PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

LESSON 7: Introduction to Vue

HTML 300

OVERVIEW

1. Vue CLI
2. Interaction with DOM
3. Conditions
4. Exercise

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

VUE CLI

VUE CLI

- Vue CLI is a set up tool for Vue.
- You enter some commands and a template for Vue project is created.
- No need for Gulp.
- No need to organize files.

VUE CLI SETUP

- Make sure you have node installed on your computer.
- In you console type "`npm install -g @vue/cli`"
  - This will install VUE CLI globally on your computer.
- If that does not work, try "npm install –g vue-cli"
- Once you are done type "vue --version"
  - You should see a number that tells you your current version of Vue.
- Then type "vue init webpack-simple vue-cli"
  - This instalizes a vue project with a simple webpack and you are calling the folder vue-cli

VUE CLI TROUBLE SHOOTING

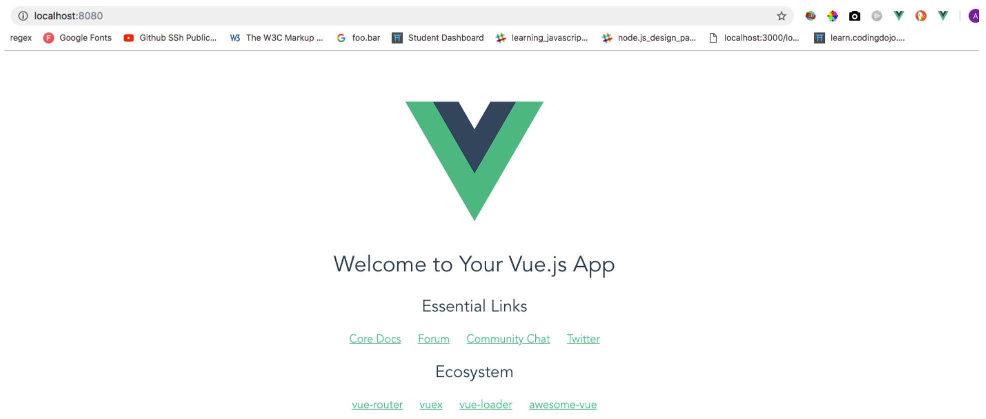
- If you are getting an error with "Permission Denied" or "cb() never called", do the following.
- In you console type "sudo npm cache verify"
- Then type "sudo npm cache verify"
- Then "node -v" to verify your version of node.
- Then "npm -v" to verify your versin
- Then try "npm install -g @vue/cli-init"

VUE CLI CONTINUES

- For project name, description, and author just press enter.
- When it asks for Sass, type yes.
- CD into you project.
- Type "sudo npm install".
- Then run "npm run dev".

VUE CLI

- And you should see this.



Welcome to Your Vue.js App

Essential Links

Core Docs Forum Community Chat Twitter

Ecosystem

vue-router vuex vue-loader awesome-vue

VUE CLI

- Take a couple of minutes to explore the file structure.
- If you are having problems, delete node from your computer and reinstall it. Then try following the directions from the previous slides.
- If you are still having trouble, please let me know.

VUE CLI EXPLANATION

Some of the files

- .babelrc allows us to write ES6 code and translates it to ES5.
- index.html is our main file, but where is all our html code?
- Webpack builds our code.
- In our src folder is where the magic happens.
  - In index.html, the id of App refers to this App.vue file. This is where all our code is.
  - Main.js holds the JavaScript that we need for Vue.
  - The assets folder holds images.

APP.VUE

- This is a template that you are going to use in Vue.
- It is broken down into three sections.
    - The template is where your html goes.
    - The script if for your JavaScript
    - And the Style is for you CSS.

VUE DEBUGGING

Go to https://github.com/vuejs/vue-devtools.

- Choose the Chrome extension.
- You should notice a V in your top tool bar.
- This will come in handy later when you are debugging Vue code.

Main.js

```
new Vue({
  el: '#app',
  render: h => h(App)
})
```

This new Vue creates a view object and lets the browser know that you are using Vue.

The "el" property lets us know which id we use to control our Vue code.

APP.VUE

```
<script>
export default {
  name: 'app',
  data () {
    return {
      msg: 'Welcome to Your Vue.js App'
    }
  }
}
</script>
```

If we look in th script tag we see data(). This is where we are going to add variable. So instead of hard coding text in our HTML, we can create a variable that will change based on conditional statements we will cover later.

APP.VUE

```
<template>
  <div id="app">
    <img src="./assets/logo.png">
    <h1>{{ msg }}</h1>
  </div>
</template>
```

In our template we see the div with an id of "app" and an h1 tag with {{msg}}. This is how you code for variable in Vue.

APP.VUE

Delete everything in the template after the h1 tag.

Then add an input.

```
<template>
  <div id="app">
    <img src="./assets/logo.png">
    <h1>{{ msg }}</h1>
      <input type="text" v-on:input="changeTitle">
  </div>
</template>
```

What is this v-on:input="changeTitle"?

APP.VUE METHODS

V-on is a Vue directive that tells the browser that when there is a change in the input to go to the method changeTitle.

```
<template>
  <div id="app">
    <img src="./assets/logo.png">
    <h1>{{ msg }}</h1>
      <input type="text" v-on:input="changeTitle">
  </div>
</template>
```

APP.VUE METHOD

Now we need to add our method. In the script section under data we add.

```
data () {
  return {
    msg: 'Welcome to Your Vue.js App'
  }
},
methods:{
  changeTitle: function(event){
    this.msg = event.target.value;
  }
}
```

APP.VUE METHOD

> We have a section that is called methods. This is where we put our methods.

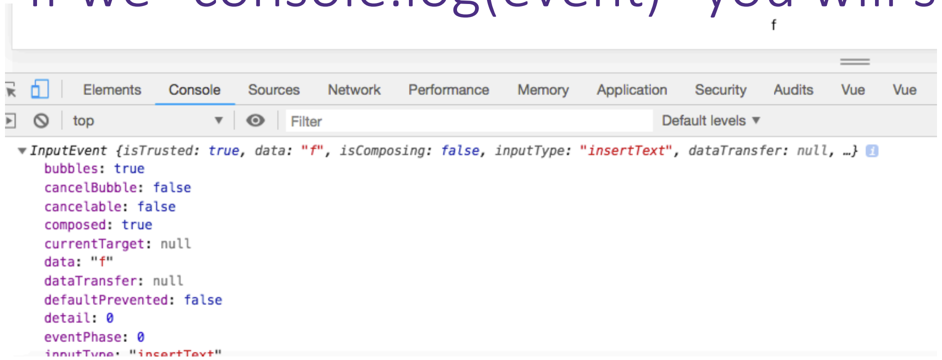
> In the function, we have a variable called event. This is the information being sent in through the binding with the input field.

> In Vue when in the script section you will preface all variable with the "this" keyword.

APP.VUE METHOD

So how do we know to use 'event.target.value' to get the information from the input field.

If we "console.log(event)" you will see this:

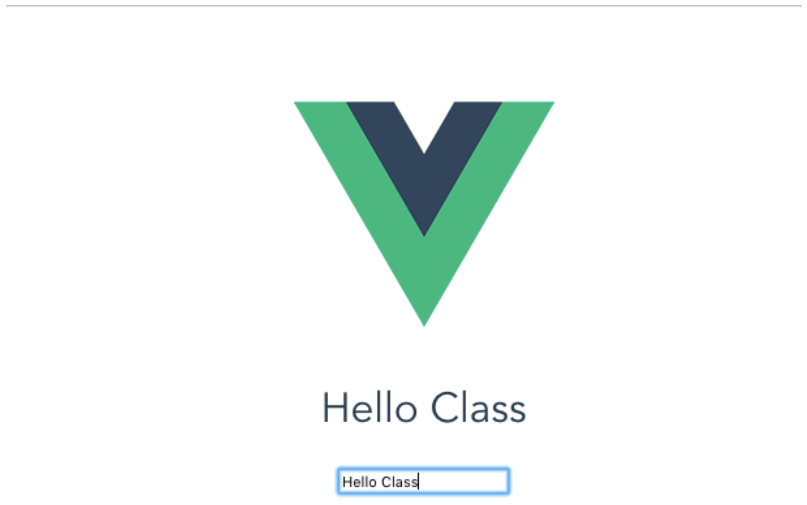


```
▼InputEvent {isTrusted: true, data: "f", isComposing: false, inputType: "insertText", dataTransfer: null, …}
   bubbles: true
   cancelBubble: false
   cancelable: false
   composed: true
   currentTarget: null
   data: "f"
   dataTransfer: null
   defaultPrevented: false
   detail: 0
   eventPhase: 0
   inputType: "insertText"
```

Eventually you scroll down to "target" and find "value"

APP.VUE METHOD

You result should look like this.



