



# BUILDING WEB APPS WITH VUE.JS

Raymond Camden

# WHO AM I?

Raymond Camden

Developer Evangelist for HERE  
Technologies

Blogging at [raymondcamden.com](http://raymondcamden.com)

Tweeting at [raymondcamden](https://twitter.com/raymondcamden)



# GAMEPLAN

Go over "requirements"

Talk about Vue at a high level (somewhat quick!)

Vue Syntax (basic stuff, but a lot of em!)

Components

Working with the CLI and Applications

Routing

Vuex (State management)

Vue 3.0 – What to Expect

Wrap Up

# REQUIREMENTS

Assets (most code, slide deck, etc) - <https://github.com/cfjedimaster/vue-workshop>

## Exercises:

- An editor (any editor is fine but Visual Studio Code is best)
- Or Codepen (codepen.io) or Code Sandbox (codesandbox.io)
- A web server (npm i -g httpster)

Web browser

# WHY VUE?



Let's back up...





SIMPLE |





**NO BUILD PROCESS**





SCALABLE



# QUICK VUE FACTS

Created by Evan You (@youyuxi)

Released in February of 2014

"Currently" is version 2.6.11

Main site: [vuejs.org](https://vuejs.org)

GitHub: <https://github.com/vuejs>



**BRACE YOURSELF**

**VUE3 IS COMING**

# WHAT DOES VUE DO?

Two-way binding between your DOM and JavaScript

Template Language (aka Declarative Rendering)

And I know I said this - but it does more. A lot more.

# INSTALLATION

`npm install`

`cli install`

Download from site

Copy and paste a script tag FTW!!!!1!





# <SCRIPT> STYLE

```
<!-- dev -->
```

```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

```
<!-- prod -->
```

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.11"></script>
```

# WRITE SOME CODE!

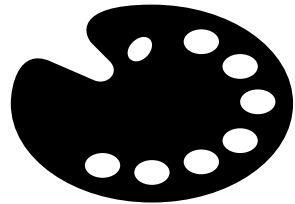
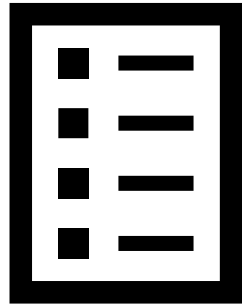
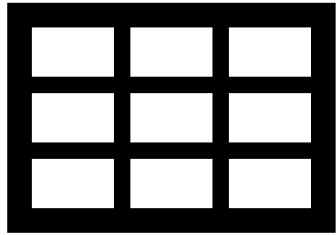
Identify where you're going to render stuff

Setup some data

Render that data

Add events (ie on this do that) and other fancy stuff

## Willy Nilly DOM Manipulation

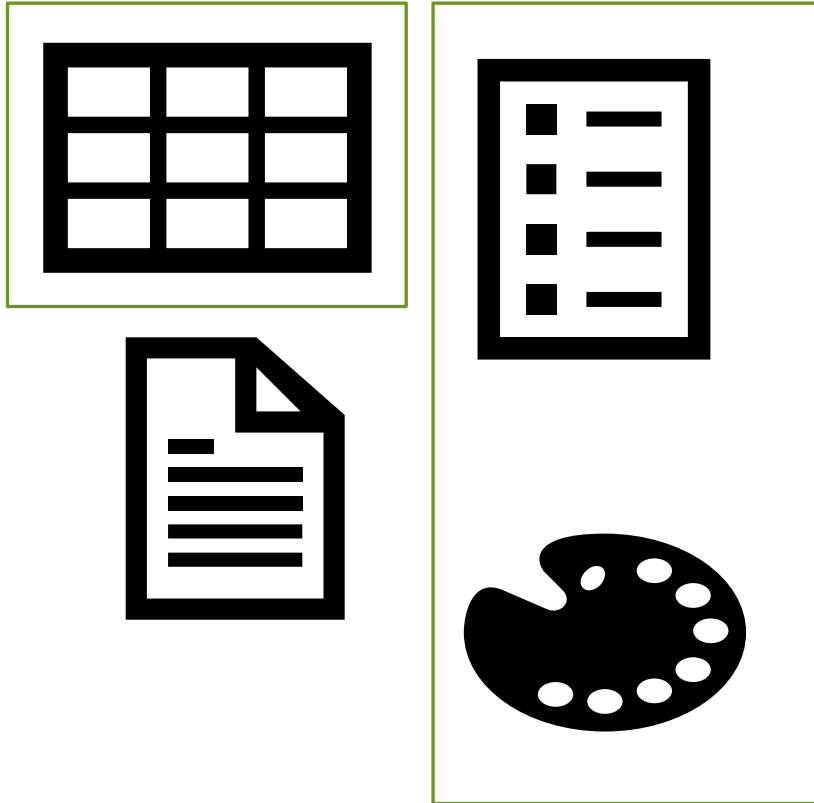


`$('#somediv').html('woot');`

`$('#someotherdiv').val(2);`

`$('#yetanotherdiv').html('<b>This is awesome</b>');`

## Defining Vue's Place...





# EXAMPLES

<http://bit.ly/rayvuedemos>



```
<!-- header stuff -->  
<div id="app">  
</div>  
<!-- footer stuff -->
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    name: "Ray",  
    age: 44  
  }  
});
```

