VUE2 BASICS

LEARN THE BASICS OF VUE JS 2.0 IN THIS QUICK & EASY GUIDE

VUE JS HAS QUICKLY BECOME ONE OF THE MOST POPULAR JAVASCRIPT LIBRARIES OF THE YEAR. SO, LETS LEARN THIS AWESOME FRAMEWORK AND HAVE SOME FUN!

DEVDOJO

BY TONY LEA & THE DEVDOJO

Vue.js 2 Basics

Tony Lea

This book is for sale at http://leanpub.com/vue2

This version was published on 2017-05-17

Contents

Introduction

Javascript is now becoming one of the most powerful web technologies of the century. If you don’t jump on board with one of these javascript frameworks, you might get left behind. Even if you choose not to use a framework in your next project, understanding the basics of a modern javascript framework will help you comprehend concepts that you didn’t know before.

Since we want to learn one of these frameworks, why not choose one of the fastest growing javascript frameworks of the year, Vue.js. You may be surprised how simple it is to implement this library into your project. The learning curve for Vue.js is not steep at all, especially when you compare it to the many other complex javascript frameworks available. I am certain that you are going to enjoy learning and working with Vue.

In this book you will earn a belt for each chapter you complete. There are 9 easy chapters to complete. You will start off by earning your white belt and then finish off by earning your black belt in the final chapter.

With all that said, let’s begin your training and learn this awesome framework.

01 - White Belt - Installing Vue.js

Let’s get started by easing into this chapter where you will learn how to install Vue and you will also be earning your white belt!

There are 2 ways that you can include Vue in your project which are:

1. Direct Script Include
2. NPM

First, we will see how to include the script directly into a page.

Direct Script Include

The first method of installing Vue is super simple. All you have to do is include a script tag in your file.

You can download the latest copy of Vue by visiting their guide located at vuejs.org/guide or you can visit cdnjs.com and search for the latest version of Vue.

After you have a link to the **vue.js** file you can simply include it in your project like so:

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Vue.js Application</title>
</head>
<body>

    <script src="/link/to/vue.min.js"></script>
</body>
</html>
```

And it's as simple as that. Now you can start using Vue.js in your page. Couldn't be easier, right?

Ok, next let's learn how you can install Vue with NPM.

Installing Vue with NPM

Using NPM we can install the **vue-cli** tool.

```
$ npm install --global vue-cli
```

This will allows us to use a new **vue** command to create our new application.

```
$ vue init webpack my-project
```

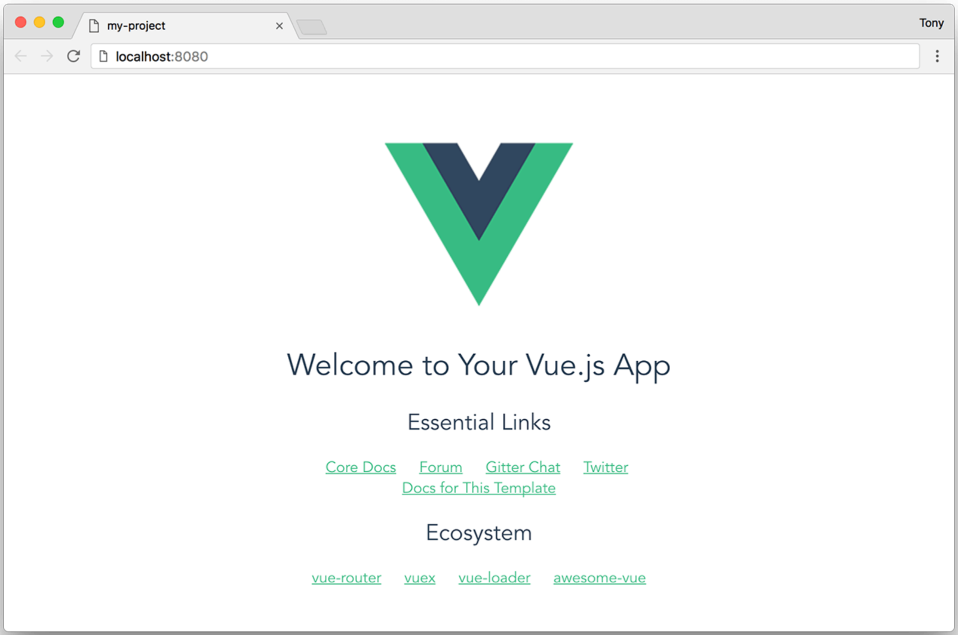
That command will initialize a new Vue.js webpack template inside of a new folder called **my-project**.

> If some of this does not make sense, do not worry. You can learn about Node and Webpack later. Throughout this book you can use the **direct script include** method of installing.

After a few command prompt steps you will then **cd** into your folder, install the node packages, and run a dev server:

