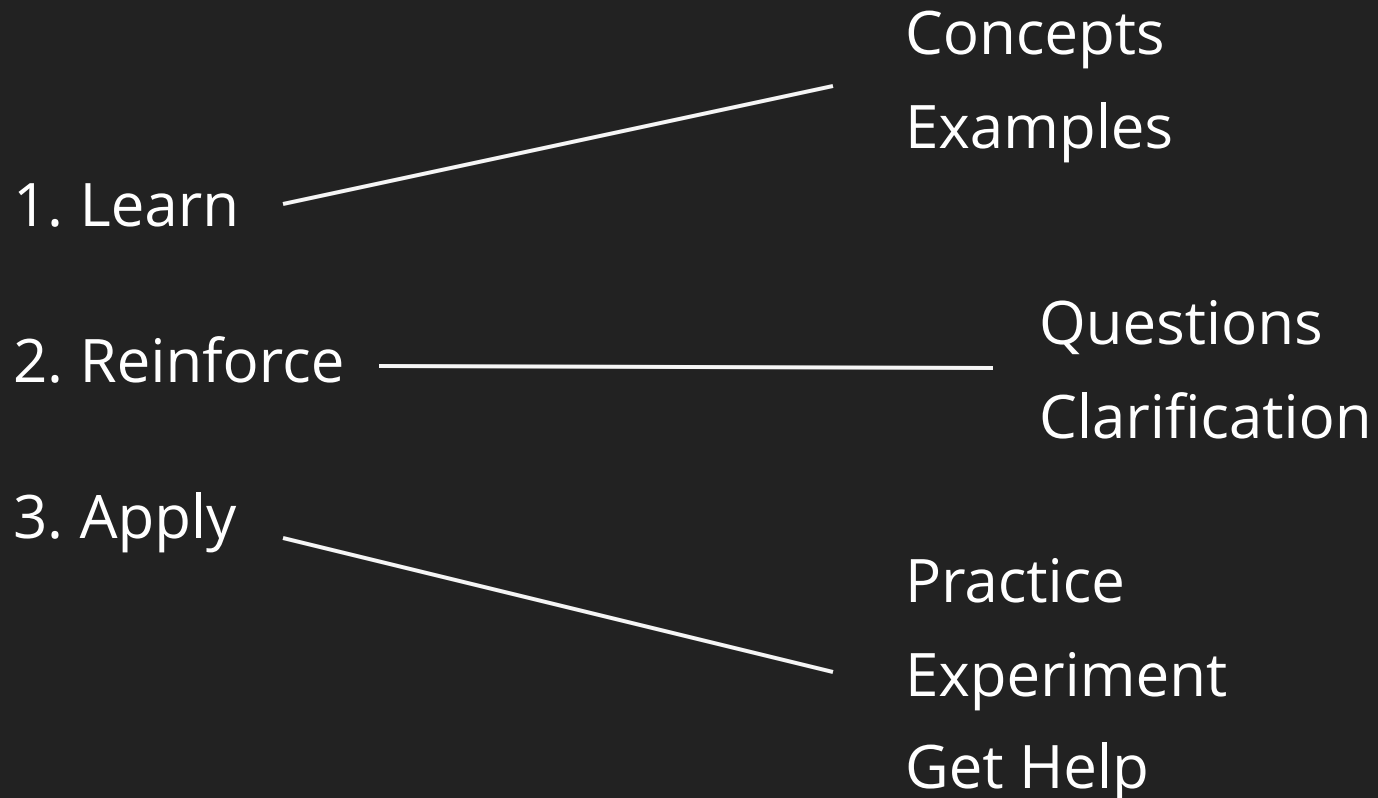
Learning Format



Learning Format



Learning Format



Participation Guidelines

1. Raise your hand for questions at any time!
2. All examples are public (no need to copy down code examples)
3. Please no recording (for the privacy of participants)

Q&A

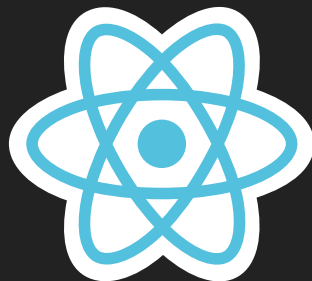
A Short Introduction

Vue.js

Vue is a **progressive framework** for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be **incrementally adoptable**.

Why use
client-side
frameworks?