SOME CODE...

```
<!-- header stuff -->  
<div id="app">  
My name is {{name}} and  
I'm {{age}} years old.  
</div>  
<!-- footer stuff -->
```

```
var app = new Vue({  
  el: '#app',  
  data: {  
    name: "Ray",  
    age: 44  
  }  
});
```

# SOME MORE CODE...

```
<div id="app">
  My name is {{name}}.
  <span v-if="age > 40">
    I remember floppies.
  </span>
</div>
```

```
<div id="app">
  My name is {{name}}.
  <span v-if="age > 40">
    I remember floppies.
  </span><span v-else>
    What's a floppy?
  </span>
</div>
```

# TEMPLATE DIRECTIVES - CONDITIONALS

```
<div id="app">
  My name is {{name}}. Here
  are things I like:
  <ul>
    <li v-for="thing in things">
      {{thing}}
    </li>
  </ul>
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    name: "Ray",
    things: [
      "cats",
      "star wars",
      "beer"
    ]
  }
});
```

# TEMPLATE DIRECTIVES - LOOPS

# WORKING WITH FORM FIELDS

v-model links to data field

Works in all kinds of form fields

Bigger demo:

<https://www.raymondcamden.com/2020/01/27/vue-and-form-fields>

```
<div id="app">  
  <input type="text" id="name" v-model="name"><br/>  
  My name is {{name}}.  
</div>
```



```
var app = new Vue({
  el: '#app',
  data: {
    count: 0
  },
  methods: {
    doIt: function() {
      this.count++;
    }
  }
})
```

# EVENTS

v-on:click="something"

Defined in methods block of Vue instance

Easy to do modifiers (v-on:submit.prevent)

@click="something"

```
<div id="app">
  <button v-on:click="doIt">Do It</button><br/>
  You did it {{count}} times.
</div>
```

# ATTRIBUTE BINDING

`v-bind:attribute="value"`

`:attribute="value"`

`v-bind:style="{bgcolor:curColor, font-style:textStyle}"`

`v-bind:class="{highlighted:isActive, color:appStatus}"`

```
<div id="app">  
    
</div>
```

# COMPUTED PROPERTIES

Virtual properties

Property X is based on real data A and B

Defined in the `computed` part of the Vue instance

```
<div id="app">
  <h2>Male Cats</h2>
  <ul>
    <li v-for="cat in cats"
        v-if="cat.gender === 'Male'">{{cat.name}}</li>
  </ul>

  <h2>Female Cats</h2>
  <ul>
    <li v-for="cat in cats"
        v-if="cat.gender === 'Female'">{{cat.name}}</li>
  </ul>
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    cats: [
      { "name": "Alred", "gender": "Male" },
      { "name": "Barry", "gender": "Male" },
      { "name": "China", "gender": "Female" },
      { "name": "Dolores", "gender": "Female" },
      { "name": "Enya", "gender": "Female" },
      { "name": "Franke", "gender": "Male" },
      { "name": "Georgia", "gender": "Female" }
    ]
  }
})
```

EXAMPLE (BEFORE)

```
<div id="app">
  <h2>Male Cats</h2>
  <ul>
    <li v-for="cat in maleCats">{{cat.name}}</li>
  </ul>

  <h2>Female Cats</h2>
  <ul>
    <li v-for="cat in femaleCats">{{cat.name}}</li>
  </ul>
</div>
```

```
var app = new Vue({
  el: '#app',
  data: {
    cats: [
      { "name": "Alred", "gender": "Male" },
      { "name": "Barry", "gender": "Male" },
      { "name": "China", "gender": "Female" },
      { "name": "Dolores", "gender": "Female" },
      { "name": "Enya", "gender": "Female" },
      { "name": "Franke", "gender": "Male" },
      { "name": "Georgia", "gender": "Female" }
    ]
  },
  computed: {
    maleCats: function() {
      return this.cats.filter(c => c.gender === 'Male' );
    },
    femaleCats: function() {
      return this.cats.filter(c => c.gender === 'Female' );
    }
  }
})
```

# EXAMPLE (AFTER)

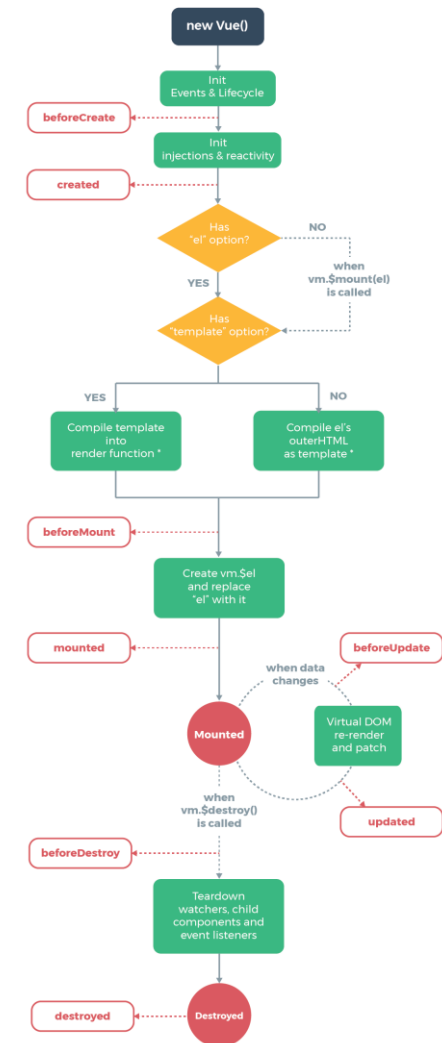


# VUE LIFECYCLE EVENTS

Events that occur while a Vue instance is doing stuff

beforeCreate, created, beforeMount, mounted, beforeUpdate, updated, activated, deactivated, beforeDestroy, destroyed, errorCaptured

<https://vuejs.org/v2/api/#Options-Lifecycle-Hooks>



\* template compilation is performed ahead-of-time if using a build step, e.g. single-file components

# EXAMPLE

```
var app = new Vue({
  el: '#app',
  data: {
    films: []
  },
  created: function() {
    fetch('https://swapi.co/api/films/')
      .then(res => res.json())
      .then(res => {
        this.films = res.results;
      })
  }
})
```

```
<div id="app">
  <ul>
    <li v-for="film in films">
      {{film.title}}
    </li>
  </ul>
</div>
```

# FILTERS

Allow to use a "formatter" to  
{{ birthdate | formatDate }}  
v something | changeValue  
Create: {{ birthdate | isNotOld | formatDate }}  
Age: {{ birthdate | age('show') }}



# QUICK TIP

Hiding the FOUC (Flash of Unstyled Content)

Add v-cloak to DOM item "holding" your app

Add a style declaration to hide it.

```
<style>  
[v-cloak] {display: none}  
</style>
```

```
<div id="app" v-cloak>
```

# FULL EXAMPLE: FORM VALIDATION

Do I need to explain this?

Check on submit, report on top

More examples: <https://vuejs.org/v2/cookbook/form-validation.html>

# FULL EXAMPLE: AJAX SEARCH

Accept user input

Send to remote API...

- <https://www.openbrewerydb.org/documentation/03-search>

Render responses



**BREAK**



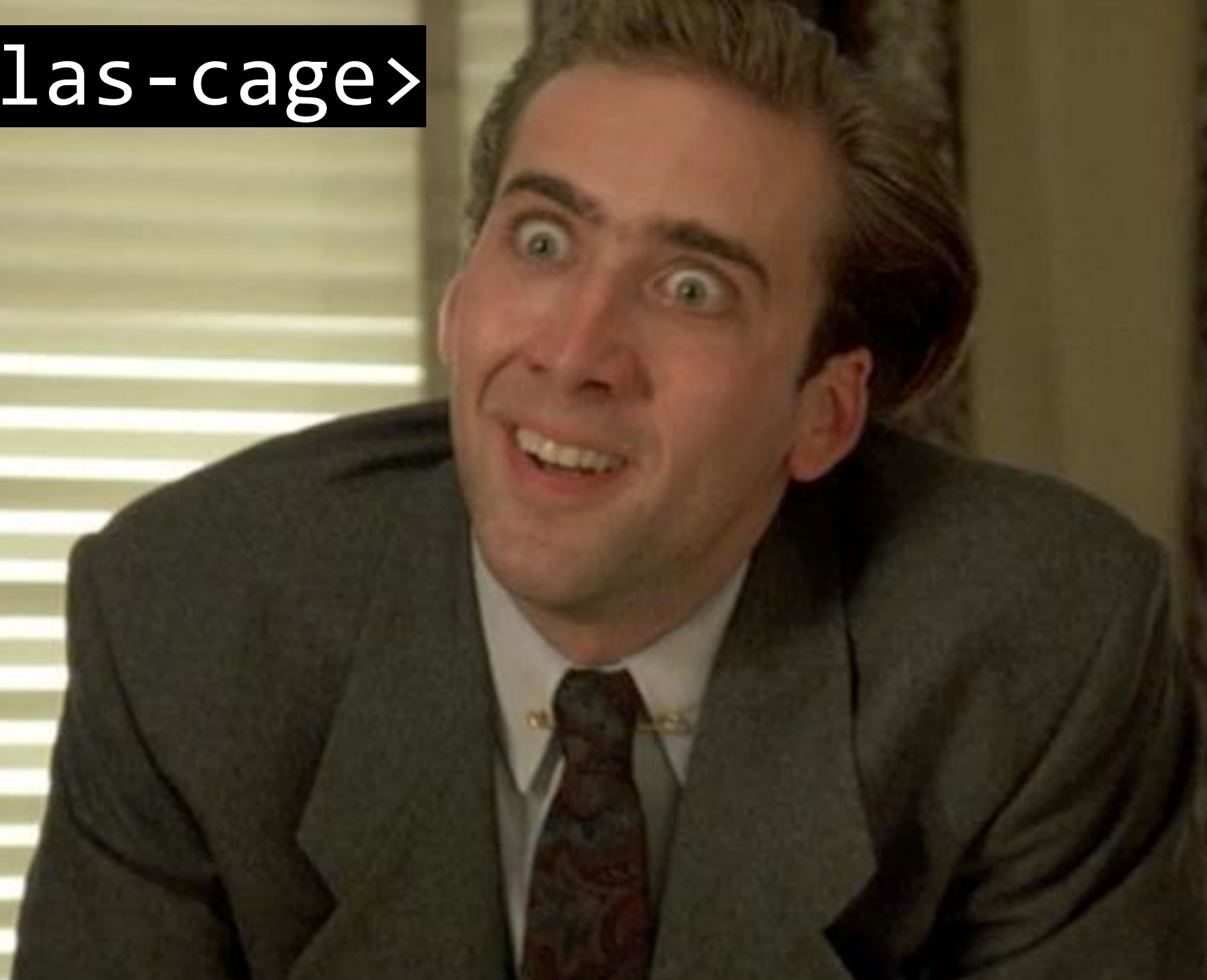
# COMPONENTS

Re-usable objects of awesome-ness

Template + Logic + Style = Component

Also solves one of HTML's most glaring problems...

<nicholas-cage>



# COMPONENT PARTS

Register the name

Contain “regular” Vue properties: data, computed, methods

Contains a template (the HTML rendered)

Possibly contains props (defining name, required, type, etc)

Can be registered globally or just to one Vue app/component

# EXAMPLE

---

```
Vue.component('nicolas-cage', {
  template: '',
  data() {
    return {
    }
  },
  computed: {
    url: function() {
      let theUrl = 'https://placecage.com/' + this.width + '/' + this.height;
      return theUrl;
    }
  },
  props: {
    height: {
      default: 400,
    },
    width: {
      default: 400
    }
  }
});
```

# EXAMPLE

```
<div id="app">
  <h2>Default</h2>
  <nicolas-cage></nicolas-cage>
  <p>
    <h2>H/W</h2>
    <nicolas-cage width="199" height="199"></nicolas-cage>
  </p>
</div>
```

# COMPONENT EXAMPLES

Google Static Maps: <https://www.npmjs.com/package/vue-static-map>

Bootstrap: <https://bootstrap-vue.js.org/>

Vuetify: <https://vuetifyjs.com/en/>

```
<b-tabs>
  <b-tab title="first" active>
    <br>I'm the first fading tab
  </b-tab>
  <b-tab title="second" >
    <br>I'm the second tab content
  </b-tab>
  <b-tab title="disabled" disabled>
    <br>Disabled tab!
  </b-tab>
</b-tabs>
```



# MORE COMPONENT STUFF

Data must be a function

You can pass data in from the parent

You can listen for events from the component (\$emit)

# EXAMPLE

---

```
Vue.component('kitten', {
  template:`
<div>
  <p>
    
  </p>
  <button @click="$emit('morecat')">MOAR CAT!</button>
</div>
`,
  props: {
    url: String
  }
})
```

```
<kitten @morecat="moreCatsPlease"
  v-for="cat in cats"
  :height="cat.height" :width="cat.width"
></kitten>
```

# MORE COMPONENT STUFF

Slots lets you pass information to components in a more free form manner.

In ways, the same as props, but you have more freedom.

```
<athing>
```

My component, "athing", has access to this text.

```
</athing>
```

...in my component template...

Your text was, <slot></slot>.

```
<mything>

  <template v-slot:thankyou>
    Hey, I want to <i>really</i> thank you for taking
    the time to do whatever. We here at Mega Corp truly
    care that you took the time.
  </template>

  <template #ending>
    My ending string for whatever.
  </template>

</mything>

...in my component template...

I'd like to thank you with <slot name="thankyou"></slot>
It was nice to see you,
let's end with <slot name="ending"></slot>
```

# FULL EXAMPLE: PLACEHOLDER

Create an `<image-placeholder>` tag

Use [placeholder.com](https://placeholder.com) (or any other service)

Accept height and width (with defaults)



**BREAK**



# PAGE ENHANCEMENT VERSUS APPS

Warning: Opinions Incoming...

What do we mean by "app"?

When do you use which?

APP TIME!

BestDemotivationalPo



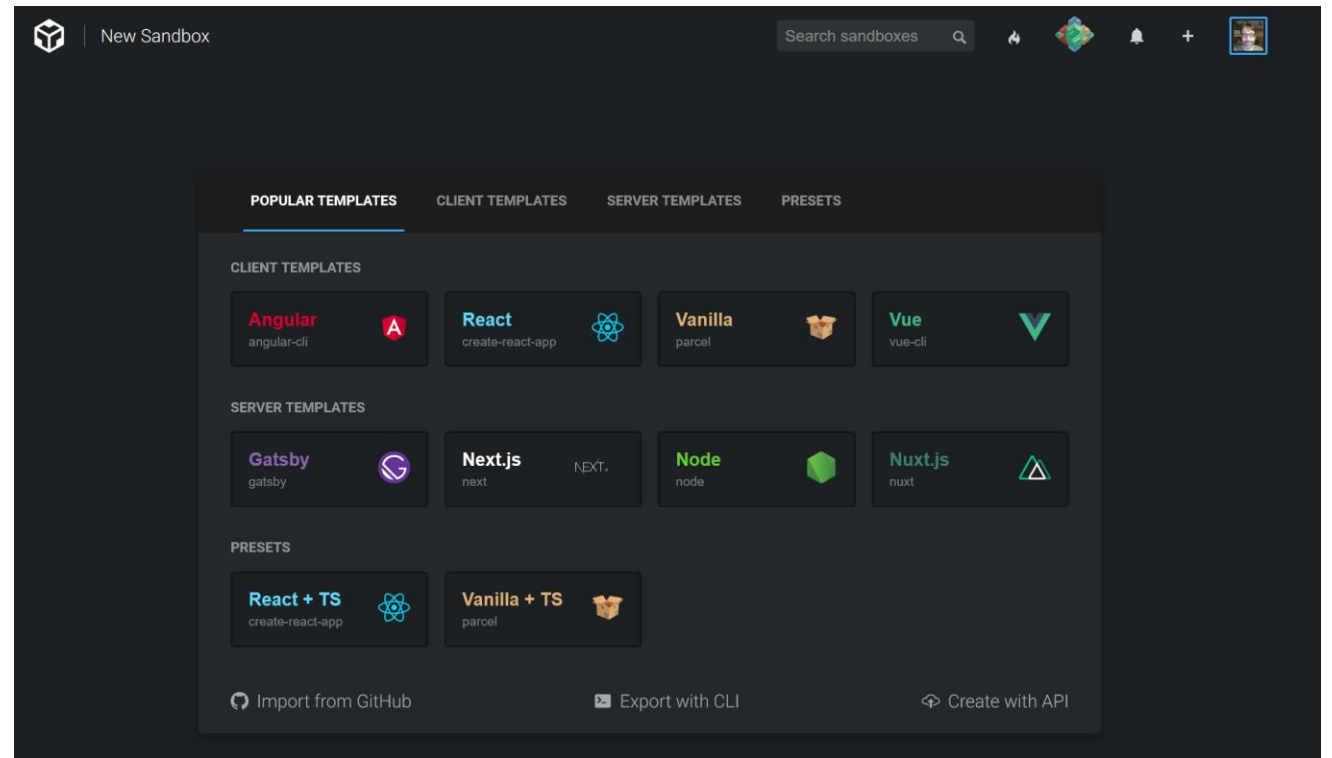
SERIOUS CAT

This is serious bussiness. Srsly.

# HOW?

Command Line tool

An online tool (CodeSandbox.io)



# THE CLI

Scaffolding and Building tool

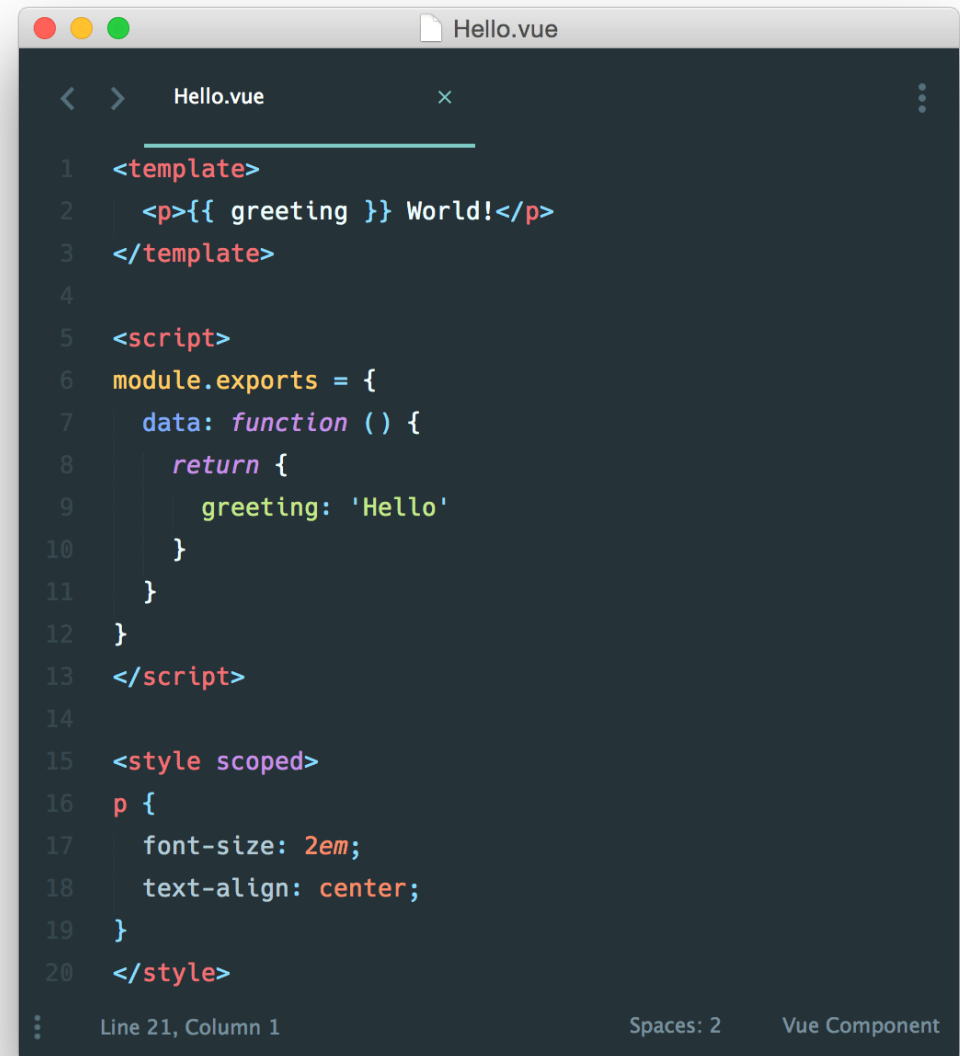
Focused on SPAs (but also does prototyping well)

<https://cli.vuejs.org>

GUI Tool

Supports Single File Components

# SINGLE FILE COMPONENTS



The image shows a code editor window titled "Hello.vue". The code is written in Vue.js Single File Component syntax and is as follows:

```
1 <template>
2   <p>{{ greeting }} World!</p>
3 </template>
4
5 <script>
6   module.exports = {
7     data: function () {
8       return {
9         greeting: 'Hello'
10      }
11    }
12  }
13 </script>
14
15 <style scoped>
16   p {
17     font-size: 2em;
18     text-align: center;
19   }
20 </style>
```

The editor interface includes a tab at the top labeled "Hello.vue", a line number gutter on the left, and a status bar at the bottom showing "Line 21, Column 1", "Spaces: 2", and "Vue Component".



# WALKTHROUGH

FYI, install via: `npm i -g @vue/cli`

- `npm i -g @vue/cli-service-global`

Then: `vue create foo`

Then: `npm run serve`

Then: `npm run build`

Then: Prototype example (note, no hot reload)

Then: UI example

# NOW WHAT?

Organization becomes a bit simpler

Optimized output

Easier to Build Up

# ROUTING

Not a part of Vue

Official library: vue-router (<https://router.vuejs.org>)

"If I'm at location /foo, I want to do Foo"

"If I'm at location /goo, I want to do Goo"

Installation via `<script>` tag or npm

# EXAMPLE (FROM THE DOCS)

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

const router = new VueRouter({
  routes // short for `routes: routes`
});

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

# EXAMPLE (STILL FROM THE DOCS)

```
<div id="app">
  <h1>Hello App!</h1>
  <p>
    <router-link to="/foo">Go to Foo</router-link>
    <router-link to="/bar">Go to Bar</router-link>
  </p>
  <router-view></router-view>
</div>
```

# FULL EXAMPLE: HOME AND ABOUT

First page – a home page

Second page – an about page

Use `<router-link>` and `<router-view>`

Use the CLI

# PARAMS AND DYNAMIC ROUTES

Routes can be dynamic

`/user/:id`

Dynamic portion available via `$route.params.X`

`/user/:id/post/:post_id` (stolen from docs)





# FULL EXAMPLE: MAIN/DETAIL

First page - list of data (based on an API, but can be anything)

Second page - detail

# MORE ROUTING FEATURES

## 404/Catch All:

- path: '\*'
- path: 'feature-\*

## Programmatic Navigation

- router.push('something')
- router.replace('something')
- router.go(X)

## HTML5 History mode

And more...