```
$ cd my-project
$ npm install
$ npm run dev
```

After running **npm run dev** your browser will open up with a local development version with a Vue.js welcome page.



Welcome to Your Vue.js App

Essential Links

Core Docs Forum Gitter Chat Twitter
Docs for This Template

Ecosystem

vue-router vuex vue-loader awesome-vue

Whichever way you choose to install Vue is completely up to you.

To keep things simple. The examples throughout the rest of this book we will be using the *Direct Script Include*.

And that’s all there is to it. You just earned your Vue Basics White Belt!

Next, you will learn how to create your first Vue.js Instance.

02 - Yellow Belt - The Vue Instance

Welcome to your Yellow Belt training. In this chapter we will be talking about the Vue instance.

Every Vue application must create a **root Vue Instance** with a Vue function. This is where the application boots or loads Vue and makes it ready for use.

Take a look at how easy it is to create your first Vue Instance:

```
<script>
    var vm = new Vue({


    });
</script>
```

And just like that we've created our first Vue Instance!

Above we are setting a variable equal to a **new Vue** instance. We can also pass some options to the Vue function which we will go into further in this chapter.

Take a look below at how the whole application will look with this Vue Instance in your applicaiton. Note that we must first include the Vue.js javascript file before creating our Vue Instance.

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Vue.js Application</title>
</head>
<body>

    <script src="/link/to/vue.min.js"></script>
    <script>
        var vm = new Vue({

        });
    </script>
</body>
</html>
```

Next, you will learn a few options that we can pass to our Vue instance.

The Element Option

Inside of our Vue instance we can pass an option that specifies which *HTML Element* we want to attach to (or mount to).

```
<script>
    var vm = new Vue({
        el: '#app'
    });
</script>
```

We are telling our Vue instance to mount to an element with an ID of **app**.

Next, our application probably won't have much functionality if we don't have any data, so it makes sense for our instance to have a **data** option, right?

The Data Option

We can set or define data that we want to use within our application by defining a **data** object inside of our Vue instance.

```
<script>
    var vm = new Vue({
        el: '#app',
        data: {
            name: 'Tony'
        }
    });
</script>
```

Anything we define inside of our **data** object will be available for us to use in our application. We can store strings, integers, arrays, boolean values, and even other objects.

As you can see in the example above we added some data to our Vue instance. We added a key called **name** which contains a string with the value of **Tony**.

Lastly, there are a few hooks or functions that we can add to our Vue instance that can be called during various lifecycles of our instance.

Lifecycle Hooks

How can we run some functionality after our Vue instance is **created**? Simple enough, we can just use the **created** function/hook.

```
<script>
    var vm = new Vue({
        el: '#app',
        data: {
            name: 'Tony'
        },
        created: function(){
            console.log('Vue instance has been created!');
        }
    });
</script>
```

There are a few more important hooks that you may want to know about which are **mounted**, **updated**, and **destroyed**. Take a quick look at each of these below:

- **created** - called when the Vue instance is created.
- **mounted** - called when the ‘#app’ element is mounted to the document.
- **updated** - called anytime the Vue instance or any data gets updated.
- **destroyed** - called when the Vue instance is destroyed.

So, that’s the basics of creating a Vue instance.

And congratulations because you just earned your Yellow Belt! You’re really starting to get the hang of this. Good Job!

Next, you will learn how to bind data from the Vue instance to your application. This is where the real fun starts to happen ;)