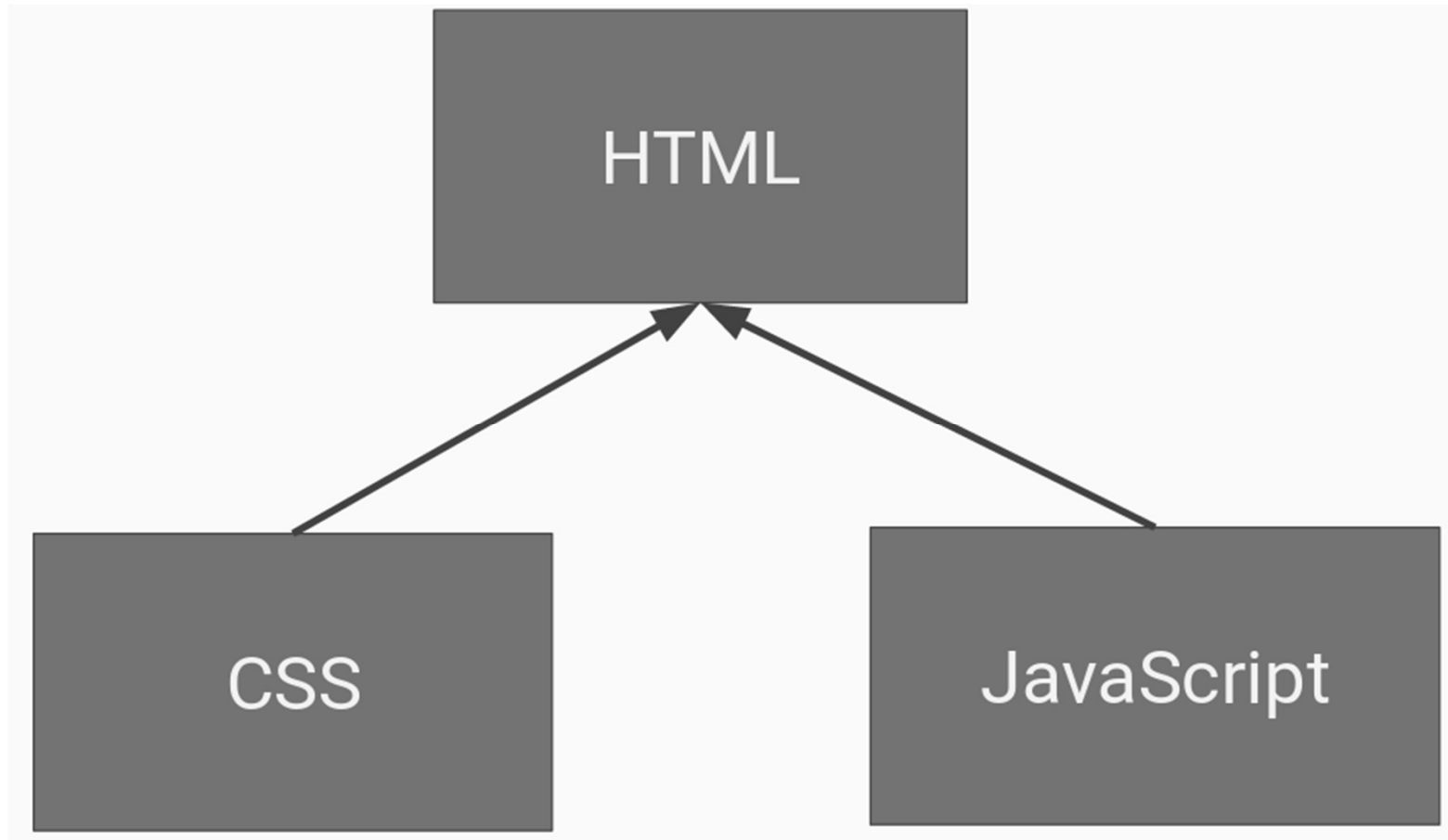


MODULE 3

Introduction to Vue.js and Data Binding



Vanilla JavaScript

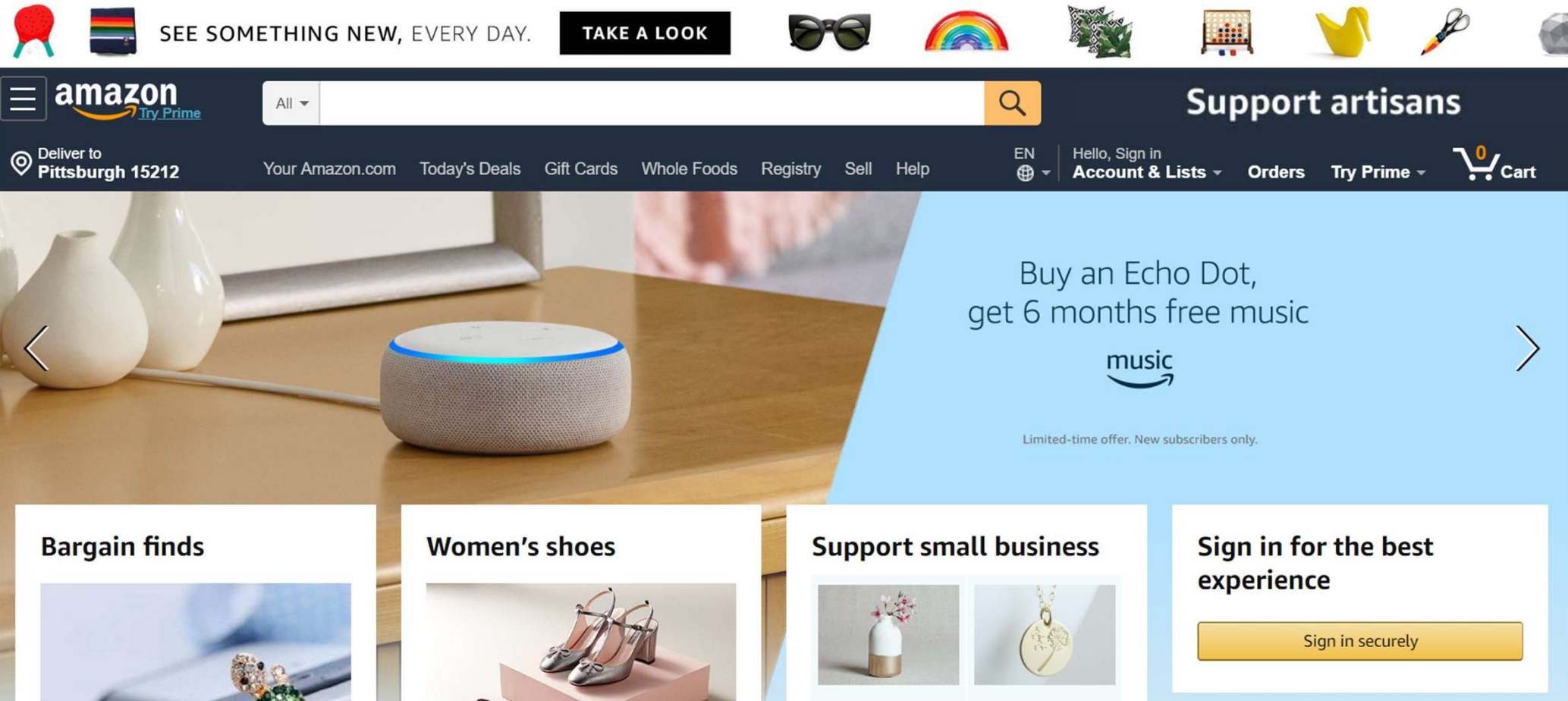


Modern Web Sites

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



Parts of a Web Page



The image is a screenshot of the Amazon homepage, illustrating various components of a web page. At the top, there is a navigation bar with a header area containing a red location pin icon, a rainbow flag icon, the text "SEE SOMETHING NEW, EVERY DAY.", a black "TAKE A LOOK" button, and a row of decorative icons including sunglasses, a rainbow, a potted plant, a chessboard, a yellow duck, and a pair of scissors. Below this is a dark blue navigation bar featuring the Amazon logo with "Try Prime" text, a search bar with a dropdown menu set to "All", a magnifying glass icon, the text "Support artisans", and a row of links: "Deliver to Pittsburgh 15212", "Your Amazon.com", "Today's Deals", "Gift Cards", "Whole Foods", "Registry", "Sell", "Help", "EN", "Hello, Sign in Account & Lists", "Orders", "Try Prime", and a shopping cart icon with a "0" and the word "Cart". The main content area features a large banner for an Echo Dot promotion, showing a white Echo Dot on a wooden table with a blue light ring. The banner text reads "Buy an Echo Dot, get 6 months free music" with the Amazon Music logo and a "Limited-time offer. New subscribers only." note. Below the banner are four promotional boxes: "Bargain finds" with a jewelry image, "Women's shoes" with a pair of high-heeled shoes, "Support small business" with two small product images, and "Sign in for the best experience" with a yellow "Sign in securely" button.

SEE SOMETHING NEW, EVERY DAY. TAKE A LOOK

amazon Try Prime

All

Support artisans

Deliver to Pittsburgh 15212 Your Amazon.com Today's Deals Gift Cards Whole Foods Registry Sell Help EN Hello, Sign in Account & Lists Orders Try Prime Cart

Buy an Echo Dot, get 6 months free music

music

Limited-time offer. New subscribers only.