Allows you to control
every aspect of your site's
user experience



ember[®]

Knockout.

NUMBER OF DAYS SINCE



LAST NEW JS FRAMEWORK

Most Popular Client-Side Frameworks



Angular Overview



Angular Overview



Developed and maintained by Google



Easy to get something up and running pretty quickly

Angular Overview



Developed and maintained by Google



Easy to get something up and running pretty quickly



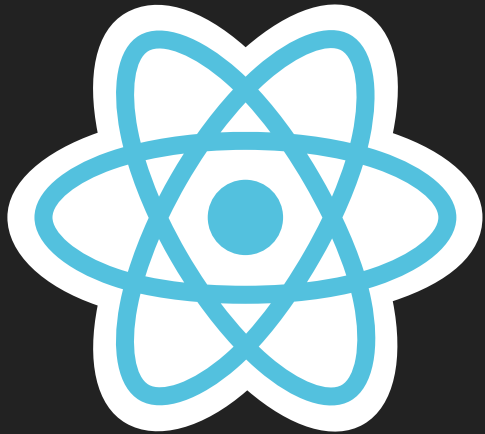
Most of what was happening seemed more like magic



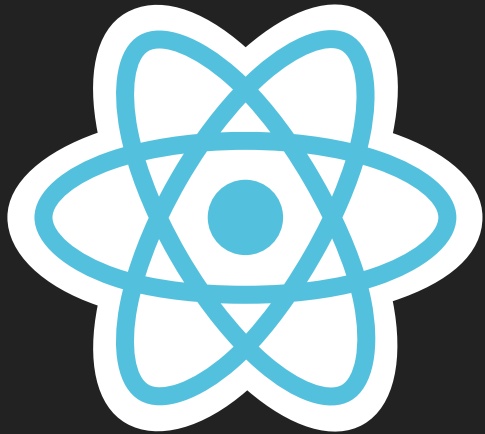
Opinionated on how you should build your app

React

Overview



React Overview



Large community base and has a model for cross-platform development

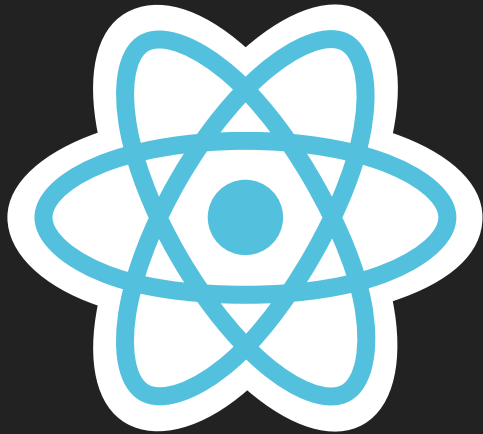


Great performance due to the use of the virtual DOM



You got a lot better at vanilla JavaScript and ES6 very quickly

React Overview



Large community base and has a model for cross-platform development



Great performance due to the use of the virtual DOM



You got a lot better at vanilla JavaScript and ES6 very quickly



There is a high learning curve to simply get started



Unfriendly to developers who are not well versed in JavaScript ES6+



It's a bit like the Wild West as far as how things should be done

Vue Overview



Vue Overview



An open-source framework with no corporate influence



Takes the best of both worlds and brings 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

Vue Overview



An open-source framework with no corporate influence



Takes the best of both worlds and brings 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



Does not currently have a formal model for cross-platform development

Which
framework
should you choose?

It depends...



Choose Vue

Vue is the most **compassionate** framework in the market right now because it allows you to choose **what's best for you.**

Ben Hong

The Basics of Vue.js

A Vue Instance

```
// Declare an instance of Vue  
const app = new Vue({  
  // Set the location your app will reside in  
  el: '#app',  
  // Contains the data you will use in the app  
  data: {  
    conference: 'FluentConf'  
  }  
})
```

Declarative Rendering

```
<div id="app">  
  <!-- Rendering data is so easy! -->  
  <h1>Greetings {{ conference }}!</h1>  
  <!-- You can do math! -->  
  <p>{{ 12 * 12 }}</p>  
  <!-- Ternary expressions work too! -->  
  <p>Hello {{ conference ? conference : 'World' }}</p>  
</div>
```

Exercise #1: Vue Basics

Convert a flat HTML file into a basic Vue app

Instructions

1. Open 01-captain-marvel.html
2. Create a new Vue instance
3. Attach to the correct HTML element
4. Extract the following data into the app:
 - Hero Name
 - Real Name
 - Height, Eye and Hair Color
 - Citizenship
 - Place of Birth
 - Powers
 - Abilities

Extra Credit

1. Refactor data model to accommodate the table format
2. Refactor abilities to use an Array of snippets to improve reuse
3. Add a new data property 'Gender' to the app and use it to determine the pronoun being used in the text
4. Add a new data property 'Location' that generates a random longitude and latitude each time the page is reloaded

Let's Talk About Directives



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

What are **directives** exactly?

