The Progressive JavaScript Framework

WHY VUE.JS? GET STARTED GITHUB

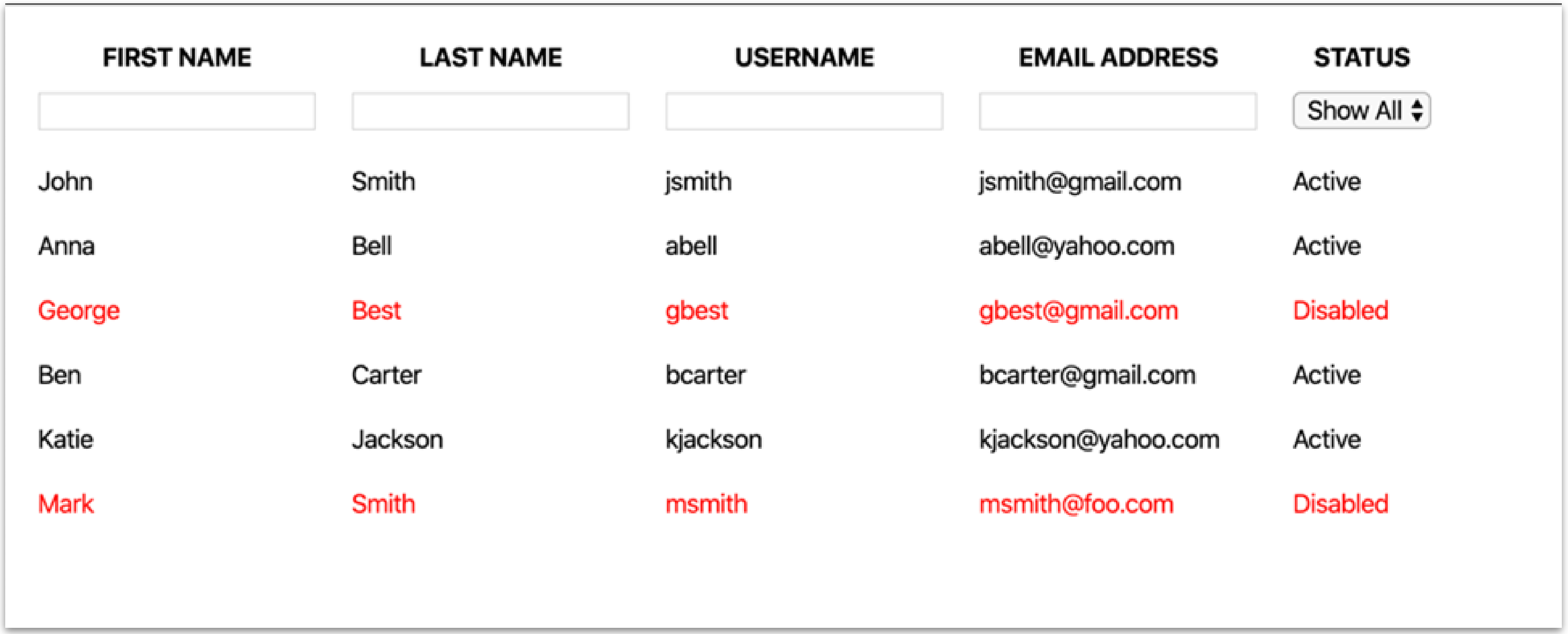
Module 4-9

Introduction to VUE

Objectives

- Create new Vue project
- What Component-based JS is
- V-model and two-way binding
- Encapsulation with components
- v-bind:class to bind data
- v-for for array of JS objects
- Computed properties

How do you build this?



FIRST NAME	LAST NAME	USERNAME	EMAIL ADDRESS	STATUS
				Show All
John	Smith	jsmith	jsmith@gmail.com	Active
Anna	Bell	abell	abell@yahoo.com	Active
George	Best	gbest	gbest@gmail.com	Disabled
Ben	Carter	bcarter	bcarter@gmail.com	Active
Katie	Jackson	kjackson	kjackson@yahoo.com	Active
Mark	Smith	msmith	msmith@foo.com	Disabled

