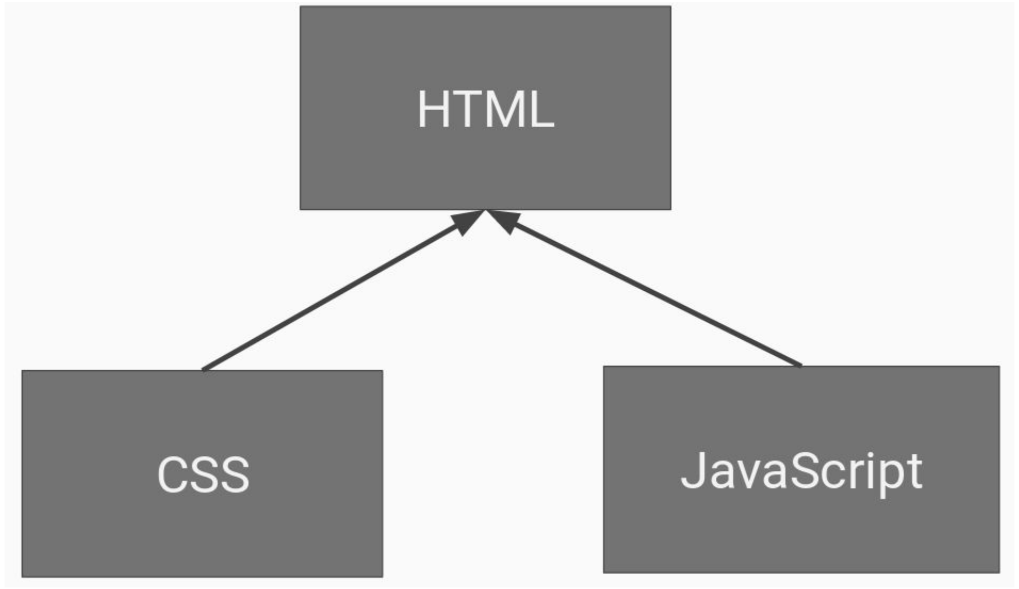
Introduction To Vue.js

Today's Objectives

- Set up a new Vue.js project via the Vue CLI
- Explain what Component-based JavaScript is and what the benefits are
- Define a new component and get it shown on the page
- Use **`v-model`** to perform two-way binding between values in HTML inputs and data properties
- Describe encapsulation and how components achieve encapsulation in JavaScript
- Use **`v-bind:class`** to bind a data property to affect applied classes to HTML elements
- Use **`v-for`** to show an array of JS objects as DOM elements
- Use computed properties to create dynamic or derived properties from data properties

Motivation for JavaScript Frameworks

Traditionally, web pages are seen as three separate pieces: a file of HTML, a file of CSS, and a file of JavaScript.



Motivation for JavaScript Frameworks

As websites got more complex, these files grew to unmanageable sizes and had a lot of repeated JavaScript code to accomplish common tasks.

- CSS was split into multiple files that could be reused, For example:
  - header
  - footer
- JavaScript code was split up in a similar way to allow code to be more reusable.

Over time, more and more functionality moved to front-end code and what was needed to support a bigger application is what developers had for years in more structured languages like Java and C#. Those languages are built to leverage good programming principles that let you reuse your code and make sure that one piece of functionality doesn't affect other pieces when they run together.

What Are JavaScript Frameworks?

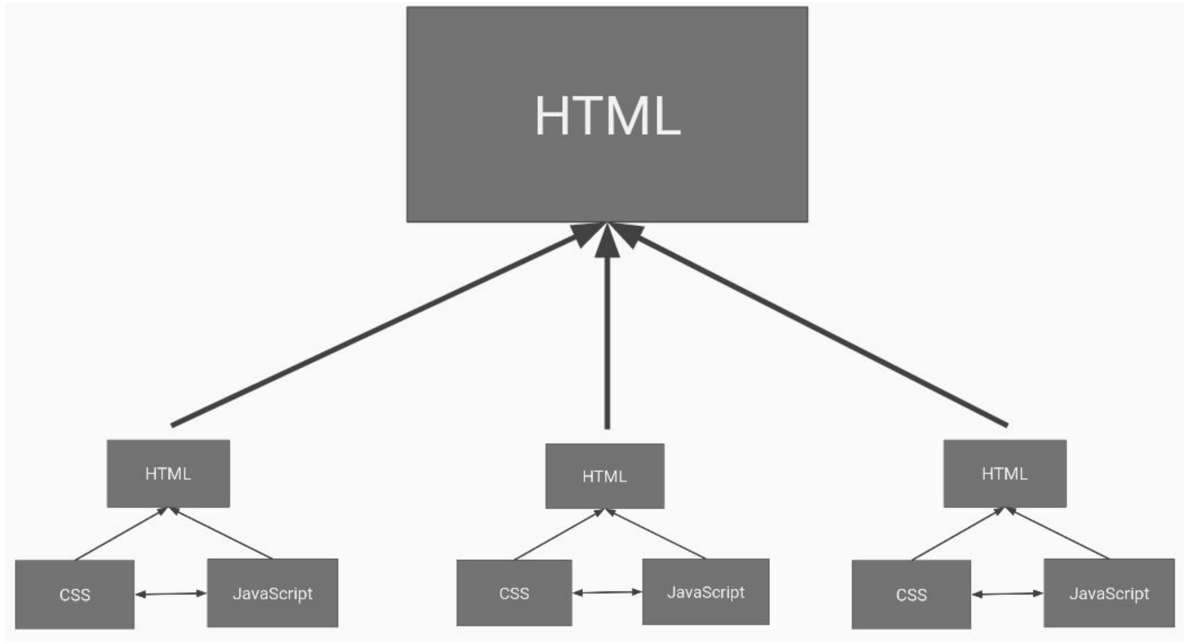
JavaScript frameworks such as Vue.js, React, and Angular, allow you to implement encapsulation in JavaScript.

