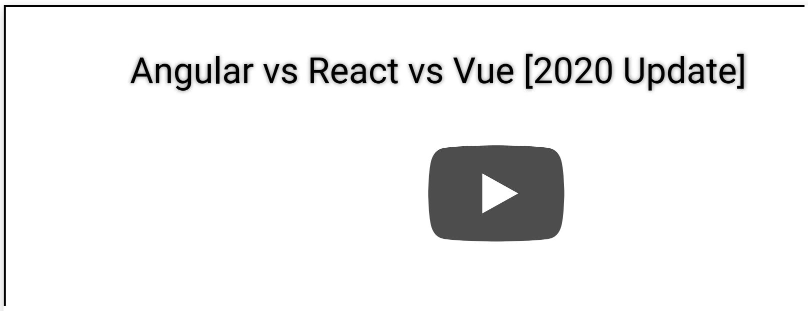


Angular vs React vs Vue - My Thoughts

React.js? Or Angular (2+)? Or Vue.js? Which one should you choose? Dive into top pros and cons!

Created by Maximilian Schwarzmüller

March 19, 2020

Watch Tutorial Video

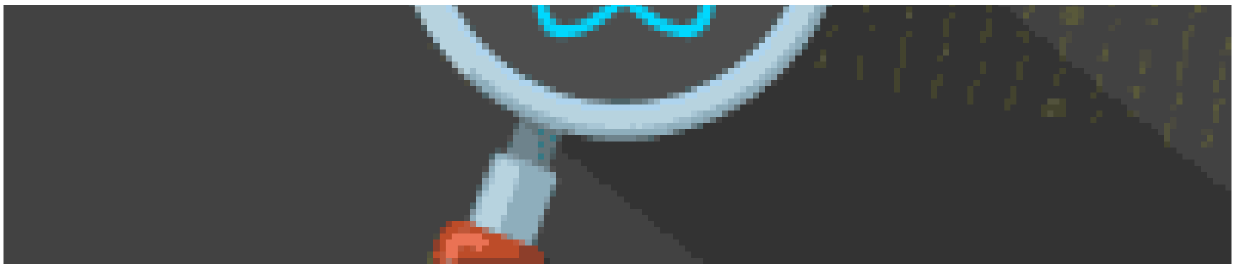
# TL;DR;

I share lots of details and thoughts throughout this article **and** the video above this article. But if you're in a hurry, here's a brief summary:

Who's working on those Frameworks? (details): Angular is developed by Google, React by Facebook, Vue is a community-driven open-source project.
Philosophies (details): Three component-focused frameworks, where Angular has a lot of built-in features, Vue has some built-in features, React is very minimalistic.
Syntax (details): Angular uses TypeScript and splits HTML + TypeScript logic apart, React uses JavaScript and a feature called "JSX" (it combines "HTML" and JavaScript logic), Vue uses regular JavaScript and splits HTML + JavaScript logic apart.

Ease of Learning (details): Vue is easiest to learn, React and Angular are on the same level and a bit more difficult than Vue.

Performance (details): All three frameworks offer great startup and runtime performance, hence "performance" will not be your main decision factor.
Adoption & Popularity (details): All three frameworks are popular but React is a bit more popular than Angular, which in turn is getting used more than Vue.
Framework Evolution (details): All frameworks are under active development, new features are being added over time to all three of them.
My Personal Opinion
Should you learn more than one framework? (details): Yes, I think you should!



React - The Complete Guide

In this course, you'll learn React from scratch. And then even further - including React Hooks, Redux & more!

VueJS - The Complete Guide

Learn Vue.js from from the ground-up and join thousands of other happy students!

# Angular, React.js & Vue

Angular, React and Vue are all highly popular JavaScript libraries and frameworks that help developers build complex, reactive and modern user interfaces for the web. Actually, with additional libraries like React Native,

Ionic (with Angular or with React) or NativeScript you can even build native mobile apps for mobile devices with help of Angular, React and Vue.

In this article, we'll not really look at **why** you would need such a framework. I'll also not start explaining those libraries here - for that, you can check out other resources, like my courses on those topics (Angular, React, Vue - all included in our Pro subscription of course).