Vue is a JavaScript Framework

Event Management

Easy DOM Manipulation

Established Patterns

Integration Points

Reusable Components

Data Binding

URL Management

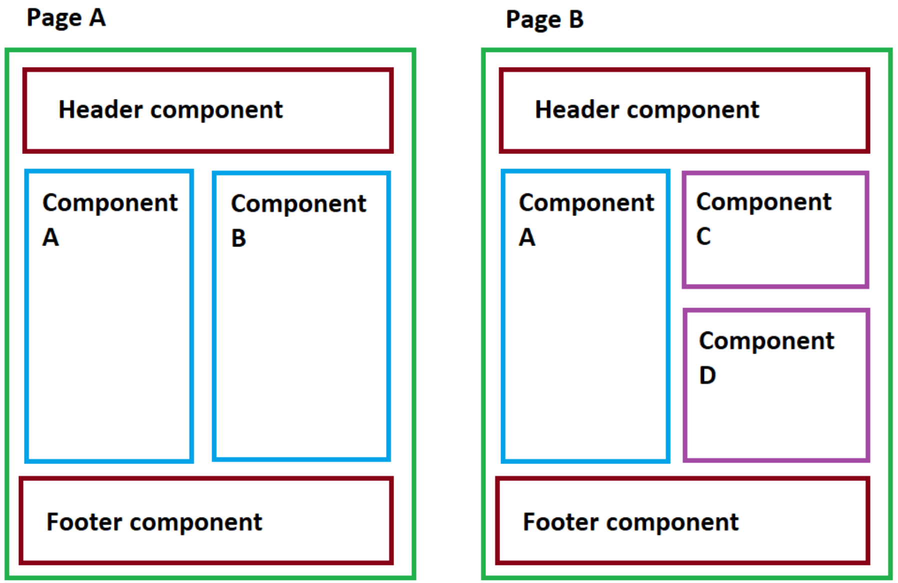
Framework

- A **programming framework** is a set of *conventions*, *tools*, and/or *processes* that you agree to follow in order to **reap a specific benefit**, usually in the form of *higher productivity* and *built-in solutions to common problems* at the **cost** of *added complexity* in your application and its development.

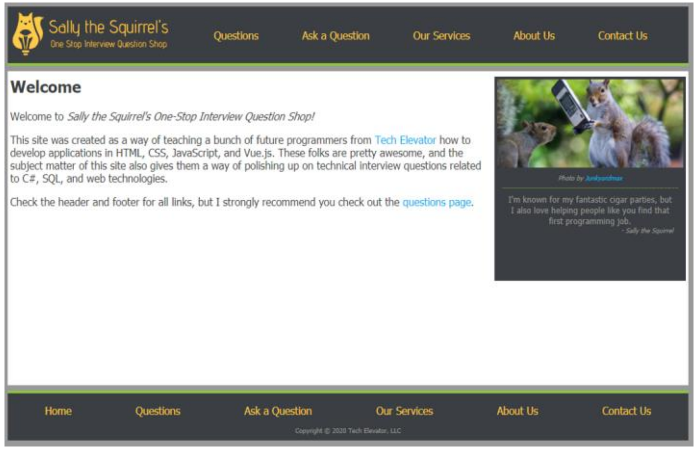
Component Based JS

- VUE.js is a component based JS Framework.
- Component based frameworks are very popular now (2020), some other frameworks you might have heard of: React and Angular.
- A component is a reusable piece of code that behaves in a “plug and play” manner, easily incorporated to other parts of the code base.
- Here is some market share data on all the various frameworks: https://www.freecodecamp.org/news/complete-guide-for-front-end-developers-javascript-frameworks-2019/

Vue Components



Vue Components



Sally the Squirrel's
One Stop Interview Question Shop

Questions · Ask a Question · Our Services · About Us · Contact Us

Welcome

Welcome to *Sally the Squirrel's One-Stop Interview Question Shop!*

This site was created as a way of teaching a bunch of future programmers from Tech Elevator how to develop applications in HTML, CSS, JavaScript, and Vue.js. These folks are pretty awesome, and the subject matter of this site also gives them a way of polishing up on technical interview questions related to C#, SQL, and web technologies.

