Module 4-8
Introduction to VUE

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/

First, let’s create a VUE 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”}
        ]
};
```

JavaScript objects can also contain functions! 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:

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 build 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:

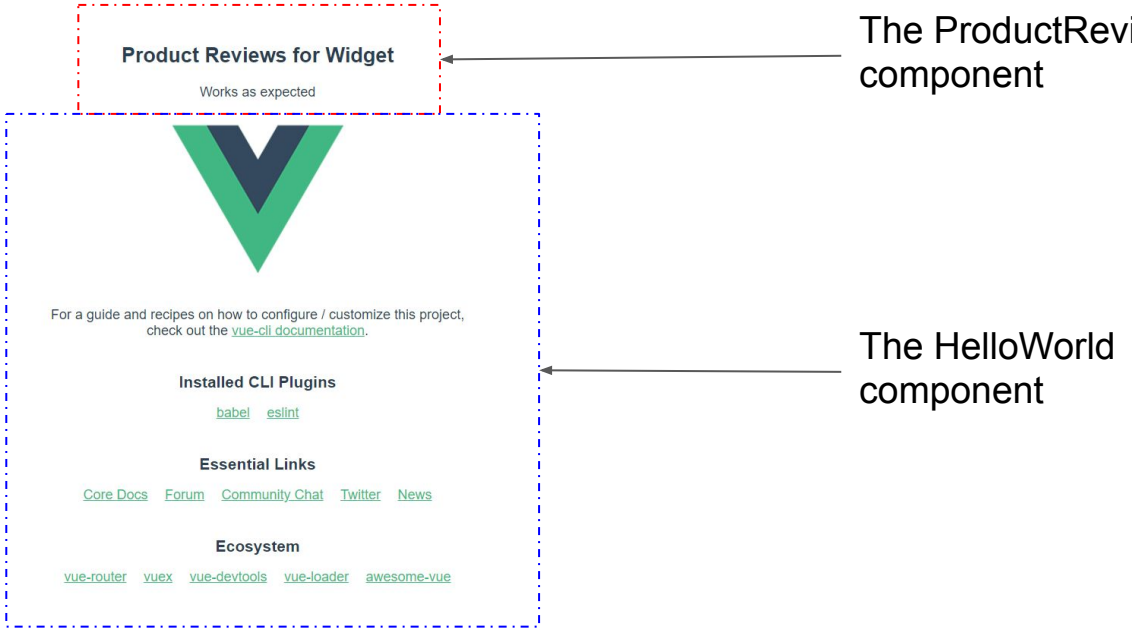
```
<template>
 <div id="app">
   <ProductReview/>
   <img alt="Vue logo" src="./assets/logo.png">
   <HelloWorld/>
 </div>
</template>

<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:



Let’s update App.vue