Instead, I want to share some thoughts on how those frameworks and libraries compare and which of the three you might want to choose for your next project.

I also want to mention that there are other, smaller, libraries or technologies like Svelte which do similar things. I do have a comparison on Svelte vs Angular etc as well.

# How Do You Decide Which One Is Best?

One important note first: There is no **best** framework or library here. All three libraries are very popular for good reasons. They all have their strengths and weaknesses and you can generally use either of the libraries for any project.

In this article, we'll have a look at some strengths and weaknesses, we'll have a look at "what the market says" (jobs, download statistics etc) and I'll share my very personal opinion as well.

But to be clear about one thing right away: Ultimately, it's very important if you like a framework/ library or not. If you like its syntax, its way of approaching things and if you like how you write code with it. If you **don't like** a technology, you'll not be able to work with it successfully! On the other hand, you'll be able to get a lot out of each library if you simply enjoy working with it!

# Framework Backgrounds

Some background first:

- **Angular is a framework developed by Google**: Google also uses Angular internally, hence we'll not see Angular disappear over night. It will be maintained and continuously improved
- **React is a library built by Facebook**: Facebook also uses React internally, hence we'll not see React disappear over night. It will be

maintained and continuously improved

- **Vue is a "standalone" project** that is not built inside of any company. It used to be a one-man show (Evan You, its founder) but those days are long gone. Nowadays, it has a dedicated team of core contributors that work on Vue.

# Comparing Philosophies

Let's start with the general philosophy of each framework (side-note: I'll say "framework" here even though React.js is a "library". Get over it ;-)).

# Angular

Angular definitely is the "biggest" framework of the three. It's sometimes even called a "platform" rather than a framework.

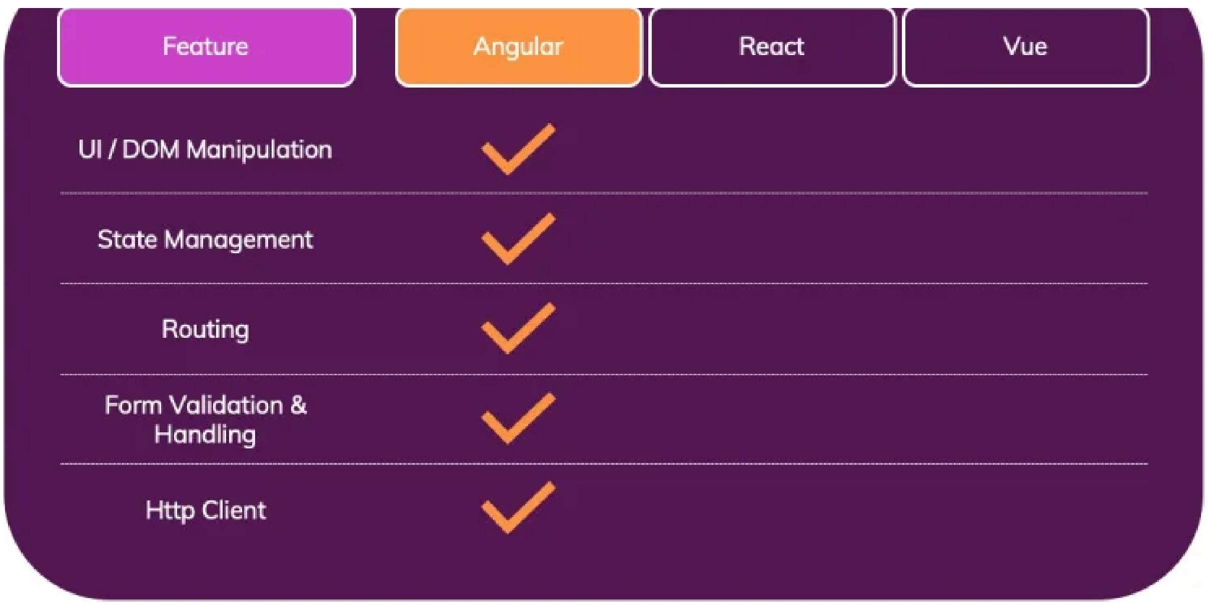
Why?

