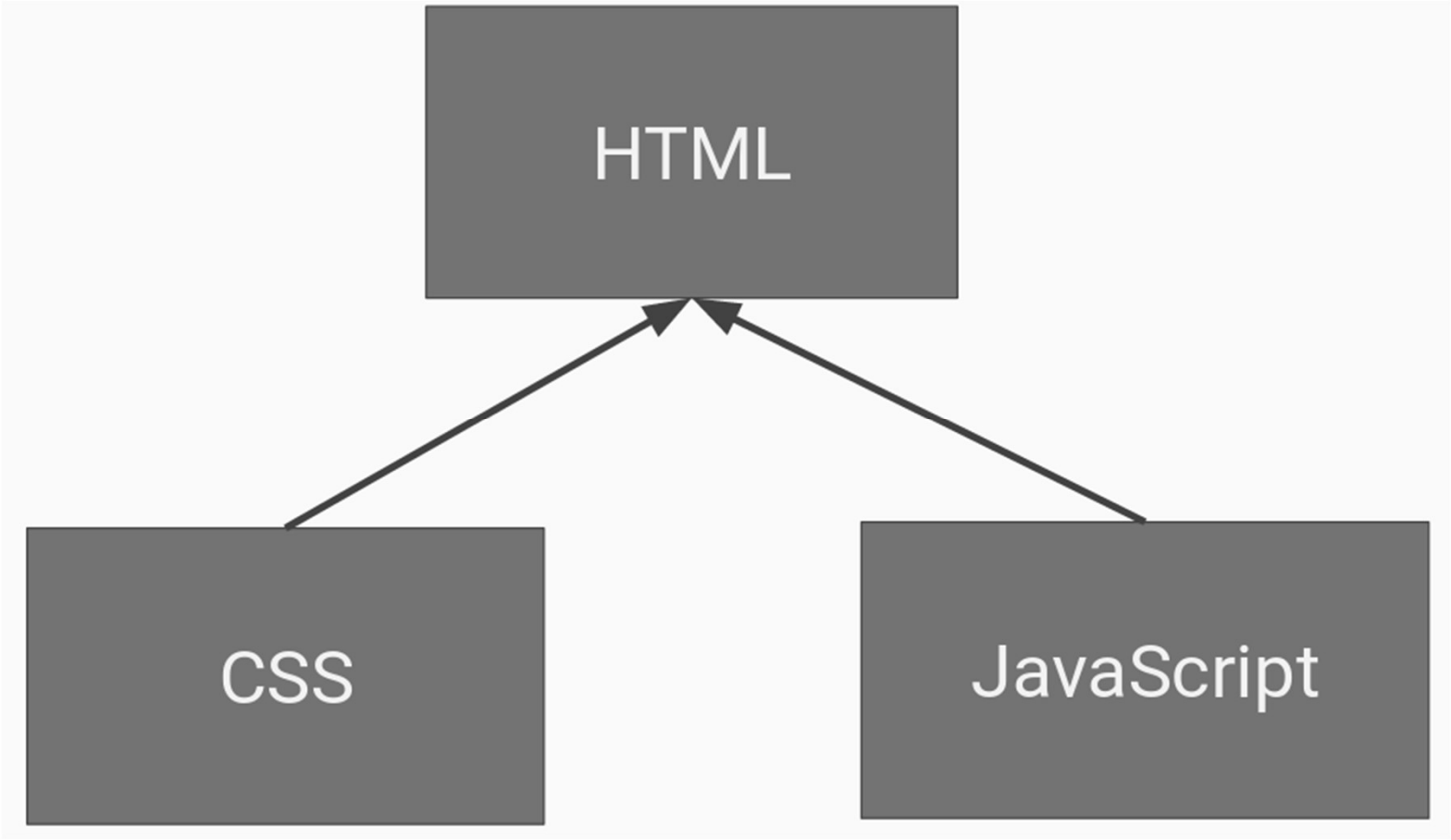
MODULE 3

Introduction to Vue.js and Data Binding

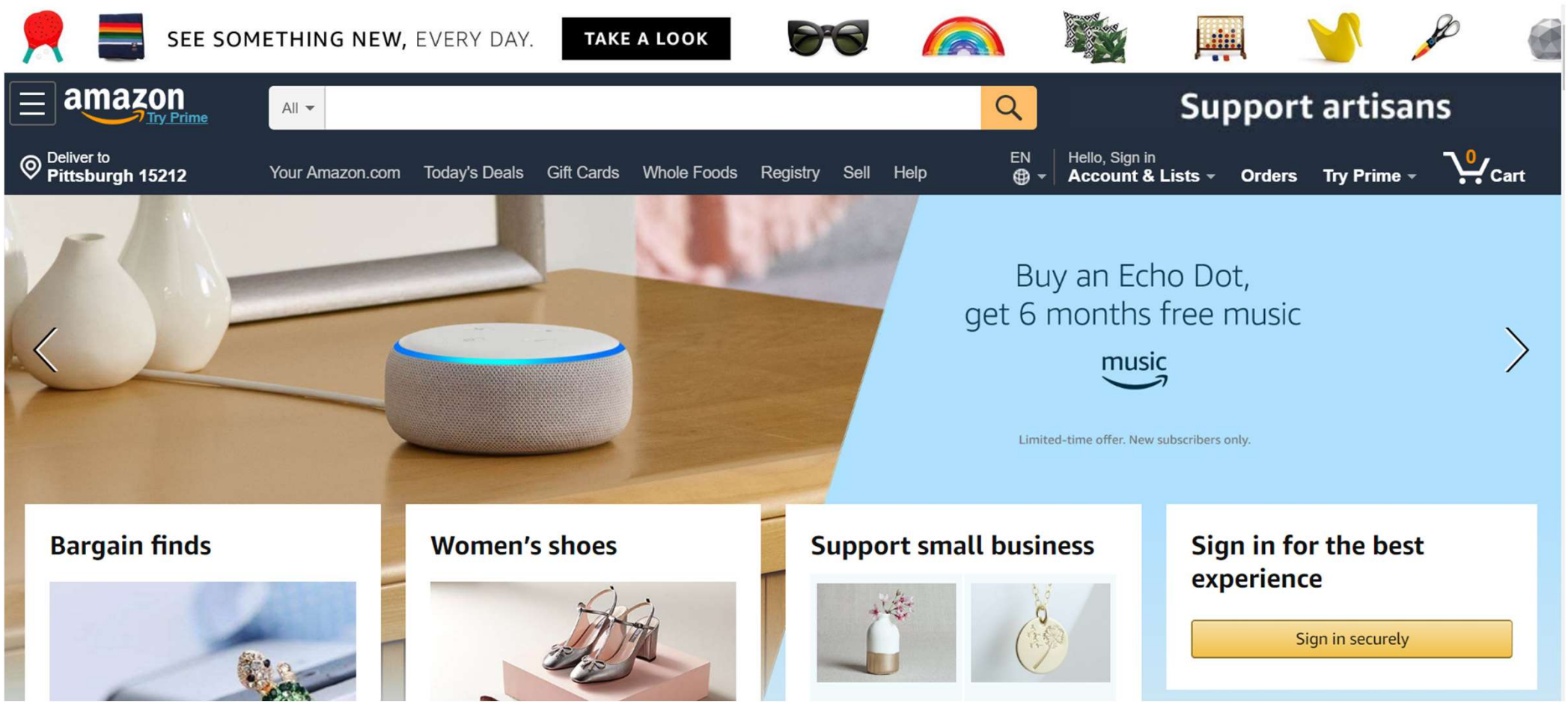
Vanilla JavaScript



Modern Web Sites

- Incredibly complex
- JavaScript files get larger and larger
- More and more difficult to maintain.

Parts of a Web Page



Building Applications: Two Methods

- MVC
  - Front End
    - CLI
  - Back End
    - Controllers
    - Models

Processing of data and requests happens on the server side. Only minor user interaction occurs on the client.

- JS and API
  - Front End
    - HTML
    - JavaScript
  - Back End
    - Controllers
    - Models

Processing of requests happens on the server side. Data manipulation and processing happens on the client side.