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

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

v-if

v-else

v-else-if

“ It ensures that event listeners and child components inside the conditional block are properly destroyed and re-created during toggles.

```
<div>
  <p v-if="userType === 'admin'">
    Admin Panel
  </p>
  <p v-else-if="userType === 'user'">
    User Panel
  </p>
  <p v-else>
    Login
  </p>
</div>
```

v-show

“ The element is always rendered regardless of initial condition, with CSS-based toggling.

```
<div v-show="modalActive" class="modal">  
  <h2>Here is my modal window</h2>  
</div>
```

v-if

v-else

v-else-if

v-show

v-if

v-else

v-else-if

- Higher toggle costs

v-show

- Higher initial cost

v-if

v-else

v-else-if

- Higher toggle costs
- It's lazy, so it only renders when the condition is true

v-show

- Higher initial cost
- Renders on the page regardless

v-if

v-else

v-else-if

- Higher toggle costs
- It's lazy, so it only renders when the condition is true
- Ensures event listeners and child components are properly destroyed

v-show

- Higher initial cost
- Renders on the page regardless
- Uses CSS to toggle the display of the element



Sam Saccone 

@samcccone

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() {
      return {
        houses: [
          'Gryffindor',
          'Hufflepuff',
          'Ravenclaw',
          'Slytherin',
        ]
      }
    }
  })
</script>
```

```
<!-- Given this markup -->
<div id="app">
  <ul>
    <li v-for="house in houses"
        class="hogwarts-house">
    >
      {{ house }}
    </li>
  </ul>
</div>
```

```
<!-- It will render this -->
<div id="app">
  <ul>
    <li class="hogwarts-house">
      Gryffindor
    </li>
    <li class="hogwarts-house">
      Hufflepuff
    </li>
    <li class="hogwarts-house">
      Ravenclaw
    </li>
    <li class="hogwarts-house">
      Slytherin
    </li>
  </ul>
</div>
```


v-bind

Allow us to manipulate
HTML attributes with
dynamic data

```
<!-- The long form -->  
<nav>  
  <a href="/" v-bind:class="{ active: isActive }">  
    Home  
  </a>  
</nav>
```

```
<!-- The common shortcut -->  
<nav>  
  <a href="/" :class="{ active: isActive }">  
    Home  
  </a>  
</nav>
```

```
<!-- If isActive is true, it will render -->  
<nav>  
  <a href="/" class="active">  
    Home  
  </a>  
</nav>
```

v-bind

Allow us to manipulate
HTML attributes with
dynamic data

```
<!-- You can bind any attribute... -->



<a :href="canonicalUrl">{{ linkText }}</a>

<div :data-company-id="companyId">
  ..
</div>

<div :id="'list-' + id">
  ...
</div>

<section :style="'columns: ${items.length}'">
  ...
</section>
```

v-on

Allow us to attach JavaScript functions to common events

```
<!-- The long form -->  
<button v-on:click="alert('Alohomora!')">  
  Cast Unlock Spell!  
</button>  
  
<!-- The common shortcut -->  
<button @click="alert('Alohomora!')">  
  Cast Unlock Spell!  
</button>
```

v-on

Allow us to attach JavaScript functions to common events

```
<script>
  const app = new Vue({
    el: '#app',
    data() {
      return {
        houses: [ ... ]
      }
    },
    methods: {
      castUnlockSpell() {
        alert('Alohomora')
      }
    }
  })
</script>
```

```
<!-- The long form -->
<button v-on:click="castUnlockSpell">
  Cast Unlock Spell!
</button>

<!-- The common shortcut -->
<button @click="castUnlockSpell">
  Cast Unlock Spell!
</button>
```

v-on

Common DOM events that you most likely be using a fair amount

- @click
- @keyup
- @keydown
- @input
- @change
- @submit

v-on

Modifiers are a syntactic sugar to help with common functionality

