**https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/**

**The Difference Between a Framework and a Library**

[**Brandon Wozniewicz**](https://www.freecodecamp.org/news/author/brandon/)

Developers often use the terms “library” and “framework” interchangeably. But there is a difference.

Both frameworks and libraries are code written by someone else that is used to help solve common problems.

For example, let’s say you have a program where you plan on working with strings. You decide to keep your code DRY (don’t repeat yourself) and write some reusable functions like these:

function getWords(str) {

const words = str.split(' ');

return words;

}

function createSentence(words) {

const sentence = words.join(' ');

return sentence;

}

Congratulations. You’ve created a library.

There isn’t anything magic about frameworks or library. Both libraries and frameworks are reusable code written by someone else. Their purpose is to help you solve common problems in easier ways.

I often use a house as a metaphor for web development concepts.

A library is like going to Ikea. You already have a home, but you need a bit of help with furniture. You don’t feel like making your own table from scratch. Ikea allows you to pick and choose different things to go in your home. You are in control.

A framework, on the other hand, is like building a model home. You have a set of blueprints and a few *limited* choices when it comes to architecture and design. Ultimately, the contractor and blueprint are in control. And they will let you know when and where you can provide your input.

**The Technical Difference**

The technical difference between a framework and library lies in a term called inversion of control.

When you use a library, you are in charge of the flow of the application. You are choosing when and where to call the library. When you use a framework, the framework is in charge of the flow. It provides some places for you to plug in your code, but it calls the code you plugged in as needed.

Let’s look at an example using jQuery (a library) and Vue.js (a framework).

Imagine we want to display an error message when an error is present. In our example, we will click a button, and pretend an error occurs.

**With jQuery:**

// index.html

<html>

<head>

<script src="https://code.jquery.com/jquery-3.3.1.min.js"

</script>

<script src="./app.js"></script>

</head>

<body>

<div id="app">

<button id="myButton">Submit</button>

</div>

</body>

</html>

// app.js

// A bunch of our own code,

// followed by calling the jQuery library

let error = false;

const errorMessage = 'An Error Occurred';

$('#myButton').on('click', () => {

error = true; // pretend some error occurs and set error = true

if (error) {

$('#app')

.append(`<p id="error">${errorMessage}</p>`);

} else {

$('#error').remove();

}

});

Notice how we use jQuery. *We* tell our program where we want to call it. This is much like going to a physical library and pulling certain books off the shelf as we want them.

That’s not to say jQuery functions don’t require certain inputs *once* we call them, but jQuery itself is a library of those functions. We are in charge.

**With Vue.js**

//index.html

<html>

<head>

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

<script src="./app.js"></script>

</head>

<body>

<div id="app"></div>

</body>

</html>

const vm = new Vue({

template: `<div id="vue-example">

<button @click="checkForErrors">Submit</button>

<p v-if="error">{{ errorMessage }}</p>

</div>`,

el: '#vue-example',

data: {

error: null,

errorMessage: 'An Error Occurred',

},

methods: {

checkForErrors() {

this.error = !this.error;

},

},

});

With Vue, we have to fill in the blanks. The Vue constructor is an object with certain properties. It tells us what it needs, and then behind the scenes, Vue decides when it needs it. Vue inverts the control of the program. We plug our code into Vue. Vue is in charge.

The difference whether it is a library or framework is whether or not there is an inversion of control.

**A note on being “opinionated”**

You’ll often hear frameworks and libraries described as “opinionated” or “un-opinionated.” These terms are subjective. They attempt to define the level of freedom a developer has when structuring their code.

Frameworks are more opinionated than not since — by definition — the inversion of control requires a concession of application-design freedom.

Again, the degree to which something is opinionated is subjective. For example, I personally would consider Angular a highly opinionated framework, and Vue.js a less-opinionated framework.

**In summary**

* Frameworks and libraries are both code written by someone else that helps you perform some common tasks in a less verbose way.
* A framework inverts the control of the program. It tells the developer what they need. A library doesn’t. The programmer calls the library where and when *they* need it.
* The degree of freedom a library or framework gives the developer will dictate how “opinionated” it is.

Thanks for reading!

[**Brandon Wozniewicz**](https://www.freecodecamp.org/news/author/brandon/)

Read [more posts](https://www.freecodecamp.org/news/author/brandon/) by this author.

If you read this far, tweet to the author to show them you care. Tweet a thanks

Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers.