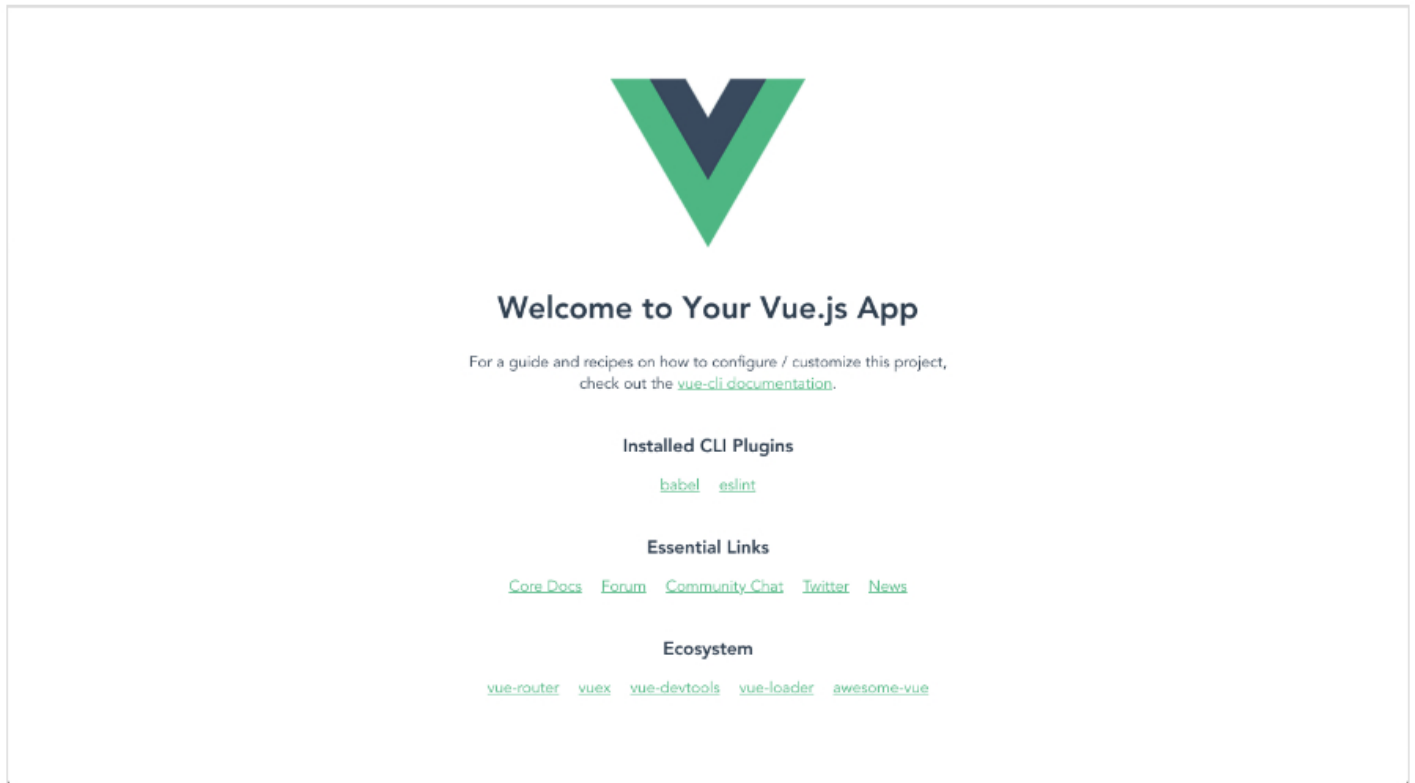


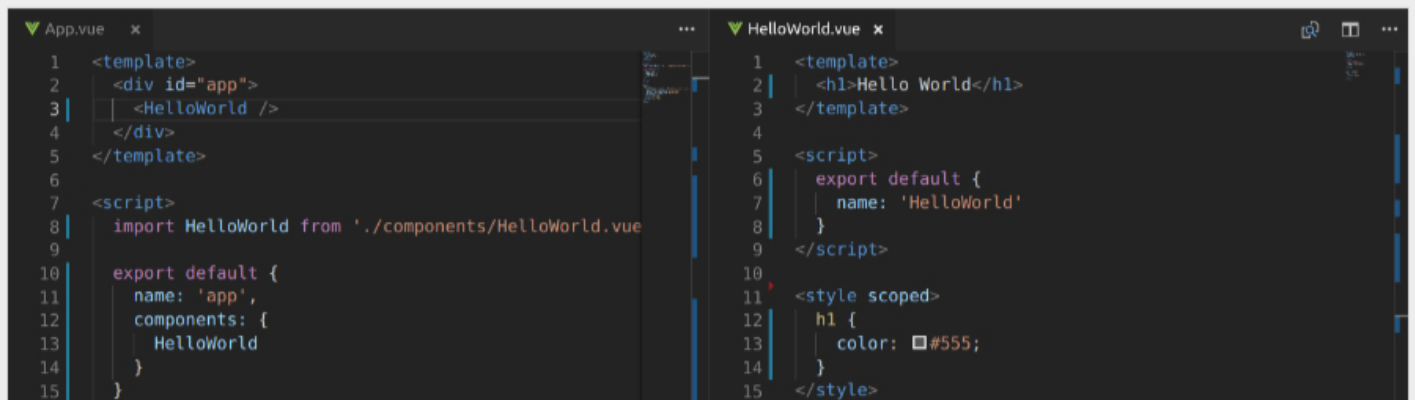


Project Setup



By running the command 'vue create app-name' an entire application is created using Vue.js and by default presents you with this screen. There are many links to documentation, references, and community forums for Vue that prove to be very useful for beginners trying to learn Vue.

Hello World Component



```

16 </script>
17
18 <style>
19   #app {
20     font-family: 'Avenir', Helvetica, Arial, sans-serif;
21     -webkit-font-smoothing: antialiased;
22     -moz-osx-font-smoothing: grayscale;
23     text-align: center;
24     color: #2c3e50;
25     margin-top: 60px;
26   }
27 </style>
28

```

```

16
17 | |

```

What you are looking at is a basic vue component that displays the text 'Hello World' on the main page of the app. On the left side is the App component that imports HelloWorld then exports it as a component. By doing this you enable the use of the HelloWorld component (shown on the right) anywhere within the '<template>' tags. In order to do this, though, you need to export the HelloWorld component from it's file under the name you wish to reference it as later on.

Passing Data Through Props

```

<template>
  <div :style="{ backgroundColor: background }">
    <h1>{{ title }}</h1>
    
    <p>{{ caption }}</p>
  </div>
</template>

<script>
  export default {
    name: 'ImageSection',
    props: ['caption', 'imgName', 'background', 'title']
  }
</script>

```

One way to get data from the parent component to the child is through the props. By defining the props name in the export, you can then use those keywords to insert the data into your component. There are two different ways you can access these variables in your component and they are either by using double curly brackets around the name of the variable, or you can use the shorthand syntax ':' prefixing an attribute to bind Vue to it and allow Vue to insert the data there.

Responsive Data

```

<template>

```

```

<div>
  <p>{{ count }}</p>
  <button @click="count += 1">Increment</button>
</div>
</template>

<script>
  export default {
    name: 'Count',
    data: () => {
      return {
        count: 0
      }
    }
  }
</script>