- Build reusable components with
  - logic
  - data
- Not easy to accomplish in traditional JavaScript but built into component-based JavaScript.

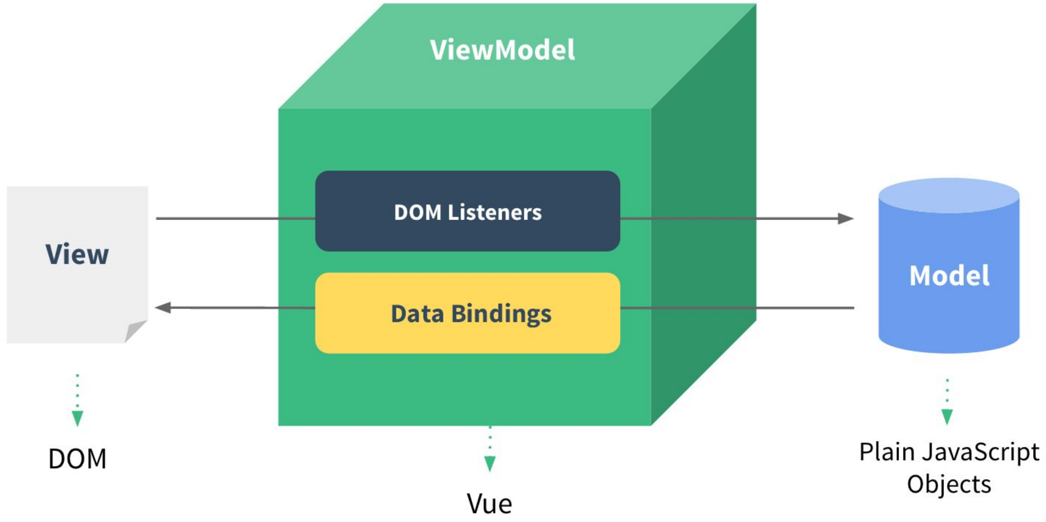
To understand components, you need to reconsider how you look at your site overall. Instead of having all of your code in three separate files, you'll start thinking of having a main HTML page that defines the main content and structure of your site, and then, include small components made out of HTML, CSS, and JavaScript to act as one single front-end element.

JavaScript Using Components

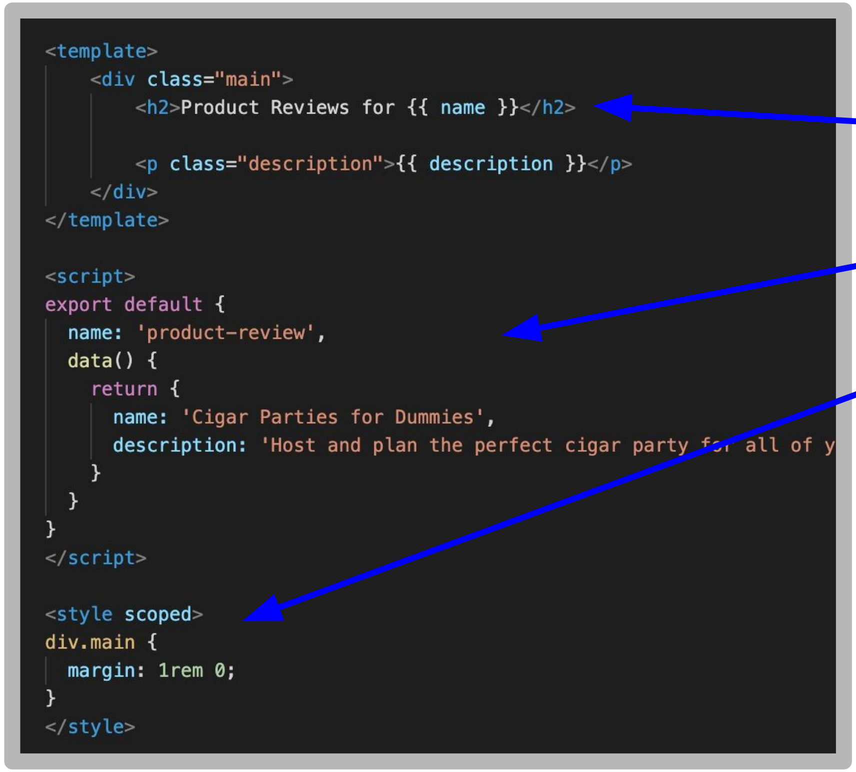
- HTML page
  - Main content
  - Structure
- Components
  - HTML
  - CSS
  - JavaScript
  - Act as single "plug-and-play" front-end element.
  - Simplify creation and maintenance



