Vue.js for Beginners

By Marina Mosti

Learning a new framework can be a very daunting process for any developer, especially for one that is still learning the base language (in this case JavaScript). This is why I have decided to create this series in which I will attempt to make the learning of Vue.js as easy and digestible as possible. ☺

I'm not a fan of making long drawn out introductions, so I will assume that if you're still reading:

1. You have some basic HTML/CSS/JS knowledge. You don't need to be an experienced front-end developer to take on on Vue as a development framework, but at the very least you need to be able to write your own HTML markup, understand the basic of how CSS works and, yes, how to write javascript. In the end, this is what this is all about.
2. That's it. No, really.

Vue as a library

There are several ways in which you can incorporate **Vue** into your web project. Let's start with the simplest one (which you will probably not end up using a lot).

Most tutorials/articles will assume that you have some understanding of how to set up a development environment in which you will use things like npm, webpack to set up your project - and while this is ideal because of what you get out of

the box - we can start with a much simpler beginner-friendly approach. The reliable old `<script>` tag.

Go ahead and fire up your favorite code editor, and create a new file called index.html. (If you don't have one yet, VS Code is a popular free choice.

```
<html>
  <head>
    <title>Vue 101</title>
  </head>

  <body>
    <h1>Hello!</h1>
    <div id="app"></div>
  </body>
</html>
```

Nothing fancy, we're just setting the bones for a simple website. Now let's get the Vue library in there. Paste this script tag before your closing `</body>`.

```
[...]
  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
</body>
```

Now that Vue is being loaded into our page, we can start using it.

Let's go ahead and create a new Vue instance, by newing it up inside a `<script>` tag. We will give it a selector by passing `#app` to the el property of the options object, and that way Vue will know where our app should be rendered. (Remember that empty `<div>` with an ID of **app**?)

Place this code after our last script tag.

```
<script>
    const app = new Vue({
        el: '#app', // 1
        data: { // 2
            myLocalProperty: 'Im a local property value' // 3
        }
    });
</script>
```

So what's happening here?

We created our new Vue instance, and pass it a configuration object. See the {} as a parameter?

1. `el:` As I mentioned before, here we tell Vue where inside our HTML we want our app to be displayed. In this case, the div with the app id.
2. `data object.` Every Vue **instance** has a local storage, like a box of variables and properties that it will hold for us and that we can use when coding our app. Data holds a JavaScript object, so we assign it one with the { } syntax. Inside, we place a property.
3. `myLocalProperty.` This property is defined inside the data object for our instance, it's name is `myLocalProperty` and the value on the right-hand side is the value - in this case, a string.

Displaying properties on our app

Right now if you open up index.html in your browser, not much is happening.

Hello!

Let's add some code to display our property inside the HTML. Your file should look like this:

```
<html>
    <head>
        <title>Vue 101</title>
    </head>

    <body>
        <h1>Hello!</h1>
        <div id="app">
          <p>My local property: {{ myLocalProperty }}</p>
        </div>

        <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>

        <script>
          const app = new Vue({
            el: '#app',
            data: {
              myLocalProperty: 'Im a local property value'
            }
          });
        </script>
    </body>
</html>
```

Pay close attention to this line:

```
<p>My local property: {{ myLocalProperty }}</p>
```

What's happening here is called *variable interpolation*, which is a fancy term for "I'm going to display the content of my `myLocalProperty` variable in this placeholder where my {{ }} are now.

Reload the page, and you will now see the string updates to reflect our variable.

Go ahead and try to change the string inside `myLocalProperty` to some other text and reload the page, you should see the text update accordingly.

Reactivity

Finally, for this lesson, let's talk about reactivity. You may have heard that **Vue** is a **reactive** framework. But what exactly does this mean? Open up your console in the chrome developer tools, and with your index.html loaded type:

```
app.myLocalProperty = 'Vue is reactive';
```

You will see the page react to this variable change!

Stay tuned for part two!

https://dev.to/vuevixens/hands-on-vuejs-for-beginners-part-1-2j2g