Building Applications: Three Methods

- MVC
  - Front End
    - Views
  - Back End
    - Controllers
    - Models

Processing of data and requests happens on the server side. Only minor user interaction occurs on the client.

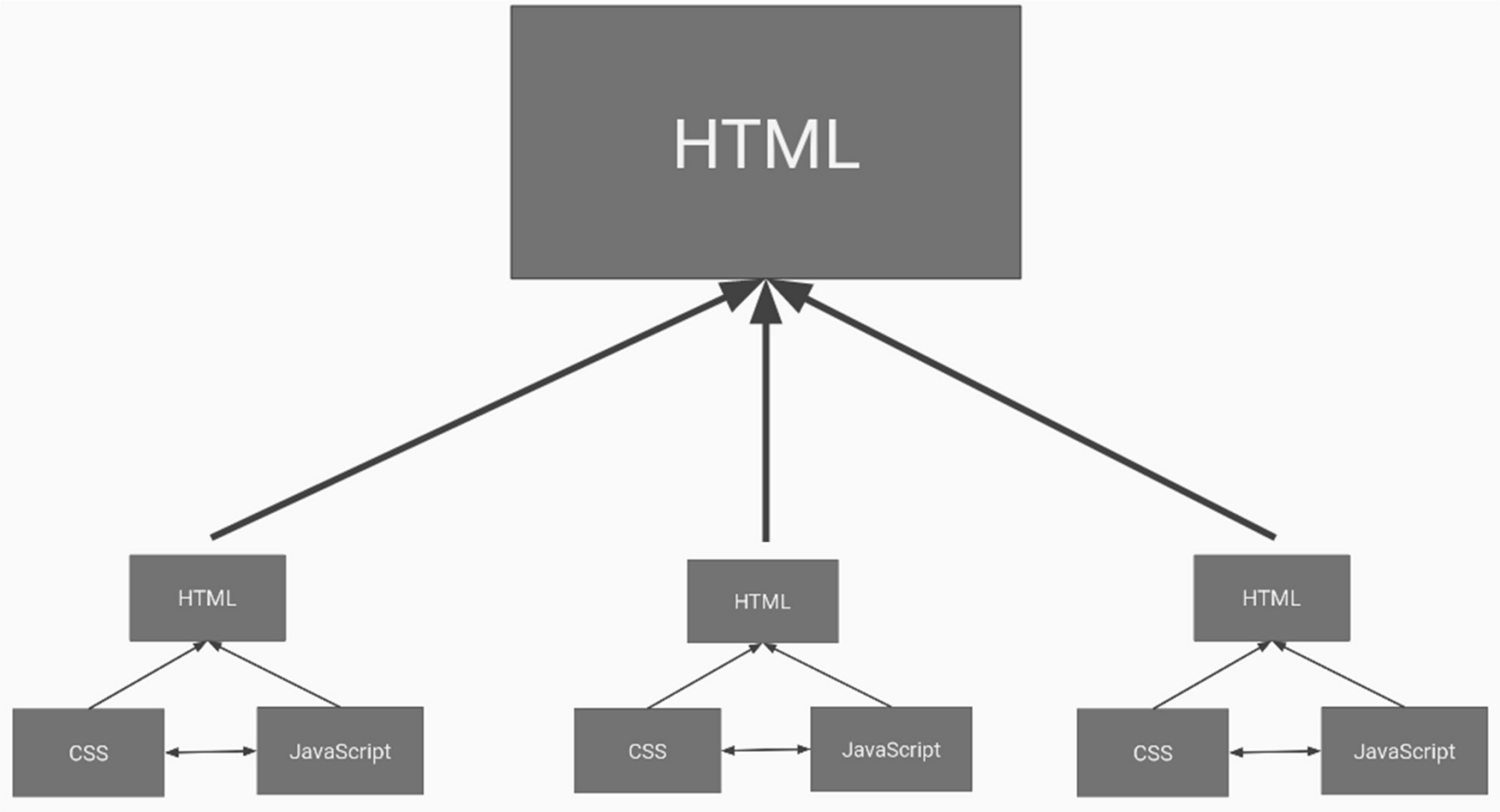
- JS and API
  - Front End
    - HTML
    - JavaScript
  - Back End
    - Controllers
    - Models

Processing of requests happens on the server side. Data manipulation and processing happens on the client side.