Hello Class

Hello Class

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

INTERACTION WITH DOM

V-BIND

What if we want to add some of our Vue variables inside of our HTML elements?

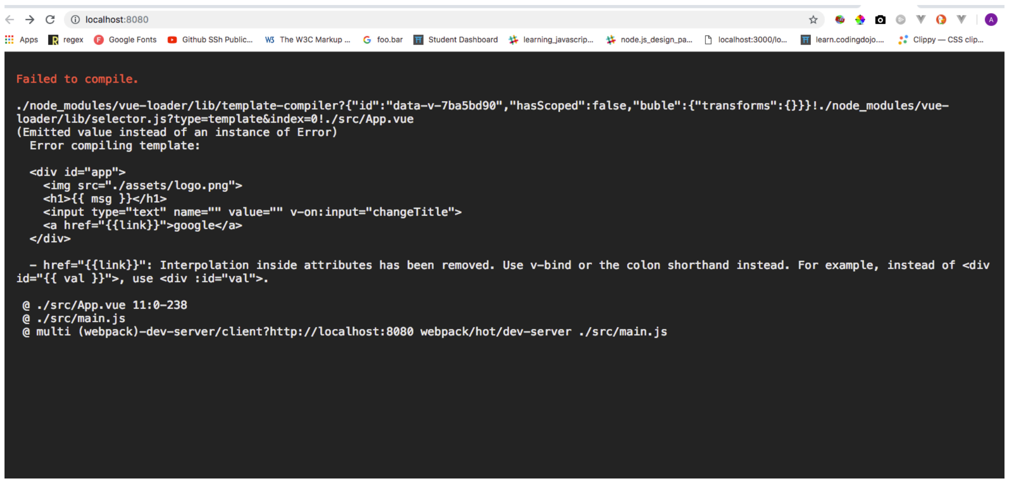
In our html we are going to add an "a" tag with the href being the variable link.

```
<a href="{{link}}">google</a>

data () {
    return {
      msg: 'Welcome to Your Vue.js App',
      link: 'https://www.google.com/'
    }
  },
```

V-BIND

Well this is the result.



```
Failed to compile.

./node_modules/vue-loader/lib/template-compiler?{"id":"data-v-7ba5bd90","hasScoped":false,"buble":{"transforms":{}}}!./node_modules/vue-
loader/lib/selector.js?type=template&index=0!./src/App.vue
(Emitted value instead of an instance of Error)
  Error compiling template:

  <div id="app">
    <img src="./assets/logo.png">
    <h1>{{ msg }}</h1>
    <input type="text" name="" value="" v-on:input="changeTitle">
    <a href="{{link}}">google</a>
  </div>

  - href="{{link}}": Interpolation inside attributes has been removed. Use v-bind or the colon shorthand instead. For example, instead of <div
id="{{ val }}">, use <div :id="val">.

 @ ./src/App.vue 11:0-238
 @ ./src/main.js
 @ multi (webpack)-dev-server/client?http://localhost:8080 webpack/hot/dev-server ./src/main.js
```

The HTML has no idea what "{{link}}" is

V-BIND

We add v-bind: to the beginning of the HTML attribute. This changes the attribute to allow Vue variable.

```
<a v-bind:href="link" target="_blank">google</a>
```

Now Test it out!

Notice "link" is not surrounded by "{{}}"

The Target blank is so that google open on another tab.

V-ON

V-on is a listener that you can attach to an HTML element.

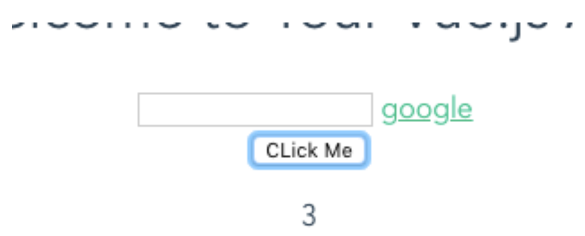
In this example we are going to use click as an event listener, but there are others DOM event syou use such a keyup, keydown, and mouseenter

```
<button v-on:click = "increase">Click Me</button>
    <p>{{counter}}</p>

  data () {
    return {
      counter: 0
    }
  },
  methods:{
    increase: function(){
      this.counter++;
    }
  }
```

V-ON

So when you click on the button the counter will increase by one.