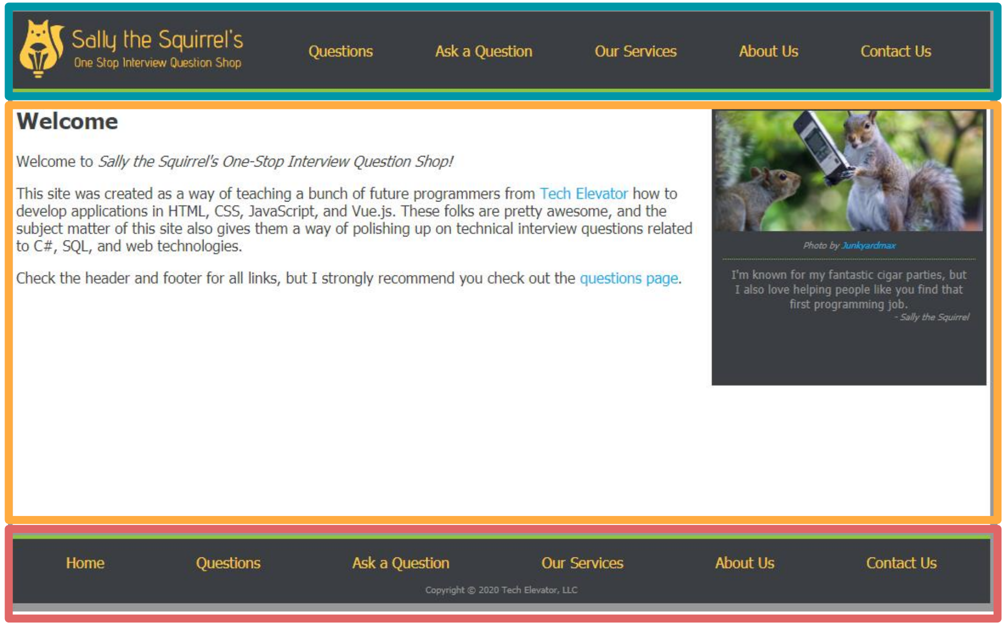
Check the header and footer for all links, but I strongly recommend you check out the questions page.

Photo by Junkyardmax

I'm known for my fantastic cigar parties, but I also love helping people like you find that first programming job.
- Sally the Squirrel

Home · Questions · Ask a Question · Our Services · About Us · Contact Us

Copyright © 2020 Tech Elevator, LLC



Sally the Squirrel's
One Stop Interview Question Shop

Questions · Ask a Question · Our Services · About Us · Contact Us

Welcome

Welcome to *Sally the Squirrel's One-Stop Interview Question Shop!*

This site was created as a way of teaching a bunch of future programmers from Tech Elevator how to develop applications in HTML, CSS, JavaScript, and Vue.js. These folks are pretty awesome, and the subject matter of this site also gives them a way of polishing up on technical interview questions related to C#, SQL, and web technologies.

Check the header and footer for all links, but I strongly recommend you check out the questions page.

Photo by Junkyardmax

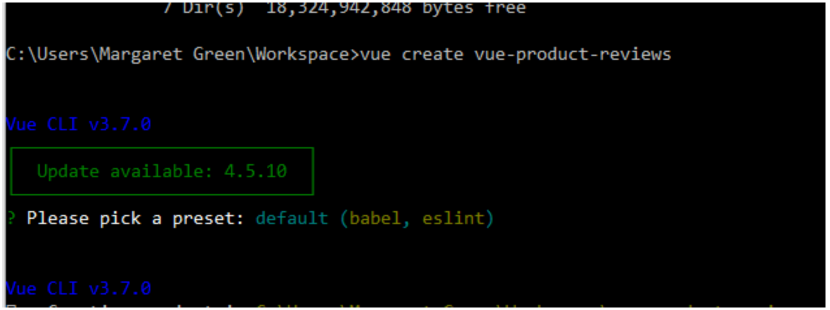
I'm known for my fantastic cigar parties, but I also love helping people like you find that first programming job.
- Sally the Squirrel