```

In order to add data scoped to your component you can give your export the property 'data' and then access it anywhere in your component. An interesting aspect to providing your component with data this way is that in order for the data to be contained solely within this component you need to make the value of the data a function returning whatever your data is, otherwise any instance of the same component will use and modify the same data.

This could be seen as a feature if you wanted only one set of data that is modified uniformly across the entire site. Here is an example of the component that the code in the image creates:

0 Increment

Learning Curve



[Learn](#)
[Ecosystem](#)
[Team](#)
[Support Vue](#)

- Class and Style Bindings
- Conditional Rendering
- List Rendering
- Event Handling
- Form Input Bindings
- Components Basics**
 - Base Example**
 - Reusing Components
 - data Must Be a Function
 - Organizing Components
 - Passing Data to Child Components with Props
 - A Single Root Element
 - Sending Messages to Parents with Events
 - Emitting a Value With an Event
 - Using v-model on Components
 - Content Distribution with Slots
 - Dynamic Components

Components Basics

Base Example

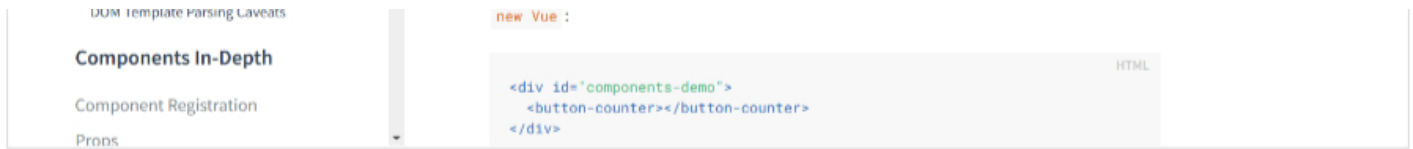
Here's an example of a Vue component:

```

// Define a new component called button-counter
Vue.component('button-counter', {
  data: function () {
    return {
      count: 0
    }
  },
  template: '<button v-on:click="count++">You clicked me {{ count }} times.</button>'
})

```

Components are reusable Vue instances with a name: in this case, `<button-counter>`. We can use this component as a custom element inside a root Vue instance created with



This photo shows a page in the Vue documentation that details how to make a component. Creating a component with responsive data in Vue appears to be relatively easy. Overall components are very easy to make in Vue and the syntax and all the different pieces that go into a component allow for a developer to quickly develop what they need.

You should use Vue.js if:

- You're looking for a **small, lightweight** library
- You're making a relatively **simple** application
- You want to be able to **easily learn** the library