- Vue.js and API
  - Front End
    - Vue.js
  - Back End
    - Controllers
    - Models

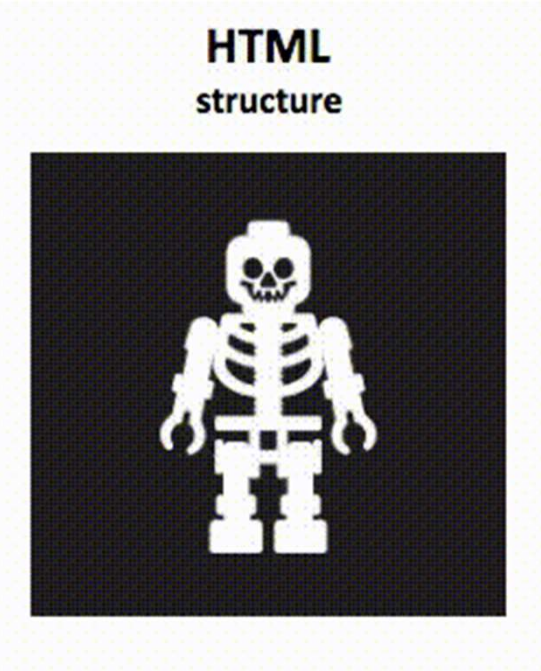
Processing of requests happens on the server side. Data manipulation and processing happens on the client side.

Component JavaScript



Parts of a Component

```
<template>
  <div id="app">
    <img alt="Vue logo" src="./assets/logo.png">
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>
```



Parts of a Component

```
<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>
```

Parts of a Component

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

export default {
  name: 'app',
  components: {
    HelloWorld
  }
}
</script>
```

Loading a Component

- In your <script> tag:

```
import HelloWorld from './components/HelloWorld.vue’
```

- Then add your component:

```
components: {
    HelloWorld
}
```

- Finally, Add to the page

```
<hello-world></hello-world>
```

LET’S CODE!

Old Terms, New World

- Encapsulation
- Data Binding
- Derived Properties

One Way Data Binding

```
<h1>{{message}}</h1>
```

Data from the data property is bound to the view

```
export default {
    name: "user-management",
    data() {
        return {
            message: ‘Hello World’
        }
    }
}
```

Two Way Data Binding

```
<input type="text" v-model="firstName"/>
```

What happens when we type into the input box?

```
export default {
  name: "user-management",
  data() {
    return {
      firstName: '',
      lastName: '',
      gender: '',
      disabled: false,
      permissions: [],
      role: '',
      notes: ''
    }
  }
}
```

v-model on HTML elements

- Radio Buttons
  - <input type="radio" name="gender" value="m" v-model="gender">
  - <input type="radio" name="gender" value="f" v-model="gender">
- Selects
  - <select name="role" v-model="role">
- Single Checkbox
  - <input type="checkbox" name="disabled" id="disabled" v-model="disabled">
- Multiple Checkboxes
  - <input type="checkbox" name="permissions" value="w" v-model="permissions">
  - <input type="checkbox" name="permissions" value="x" v-model="permissions">
  - <input type="checkbox" name="permissions" value="d" v-model="permissions">
- Textareas
  - <textarea name="notes" cols="60" rows="10" v-model="notes"></textarea>

v-model Modifiers

- .lazy
  - Data property updated ***after*** the user leaves the element
  - <textarea name="notes" cols="60" rows="10" v-model.lazy="notes"></textarea>
- .number
  - Save the value as a number instead of string.
  - Useful for data that needs calculations
- .trim
  - Removes white space from beginning and end of string.

Vue Conditionals

- v-if = <Boolean expression>

```
<div v-if="isFriday">
    <span>Happy Friday</span>
</div>
```

- v-show = <Boolean expression>

```
<div v-show="isFriday">
    <span>Happy Friday</span>
</div>
```

It’s EVERYWHERE!!

- Looping in Vue.js
  - v-for directive to render a list of items based on an array