# STATE MANAGEMENT

Solves a problem you may not have!

One component, one set of data = easy

Two components, two set of data = mostly easy

Many components, one set of data = um....

# VUEX

(Another) library for Vue (deeply integrated)

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

"Store"

Components can read from the store

Components cannot change data in the store (directly)



# PARTS OF VUEX

State

Mutations

Getters

Actions

Modules

# INCLUDING VUEX

Examples taken from docs....

```
<script src="/path/to/vue.js"></script>  
<script src="/path/to/vuex.js"></script>
```

html

Or...

```
npm install vuex --save
```

sh

```
import Vue from 'vue'  
import Vuex from 'vuex'  
  
Vue.use(Vuex)
```

js

# CREATING A VUEX STORE

```
const store = new Vuex.Store({  
  
  state: {  
    hp: 20,  
    str: 15,  
    con: 16,  
    dex: 12,  
    int: 9,  
    wis: 11,  
    chr: 16,  
    type: 'fighter'  
  }  
  
});
```

# ADDING THE STORE

```
const app = new Vue({  
  el: '#app',  
  data: {  
  },  
  store: store  
});
```



# MUTATIONS

Change state

Called externally via: `store.commit('name of mutation');`

Or: `store.commit('name of mutation', someAdditionalValue);`

# EXAMPLE MUTATION

To call: `store.commit('hit');`

```
const store = new Vuex.Store({  
  
  state: {  
    hp: 20,  
    str: 15,  
    con: 16,  
    dex: 12,  
    int: 9,  
    wis: 11,  
    chr: 16,  
    type: 'fighter'  
  },  
  mutations: {  
    hit(state) {  
      state.hp--;  
    }  
  }  
});
```

# ACTIONS

Async

Chain to mutations

Called externally via `store.dispatch('name of action', optional additional data);`

If the distinction seems a bit weird... yes.

# EXAMPLE ACTION

```
const store = new Vuex.Store({  
  
  state: {  
    hp: 20,  
    str: 15,  
    con: 16,  
    dex: 12,  
    int: 9,  
    wis: 11,  
    chr: 16,  
    type: 'fighter'  
  },  
  
  mutations: {  
    heal(state, amt) {  
      state.hp+=amt;  
    },  
    hit(state) {  
      state.hp--;  
    }  
  },  
  
  actions: {  
    heal(context) {  
      setTimeout(() => {  
        context.commit('heal', 2);  
      }, 2000);  
    }  
  }  
});
```

# GETTERS

Pretty much just computed for Vuex

Values are cached based on dependencies

Accessed externally via `store.getters.nameOfGetter`

Can return a function so you can do: `store.getters.nameOfGetter(arg)`.

# EXAMPLE GETTER

```
const store = new Vuex.Store({  
  
  state: {  
    hp: 20,  
    str: 15,  
    con: 16,  
    dex: 12,  
    int: 9,  
    wis: 11,  
    chr: 16,  
    type: 'fighter'  
  },  
  
  getters: {  
    condition(state) {  
      if (state.hp < 2) return 'critical';  
      if (state.hp < 5) return 'near death';  
      if (state.hp < 10) return 'deeply wounded';  
      if (state.hp < 15) return 'wounded';  
      return 'just fine'  
    }  
  },  
});
```