```
<!-- the click event's propagation will be stopped -->  
<a @click.stop="doThis"></a>  
  
<!-- the submit event will no longer reload the page -->  
<form @submit.prevent="onSubmit"></form>  
  
<!-- only call `vm.submit()` when the `keyCode` is 13 -->  
<input @keyup.13="submit">  
  
<!-- same as above -->  
<input @keyup.enter="submit">  
  
<!-- the click event will be triggered at most once -->  
<a v-on:click.once="doThis"></a>
```

v-model

Allows us to use two-way
data binding

```
<div id="app">
  <input v-model="name" type="text" />
  <button @click="castSpell(spellName)">
    Cast {{ spellName }}!
  </button>
</div>

<script>
  const app = new Vue({
    el: '#app',
    data() {
      return {
        houses: [ ... ],
        spellName: 'Alohomora'
      }
    },
    methods: {
      castSpell(spell) {
        alert(spell)
      }
    }
  })
</script>
```

Q&A

Exercise #2: Counter

Build a counter app

Instructions

1. Open 02-counter-app.html
2. Create a new Vue instance
3. Attach to the correct HTML element
4. Functionality:
 - Render dynamic count data
 - Add ability to increment count
 - Add ability to decrement count

Extra Credit

1. Add the ability to reset the count data
2. Allow the user to dynamically set the amount that the counter is incremented or decremented by
3. Allow the user to save a snapshot of the current count and restore it if desired
4. Allow the user to generate a list of snapshots that can be restored at any point

Quick Break!

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

Slides: <https://slides.com/bencodezen/vuejs-101-tutorial>

Quick Break!

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

Slides: <https://slides.com/bencodezen/vuejs-101-tutorial>

Please fill out this quick survey!

<https://bencodezen.typeform.com/to/Q1pcsZ>

Quick Break!

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

Slides: <https://slides.com/bencodezen/vuejs-101-tutorial>

Please fill out this quick survey!

<https://bencodezen.typeform.com/to/Q1pcsZ>

Quick Debrief

Quick Debrief

- A little more about you...

Quick Debrief

- A little more about you...
- There are many ways to accomplish the things we are doing in this tutorial

Let's talk about
Vue.js CLI
applications



Vue CLI

 Standard Tooling for Vue.js Development

Get Started →

<https://cli.vuejs.org/>

1. bhong@C02PF1P7G3QP: ~/Projects (zsh)

Last login: Tue Jun 12 09:25:54 on ttys000

~/Projects vue init










Vue.js App

Tour

▲ SAMPLE-APP

- build
- config
- node_modules

▲ src

- ▲ assets
 -  logo.png
- ▲ components
 - ▼ HelloWorld.vue
 - ▼ App.vue
-  main.js
- static
-  .babelrc
-  .editorconfig
-  .gitignore
-  .postcssrc.js
-  index.html
-  package.json
-  README.md

Single File Component *.vue

```
<template>
  <div id="app" class="wrapper">
    
    <h1>Hello {{ conference }}!</h1>
  </div>
</template>

<script>
export default {
  name: 'App',
  data() {
    return {
      conference: 'FluentConf'
    }
  }
}
</script>

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

Q&A

Exercise #3: Intro to SFCs

Migrate your Counter app into the CLI

Instructions

1. Open App.vue
2. Copy over your:
 - Template
 - Vue Instance

Extra Credit

1. Migrate the CSS over to the app

If you think **SFCs** are
cool now...

```
<template>
  <div id="app">
    
    <hello-world />
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld'

export default {
  name: 'App',
  components: {
    'hello-world': HelloWorld
  }
}
</script>

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



```
<template>
  <div id="app">
    
    <hello-world />
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld'

export default {
  name: 'App',
  components: {
    'hello-world': HelloWorld
  }
}
</script>

<style lang="scss">
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

```
<template>
  <div id="app">
    
    <hello-world />
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld'

export default {
  name: 'App',
  components: {
    'hello-world': HelloWorld
  }
}
</script>

<style lang="scss" scoped>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

```
<template>
  <div id="app">
    
    <hello-world />
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld'

export default {
  name: 'App',
  components: {
    'hello-world': HelloWorld
  }
}
</script>

<style lang="scss" module>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

Exercise #4: Create a SFC

Refactor out your counter app to a SFC!