```
<ul id="example-1">
  <li v-for="item in items">
    {{ item.message }}
  </li>
</ul>
```

```
data() {
    return items: [
      { message: 'Foo' },
      { message: 'Bar' }
    ]
  }
```

- Foo
- Bar

v-for

```
creditCardType : '',
creditLogoSrc: '../assets/credit.png',
availableCardTypes: {
   'visa': 'Visa',
   'mc': 'MasterCard',
   'dc': 'Discover Card'
}
```

```
<select name="creditCardType" id="creditCardType" class="creditCardType" v-model="creditCardType">
   <option value="" disabled>-- Select One --</option>
   <option v-for="(creditCardName, creditCardAbbrev) in availableCardTypes"
      v-bind:value="creditCardAbbrev" v-bind:key="creditCardAbbrev">{{ creditCardName }}</option>
</select>
```

v-for

```
creditCardType: '',
creditLogoSrc: '../assets/credit.png',
availableCardTypes: {
   'visa': 'Visa',
   'mc': 'MasterCard',
   'dc': 'Discover Card'
}
```

```
<select name="creditCardType" id="creditCardType" class="creditCardType" v-model="creditCardType">
   <option value="" disabled>-- Select One --</option>
   <option v-for="(creditCardName, creditCardAbbrev) in availableCardTypes"
      v-bind:value="creditCardAbbrev" v-bind:key="creditCardAbbrev">{{ creditCardName }}</option>
</select>
```

v-for

```
creditCardType: '',
creditLogoSrc: '../assets/credit.png',
availableCardTypes: {
   'visa': 'Visa',
   'mc': 'MasterCard',
   'dc': 'Discover Card'
}
```

```
<select name="creditCardType" id="creditCardType" class="creditCardType" v-model="creditCardType">
   <option value="" disabled>-- Select One --</option>
   <option v-for="(creditCardName, creditCardAbbrev) in availableCardTypes"
      v-bind:value="creditCardAbbrev" v-bind:key="creditCardAbbrev">{{ creditCardName }}</option>
</select>
```

v-for

```
creditCardType: '',
creditLogoSrc: '../assets/credit.png',
availableCardTypes: {
   'visa': 'Visa',
   'mc': 'MasterCard',
   'dc': 'Discover Card'
}
```

```
<select name="creditCardType" id="creditCardType" class="creditCardType" v-model="creditCardType">
   <option value="" disabled>-- Select One --</option>
   <option v-for="(creditCardName, creditCardAbbrev) in availableCardTypes"
      v-bind:value="creditCardAbbrev" v-bind:key="creditCardAbbrev">{{ creditCardName }}</option>
</select>
```

v-for

```
creditCardType: '',
creditLogoSrc: '../assets/credit.png',
availableCardTypes: {
   'visa': 'Visa',
   'mc': 'MasterCard',
   'dc': 'Discover Card'
}
```

```
<select name="creditCardType" id="creditCardType" class="creditCardType" v-model="creditCardType">
   <option value="" disabled>-- Select One --</option>
   <option v-for="(creditCardName, creditCardAbbrev) in availableCardTypes"
      v-bind:value="creditCardAbbrev" v-bind:key="creditCardAbbrev">{{ creditCardName }}</option>
</select>
```

v-bind

- Toggle classes based on Boolean
  - <input type="text" v-model="firstName” **v-bind:class="{ 'needs-content': firstName == '' }"**>
- Change images
  - <img class="creditLogo" v-bind:src="creditCardLogoSrc">
- Disable elements
  - <button type="submit" v-bind:disabled="creditCardNumber == ''">Send Order</button>

- Inject HTML (kind of)
  -

Computed Properties

- Remember derived properties?

```
computed: {
    shippingStates(vm) {
        return vm.availableStates.filter( (state) => {
            return state.canShip;
        });
    }
}
```

```
<option v-for="stateObject in shippingStates“ v-bind:value="stateObject.abbreviation" v-bind:key="stateObject.abbreviation">{{stateObject.name}}</option>
```

LET’S CODE!

WHAT QUESTIONS DO YOU HAVE?

Reading for tonight:

Vue Event Handling