TWO-WAY DATA BINDING

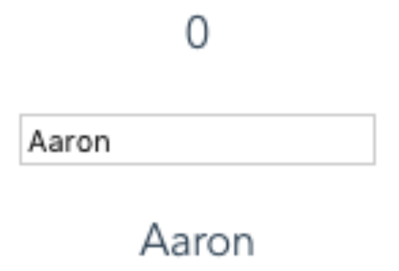
Two-way data binding is when you have an event that sends and listens for messages.

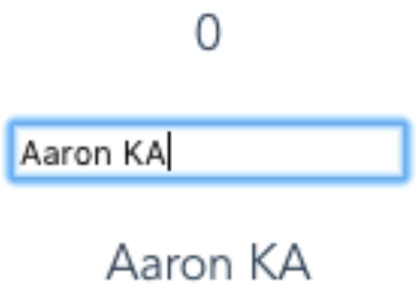
```
  <template>
   <div id="app">
     <input type="text" v-model="name">
     <p>{{name}}</p>
   </div>
</template>

   data () {
     return {
       name: "Aaron"
     }
   },
```

So any changes made in the input box will be reflected in the p tag.

TWO-WAY DATA BINDING





DYNAMIC STYLING WITH CSS

Just like in jQuery we can change the CSS dynamically.

Here we are going to change the color of a box when we click on it.

DYNAMIC STYLING WITH CSS

```
<template>
  <div id="app">
        <div class="demo"
    v-on:click="attachRed = !attachRed"
    v-bind:class="{red:attachRed}">

    </div>
  </div>
</template>
```

We create a div with a class of demo. Then we create an onclick event that changes toggles a variable called attachRed.

Finally we bind the class red with the variable red.

DYNAMIC STYLING WITH CSS

```
.demo{
  height: 100px;
  width: 100px;
  background-color: green;
}

.red{
  background-color: red;
}
```

In our CSS we create a box with a green background and a class called red with a red background.

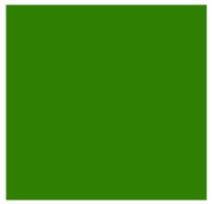
DYNAMIC STYLING WITH CSS

```
data () {
  return {
    attachRed: false,
  }
},
```

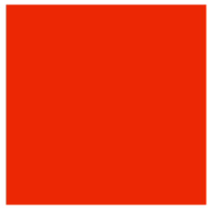
Finally in our JavaScript we set attachRed to false.

DYNAMIC STYLING WITH CSS

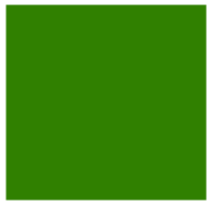
On Load



After Click



One More Time



PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

CONDITIONS

V-IF

The v-if is an if statement that you can add in your html.

```
<template>
  <div id="app">
    <p v-if="show">Can you see me</p>
    <button v-on:click="show = !show"></button>
  </div>
</template>

data () {
    return {
      show: true,
    }
  },
```

When we click on the button the p tag will toggle.

V-FOR

The v-for is used to iteration though an array.

```
<template>
  <div id="app">
    <ul>
      <li v-for="ingredient in ingredients">
{{ingredient}}</li>
    </ul>
  </div>
</template>

  data () {
    return {
      ingredients: ['meat', 'fruit', 'cookies'],
    }
  },
```

V-FOR

The v-for can also be used to iteration though an object.

```
<template>
  <div id="app">
    <ul>
        <li v-for="person in persons">
        {{person.name}}
        {{person.age}}
      </li>
    </ul>
  </div>
</template>

  data () {
    return {
        persons:[
          {name: 'Aaron', age: 40, color: 'red'},
          {name: 'Max', age: 27, color: 'blue'}
      ],
    }
  },
```

e

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

EXCERCISE

EXERCISE

EARLIER, YOU CREATED A FORM USING BOOTSTRAP 4 AND JQUERY WITH THESE CHARACTERISTICS:

- The form has two buttons.
- One that has "click me" and other has "submit".
- The submit button should be greyed out and unclickable (the form should not submit).
- When you hover over the submit button a tooltip should appear letting the user know to click the other button.
- Once the user clicks on the other button, the submit button should no longer be greyed out, clickable (form will submit), and tooltip should not show when you hover over it.

NOW DO THE SAME WITH VUE CLI.

QUESTIONS

html3@uw.edu

As always feel free to contact the instructional team if you have any questions. They do have a full-time jobs, so they might not get back to you immediately.

If you do not hear back from them in 48 hours, please try again.