Home · Questions · Ask a Question · Our Services · About Us · Contact Us

Copyright © 2020 Tech Elevator, LLC

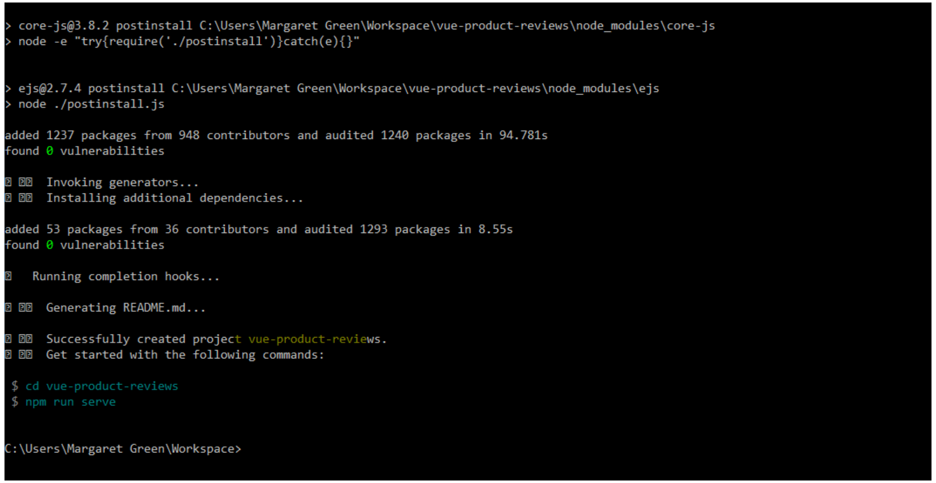
First, let’s create a VUE project!

Vue-product-reviews (lower case, use kebab-casing and choose default)



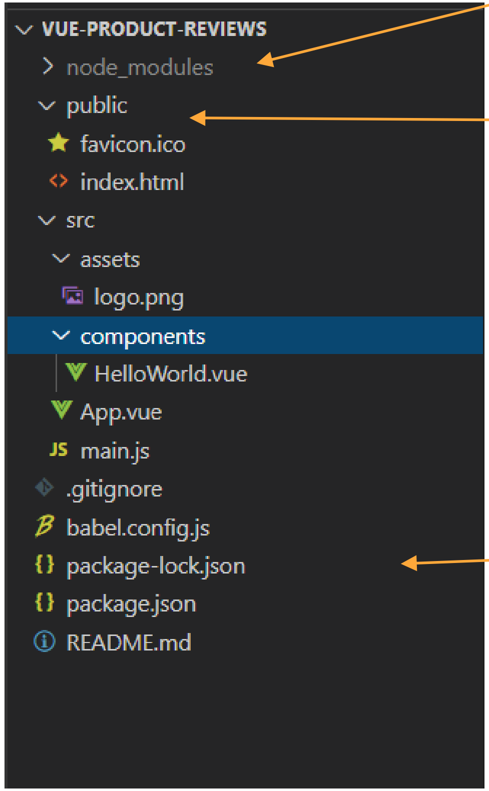
```
              7 Dir(s)  18,324,942,848 bytes free

C:\Users\Margaret Green\Workspace>vue create vue-product-reviews


Vue CLI v3.7.0
┌───────────────────────────┐
│  Update available: 4.5.10  │
└───────────────────────────┘
? Please pick a preset: default (babel, eslint)


Vue CLI v3.7.0
```

And after a long time…



```
> core-js@3.8.2 postinstall C:\Users\Margaret Green\Workspace\vue-product-reviews\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"


> ejs@2.7.4 postinstall C:\Users\Margaret Green\Workspace\vue-product-reviews\node_modules\ejs
> node ./postinstall.js

added 1237 packages from 948 contributors and audited 1240 packages in 94.781s
found 0 vulnerabilities

▯ ▯▯  Invoking generators...
▯ ▯▯  Installing additional dependencies...

added 53 packages from 36 contributors and audited 1293 packages in 8.55s
found 0 vulnerabilities

▯   Running completion hooks...

▯ ▯▯  Generating README.md...

▯ ▯▯  Successfully created project vue-product-reviews.
▯ ▯▯  Get started with the following commands:

 $ cd vue-product-reviews
 $ npm run serve


C:\Users\Margaret Green\Workspace>
```

Open in VSC



```
VUE-PRODUCT-REVIEWS
  > node_modules
  v public
      favicon.ico
      index.html
  v src
    v assets
        logo.png
    v components
        HelloWorld.vue
      App.vue
      main.js
    .gitignore
    babel.config.js
    package-lock.json
    package.json
    README.md
```

node_modules – contains all the npm packages needed for the app to run

public directory – contains index.html

src directory – contains components, stylesheets, assets, and more in here. Most important folder!