# ONE MORE VUEX DEMO

## Stocks

IBM	\$87.74
Amazon	\$80.45
Microsoft	\$118.94
Disney	\$109.11

## Holdings

IBM	0
Amazon	0
Microsoft	0
Disney	0

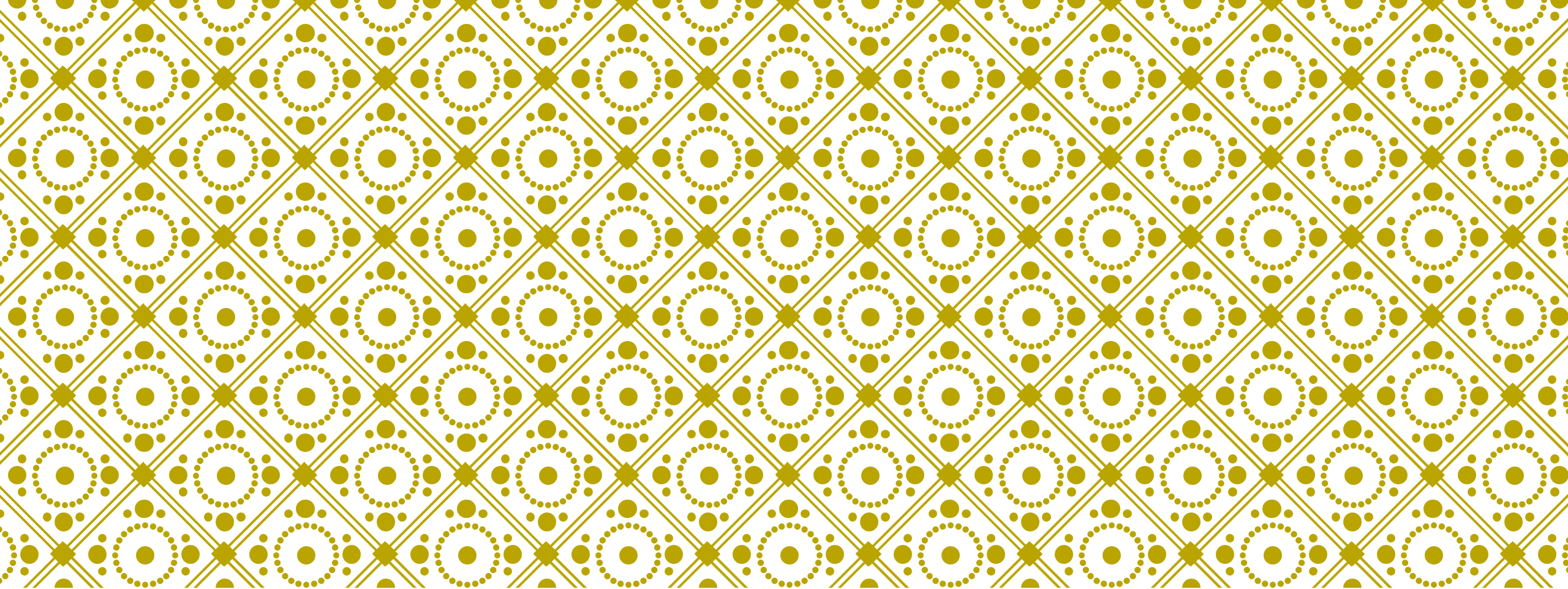
Total value of holdings: \$0.00

## Purchase and Sell Stocks

Buy  shares of  Purchase

Sell  shares of  Sell

You currently have \$1000.00 in cash.



## 3.0 - WHAT'S NEXT?



# VUE 3.0

## Faster, smaller

- Virtual DOM rewrite, native Proxy, 100% faster component instance initialization
- 20kb to roughly half (gzipped)

## Multiple v-model support

- `<MyThing v-model:rank="userRank" v-model:title="userTitle" />`

## Suspense

- Fallback support (ie, show this while loading)

## Portals

- Render content elsewhere, uses a `<Teleport>` tag

# VUE 3.0

## TypeScript Support

- Vue itself - all in TS

No need for "wrapper" divs in template blocks (aka "Fragments")

# VUE 3.0

temp.vue > template

```
1  <template>
2    <p>
3      Hello {{ name }}
4    </p>
5    <p>
6      How are you?
7    </p>
8  </template>
```

# VUE 3.0

```
▼ temp.vue > ...  
1  <template>  
2    <div>  
3      <p>  
4        Hello {{ name }}  
5      </p>  
6      <p>  
7        How are you?  
8      </p>  
9    </div>  
10 </template>
```

# VUE 3.0



# VUE 3.0

## TypeScript Support

- Vue itself - all in TS

No need for "wrapper" divs in template blocks (aka "Fragments")

Mounting changes to global libraries (add stuff to your Vue app, not global Vue object)

## Composition API

- OPTIONAL new way of writing components
- Better for larger, complex architectures

Official repo: <https://github.com/vuejs/vue-next>

# VUE 3 - RIP

Filters (deprecated)

render function (changed)

Event Bus (deprecated)



Photo by [Charles Deluvio](#) on [Unsplash](#)

# VUE 3.0 - LEARN MORE

Dan Vega on Vue 3 - <https://www.youtube.com/watch?v=HmdKqXP8JR8>

Upgrade guide - <https://dev.to/blacksonic/the-vue-3-upgrade-guide-4dc4>

Roundup - <https://madewithvuejs.com/blog/vue-3-roundup>



BUT WAIT —  
THERE'S MORE!



# EVEN MOAR!!!

Devtools Extension (Chrome, Firefox, Electron, Apple Keynote Browser)

- <https://github.com/vuejs/vue-devtools>

Sarah Drasner on CSS-Tricks: <https://css-tricks.com/intro-to-vue-1-rendering-directives-events/>

Vue docs has a cool cookbook: <https://vuejs.org/v2/cookbook/>

Newsletters:

- <https://news.vuejs.org>
- <https://vuejsdevelopers.com/newsletter/>

My Stuff: <https://www.raymondcamden.com/tags/vuejs>

# STUFF I DIDN'T COVER

Unit Testing (<https://vuejs.org/v2/guide/unit-testing.html>)

Transitions (<https://vuejs.org/v2/guide/transitions.html>)

Custom Directives (<https://vuejs.org/v2/guide/custom-directive.html>)

More....

ANY QUESTIONS?