Instructions

1. Create a SFC in the /component directory called Counter.vue
2. Migrate all Counter properties to this new SFC
 1. Template
 2. Data
 3. Methods
 4. CSS (Optional)
3. Import your component into App.vue so that your component renders on the page!

You can pass
data to your SFCs!

```

<template>
  <div id="app">
    
    <hello-world message="Hello FluentConf!" />
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld'

export default {
  name: 'App',
  components: {
    'hello-world': HelloWorld
  }
}
</script>

```

```

<template>
  <div class="hello">
    <h1>{{ message }}</h1>
  </div>
</template>

<script>
export default {
  name: 'HelloWorld',
  props: {
    message: {
      type: String,
      required: true
    }
  },
  data () {
    return {
      msg: 'Welcome to Your Vue.js App'
    }
  }
}
</script>

```

And just when you
thought **SFCs** could
not be cooler...

You can use
directives on
SFCs too!

(/◉ 7 ◉)/*:·° ✨

```
<template>
  <div id="app">
    
    <hello-world
      v-for="message in messages"
      :message="message"
      :key="message"
    />
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld'

export default {
  name: 'App',
  components: {
    'hello-world': HelloWorld
  },
  data() {
    return {
      messages: [
        'Thou count to three, no more, no less',
        'Three shall be the number thou shalt count',
        'And the number of the counting shall be three'
      ]
    }
  }
}
</script>
```


Exercise #5: Sign Up Form

Convert flat web page file to a SFC

Instructions

1. Locate 05-sign-up.html and 05-sign-up.css for the template and styles
2. Create a new component called AccountCreation.vue and import it into App.vue
3. Functionality to build:
 - Toggle visibility of error messages based on criteria per input field
 - Make Submit button dynamically disable based on login

Extra Credit

1. Add additional password logic
2. Create password verification functionality with the appropriate error toggling ability
3. Refactor the UI to reduce repetition of HTML
4. Dynamically style the input fields based on whether error or success

Vue.js has its own
DevTools extension...

Exercise #6: To Do App

Build a to do list app from scratch

Instructions

1. Create a new Todo component called `Todo.vue`
2. Import the component into the page and make sure it renders:
3. Basic Functionality
 - App should render a list of tasks
 - User should be able to add new tasks
 - User should be able to complete tasks
 - Dynamically style tasks that are completed
 - User should be able to delete tasks

Extra Credit

1. Refactor HTML into single file components as you see fit to reduce clutter and increase reuse
2. Create a "Trash Can" list that keeps the items the user has "deleted" so that they can undo the deletion
3. Add "Due Date" property to tasks
4. Dynamically style the task if it is overdue
5. Add "Tags" property to tasks that allow you to sort and filter your tasks

So let's do a quick
revue...

What We Covered Today

- Concepts
 - Basics of Vue.js
 - Declarative Rendering
 - Data Store
 - Directives
 - Methods
 - Vue CLI
 - Single File Components
 - Vue DevTools
- Built three apps
 1. Counter
 2. Sign Up Form UI
 3. To Do List

Q&A

Congratulations!



You are ready to build and work on
Vue.js applications!

But wait,
there's more!

Concepts

- Computed Properties
- Filters
- Props
- Mixins
- Lifecycle Methods
- State Management
- Custom Directives
- Routing
- Animations

Workflow

- App Architecture w/ Vue.js
- Testing with Vue.js
- Managing Styles w/ Vue.js
- Animate All Things w/ Vue.js
- Popular Vue.js Tools
 - vuex
 - vetur
 - Vue DevTools

Gives me what I want
when I need it, and then it
gets out of my way.

Sarah Drasner (@sarah_edo)

Additional Resources

- **Official Vue.js Docs**
 - <https://vuejs.org/>
- **FEM: Introduction to Vue.js**
 - <https://frontendmasters.com/courses/vue/>
- **Udemy: Vue.js Courses**
 - <https://www.udemy.com/courses/search/?q=vuejs>
- **Vue.js Discord Channel**
 - <https://vue-land.js.org/>
- **Vue.js Meetups**
 - <https://www.vuemeetups.org>

Vue is the most **compassionate** framework in the market right now because it allows you to choose **what's best for you.**

Ben Hong

Thank you!

If you have any additional questions, please feel free to reach out to me.

@bencodezen