The rest:

.gitignore – sensible defaults for Git
babel.config.js – babel configuration
package-lock.json – automatically generated for any operations where npm modifies the node_modules tree or package.json
package.json – npm package meta file (containsall the build dependencies and build commands
README.md – Readme for the project

Anatomy of a VUE Component

```
<template>
   <div class="main">
       <h2>Product Reviews for {{ name }}</h2>

       <p class="description">{{ description }}</p>
   </div>
</template>

<style scoped>
div.main {
 margin: 1rem 0;
}
</style>

<script>
export default {
 name: 'product-review',
 data() {
   return {
     name: 'Widget',
     description: 'Working as intended.'
   }
 }
}
</script>
```

A VUE component is made of up of three parts:

- The **<template>** section which contains HTML code.
- The **<style>** section which contains CSS code.
- The **<script>** section which contains JavaScript code.

What the user sees on the screen is defined mostly by the HTML elements created in the template section and any CSS styles applied in the style section.

The behavior of the component is defined by what's in the script section.

A Review of JS Objects

Recall that a JS Object has a JSON data structure:

```
const employee = {
          firstName: "John",
          lastName: "Smith",
          age: 40,
          lang: ["English", "Spanish", "Esperanto"]
 };
```

- The object itself is enclosed with a set of curly braces.
- Each property of the object is listed as a key value pair.
- An array is enclosed with square brackets.

A Review of JS Objects

An objects can have other objects. Note that company has a property called employees which contains an array of “people” objects.

```
const company = {
        employees: [
                {id: “1”, name; “Alice”},
                {id: “2”, name; “Bob”},
                {id: “3”, name; “Cindy”}
        ]
};
```

<u>JavaScript objects can also contain functions</u>! This is rare, but it can be done if needed.

JS Objects in VUE

Within VUE the data function can define a JS object.

```
<script>
export default {
 name: 'product-review',
 data() {
   return {
     name: 'Widget',
     description: 'Working as intended.'
   }
 }
}
</script>
```

Note that we are returning a JS object with 2 properties, a name, and a description.

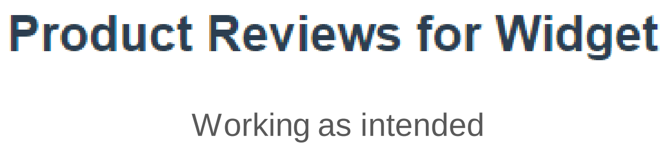
Displaying Data on the Template

Consider the following code:

```
<script>
export default {
 name: 'product-review',
 data() {
   return {
     name: 'Widget',
     description: 'Working as intended.'
   }
 }
}
</script>
```

```
<template>
   <div class="main">
       <h2>Product Reviews for {{ name }}</h2>

       <p class="description">{{ description }}</p>

   </div>
</template>
```

The HTML page will render the following:



Product Reviews for Widget

Working as intended

Formatting the Template

The <style> section can contain any valid CSS rules. Consider the code below, which applies some formatting to a div with a class name of main (which happens to be the class name on the previous slide):

```
<style scoped>
div.main {
 margin: 1rem 0;
}
</style>
```

```
<template>
   <div class="main">
       <h2>Product Reviews for {{ name }}</h2>

       <p class="description">{{ description }}</p>

   </div>
</template>
```

Let’s create a fully functional component

Integrating All your Components

- The key benefit of using a component based framework is that we can take a bunch of components and bring them together to help us achieve our goal.

- After we’ve created all necessary components, we can integrate them into the App.vue file.

- Let’s assume that the component we just built is called **ProductReview.vue**. Let’s also assume that there is another component called **HelloWorld.vue**.

- Our goal is to display both of these components on the same HTML page.

The App.VUE file:

To integrate the two components, we need to add the following code to App.VUE:

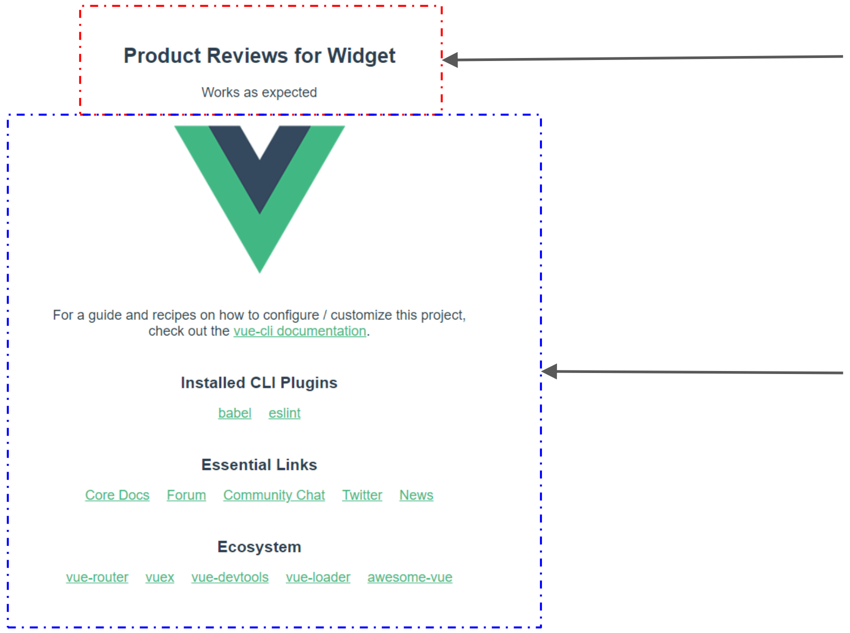
```
<template>
 <div id="app">
   <ProductReview/>
   <img alt="Vue logo" src="./assets/logo.png">
   <HelloWorld/>
 </div>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue';
import ProductReview from './components/ProductReview.vue';

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

Here we have integrated both components into the main VUE app. Note the following checklist:

- In <script> the components have been imported
- In <script>, the components object is populated with the two component names.
- You are rendering the components in the <template>

Integrating All your Components

These are the results of our labor, two fully integrated components:



Product Reviews for Widget

Works as expected

For a guide and recipes on how to configure / customize this project,
check out the vue-cli documentation.

Installed CLI Plugins

babel eslint

Essential Links

Core Docs Forum Community Chat Twitter News

Ecosystem

vue-router vuex vue-devtools vue-loader awesome-vue

The ProductReview component

The HelloWorld component

Let’s update App.vue

VUE-Directives

Before we get started on data-binding let’s introduce several VUE directives.

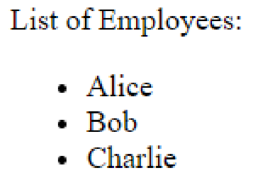
- A VUE directive is an extra attribute on a HTML element that asks the VUE library to take some kind of action on that element.

- Now we will discuss the following:
    - **v-for:** (with v-bind): loops
    - **v-model**: directly associates a DOM element to a chunk of the JSON model.

v-for

The v-for directive is used for looping. This operates in a similar manner as for each loop in Java. We want to apply the v-for on the HTML element that is going to repeat!

```
<template>
   <div class="main">
       <p>List of Employees:</p>
       <ul>
         <li v-for='employee in empList' :key='employee'>{{employee}} </li>
       </ul>
   </div>
</template>

<script>
export default {
 name: 'product-review',
 data() {
   return {
     empList: ['Alice','Bob','Charlie']
   }
 }
}
</script>
```

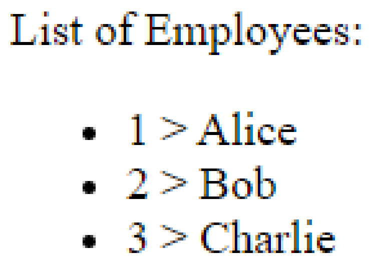


```
List of Employees:

  • Alice
  • Bob
  • Charlie
```

In here, we are looping through an array of Strings, the <li> element will be repeated three times, one for each element on the array.

v-for : (but with an array of objects)

```
<template>
   <div class="main">
       <p>List of Employees:</p>
       <ul>
<li v-for='employee in empList' :key='employee'>
{{employee.id}} > {{employee.name}}
</li>
       </ul>
   </div>
</template>
<script>
export default {
 name: 'product-review',
 data() {
   return {
     empList: [
       {id: 1, name: 'Alice'},
       {id: 2, name: 'Bob'},
       {id: 3, name: 'Charlie'}
     ]
   }
 }
}
</script>
```



List of Employees:

- 1 > Alice
- 2 > Bob
- 3 > Charlie

In the previous example we had an array of Strings, now we are working with an array of objects, necessitation dot notation, i.e. employee.name

Computed Properties

Computed properties can be thought of as custom fields based on the JSON data model. Computed properties are defined in the script section of a VUE component:

```
<script>
export default {
 name: 'product-review',
 data() {
        ...
 },
 computed: {
   metricUnits() {
      let metricMeasure = this.volumeImperial / 0.061024;
      return metricMeasure;
   }
 }
}
</script>
```

- The way computed properties are defined greatly resemble functions!
- Note that in relation to the data() section, the computed section is a peer (not a descendant) of data.

Computed Properties

We can now refer to these computed properties using the double mustache.