Bargain finds

Women's shoes

Support small business

Sign in for the best experience

Sign in securely

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

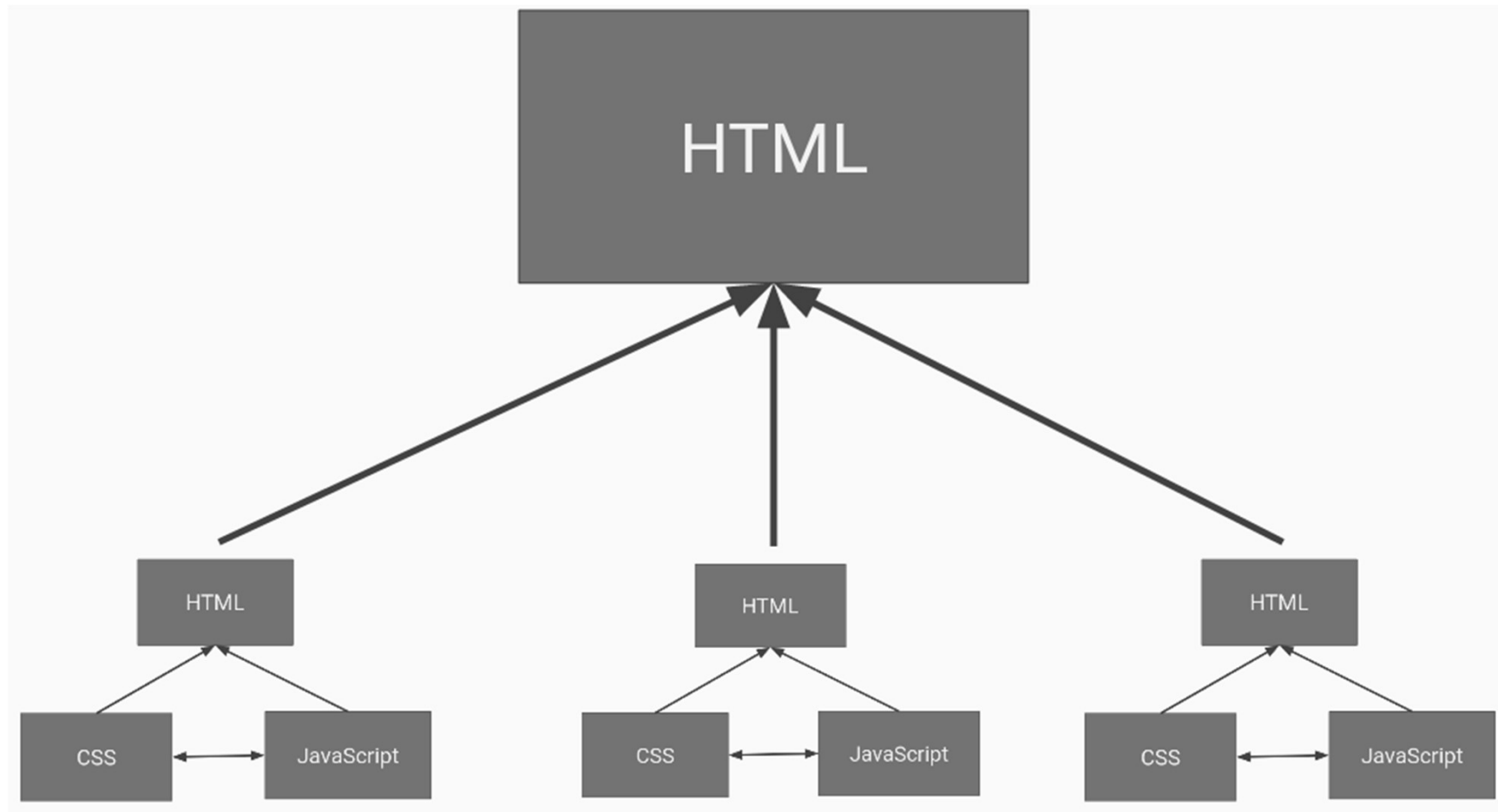
Processing of requests happens on the server side. Data manipulation and processing happens on the client side.

- 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">  
      
    <HelloWorld msg="Welcome to Your Vue.js App"/>  
  </div>  
</template>
```

HTML
structure



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>
```

CSS
presentation/appearance



Parts of a Component

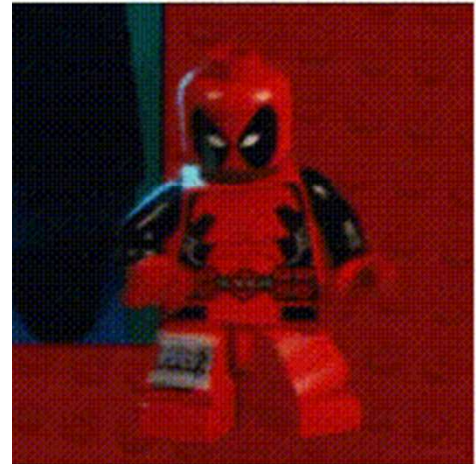
```
<script>
```

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

```
export default {  
  name: 'app',  
  components: {  
    HelloWorld  
  }  
}
```

```
</script>
```

JavaScript
dynamism/action



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!



ELEVATE  YOURSELF

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
 - ``
- Disable elements
 - `<button type="submit" v-bind:disabled="creditCardNumber == ''>Send Order</button>`
- Inject HTML (kind of)
 - `<p v-html="rawHtmlContent"></p>`

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!



ELEVATE  YOURSELF

WHAT QUESTIONS DO
YOU HAVE?



Reading for tonight: **Vue Event Handling**