Vue.js Design



Anatomy of a Vue.js Component



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

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

<script>
export default {
  name: 'product-review',
  data() {
    return {
      name: 'Cigar Parties for Dummies',
      description: 'Host and plan the perfect cigar party for all of y
    }
  }
}
</script>

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

A VUE component is made of up of three parts:

- The `<template>` section which contains HTML elements.
- The `<script>` section contains the JavaScript code.
- The `<style>` section which contains CSS styling.

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.

Let's Create a Vue.js Project!!!

Integrating All your Components

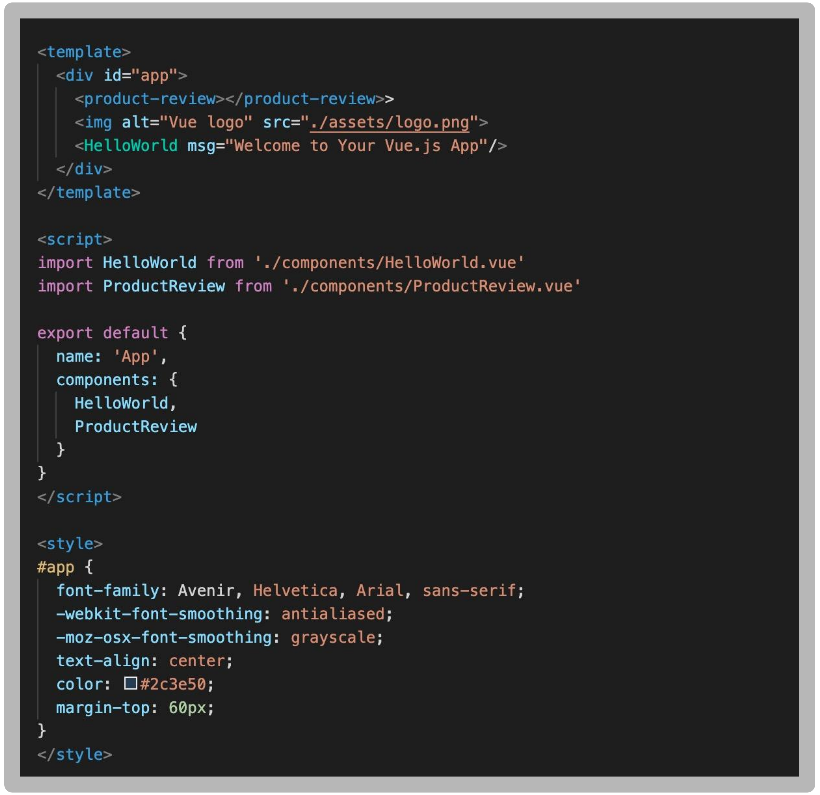
- The key benefit of using a component based framework is that we can take a bunch of components and bring them together to help us achieve our goal.

- 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.

App.vue: The Main Page

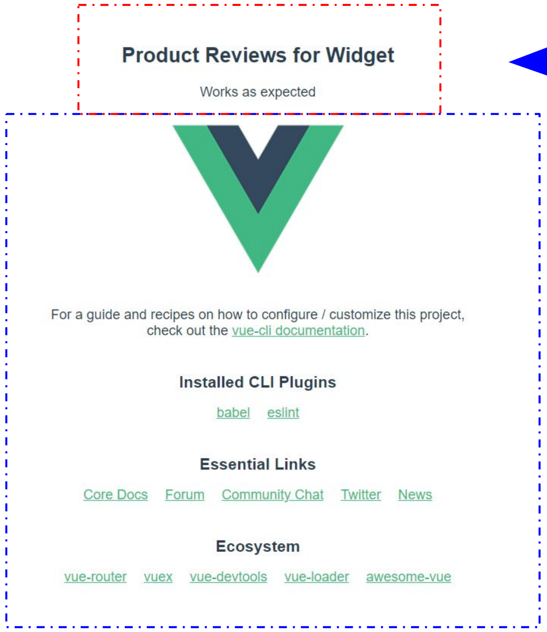


```
<template>
  <div id="app">
    <product-review></product-review>>
    <img alt="Vue logo" src="./assets/logo.png">
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>

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

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

<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>
```

Here we have integrated both components into the main App.vue file. Note the following checklist:

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

Integrating All the 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 Build Out Our Component!