```
<template>
   <div class="main">
       <h2>Product Reviews for {{ name }}</h2>
       <p class="description">{{ description }}</p>
       <p>Volume in Imperial Units: {{ volumeImperial }}</p>
       <p>Volume in Metric Units:{{ metricUnits }}</p>
   </div>
</template>
```

```
<script>
export default {
 name: 'product-review',
 data() {
   return {
     name: 'Cigar Parties for Dummies',
     description: 'Banned in 50 countries',
     volumeImperial: '100'
   }
 },
 computed: {
   metricUnits() {
      let metricMeasure = this.volumeImperial / 0.061024;
      return metricMeasure;
   }
 }
}
</script>
```

A comprehensive example

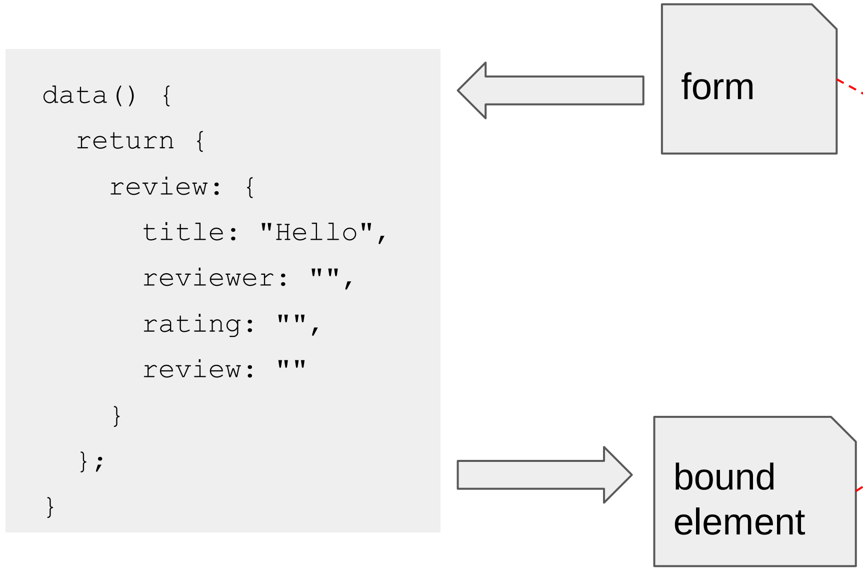
Data Binding Definition

- Data Binding techniques allow your HTML data-dependent elements to remain synchronized with its data source.
  - Consider the case of a drop-down box on HTML that lists all the Canadian provinces and US states… you could write A LOT of HTML and build this drop-down.
    - Or… you could bind the box to a JSON representation of the data.

- We have already seen plenty of examples for one way data binding where the HTML comment is derived from the JSON object inside the script section.

Two way binding: a visual

Suppose we had the following JSON object:



```
data() {
  return {
    review: {
      title: "Hello",
      reviewer: "",
      rating: "",
      review: ""
    }
  };
}
```

In our view, we have a form, input from the form will update the values of the data model.

The current values from the data model will be reflected on a bound element within the view

Two way binding: v-model

Let’s take a look at part of the form that will update the data model first using v-model:

```
<template>
 <div class="container">
   <h1>Add New Review</h1>
   <div class="row">
     <div class="col-7">
       <form>
         <div class="form-group">
           <label for="title">Title</label>
           <input
             type="text"
             class="form-control"
             id="title"
             placeholder="Enter title"
             v-model="review.title"
           />
         </div>
...
```

```
data() {
  return {
    review: {
      title: "Hello",
      reviewer: "",
      rating: "",
      review: ""
    }
  };
}
```

Note how v-model allows us to associate a form element with the JSON data model.

Two way binding: Bound Elements

We can use a mustache to have an HTML element reflect the value of the data model:

```
data() {
  return {
    review: {
      title: "Hello",
      reviewer: "",
      rating: "",
      review: ""
    }
  };
}
```

```
<div class="col-5">
  <h2>Submission</h2>
  <hr />
  <p>Title: {{ review.title }}</p>
  <p>Reviewer: {{ review.reviewer }}</p>
  <p>Rating: {{ review.rating }}</p>
  <p>Review: {{ review.review }}</p>
</div>
```

The value of the JSON object will be properly reflected on the view.

An example of two way binding