Because Angular out of the box **includes support for a lot of things**. It helps you (= the developer) with controlling the UI, reacting to user input, validating user input in forms, routing, state management sending Ajax Http requests, providing offline support & PWA capabilities, testing, building your application, managing multiple applications and connecting them and much, much more!

All frameworks have the goal of making it easier for you, as a developer, to build reactive, complex user interfaces. But Angular gives you the full set of tools for that. It doesn't stop after DOM manipulation support - it adds the above-mentioned features to help you with any aspect you could require in apps you're working on.

In addition, there's an official CLI which helps you with creating and managing Angular projects, with keeping them up-to-date, with adding dependencies and even with deployment!

In its core, Angular is all about building re-usable user interface components which you then control with Angular and which you can combine with other components to build an entire user interface from those Angular-controlled components.



Feature	Angular	React	Vue
UI / DOM Manipulation	✓		
State Management	✓		
Routing	✓		
Form Validation & Handling	✓		
Http Client	✓		

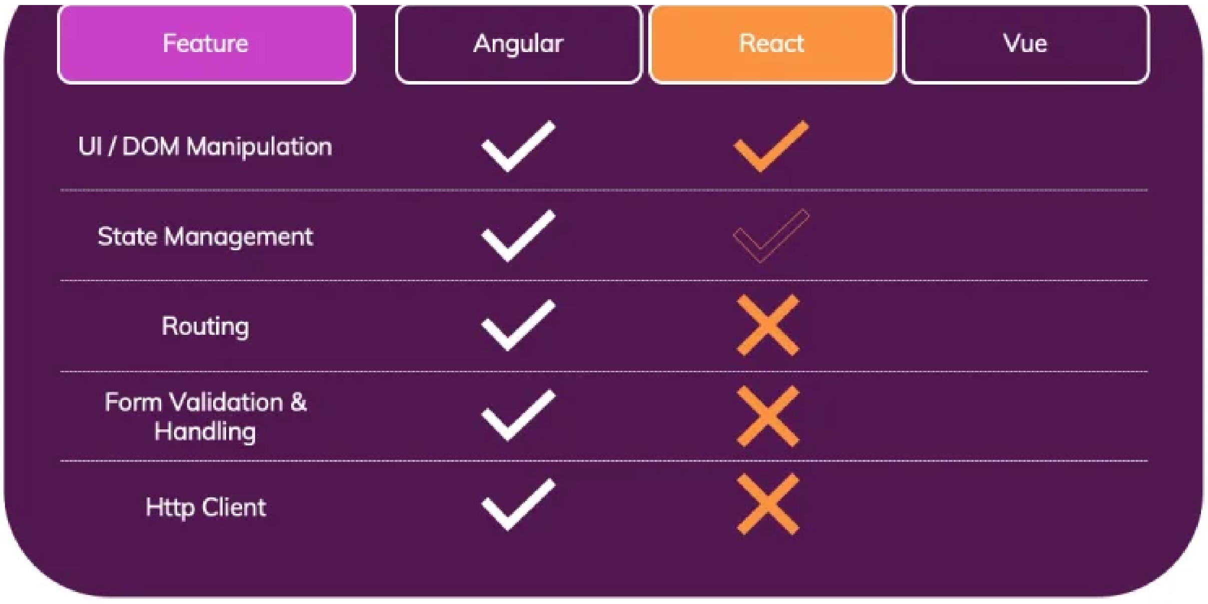
This table does not include all features Angular offers (because it does offer more than that) but instead I focused on the most important features you need in many applications.

# React (or React.js)

React is totally different!

Where Angular gives you everything, **React gives you only one thing**: A library for rendering content to the DOM and controlling it efficiently thereafter. It's also all about components and all about building user interfaces from components. It also gives you all the "tools" you need to define what should be rendered in which way under which circumstances.

But it does **not** include built-in form validation support. It does not include a router (for rendering different components based on URL changes) and it does not ship its own Http client. It has some state management support built-in but not for all scenarios. It also does not come with any other special features and it's definitely "slimmer" than Angular in that regard. For those features, you have to rely on the (arguably quite active) React community.



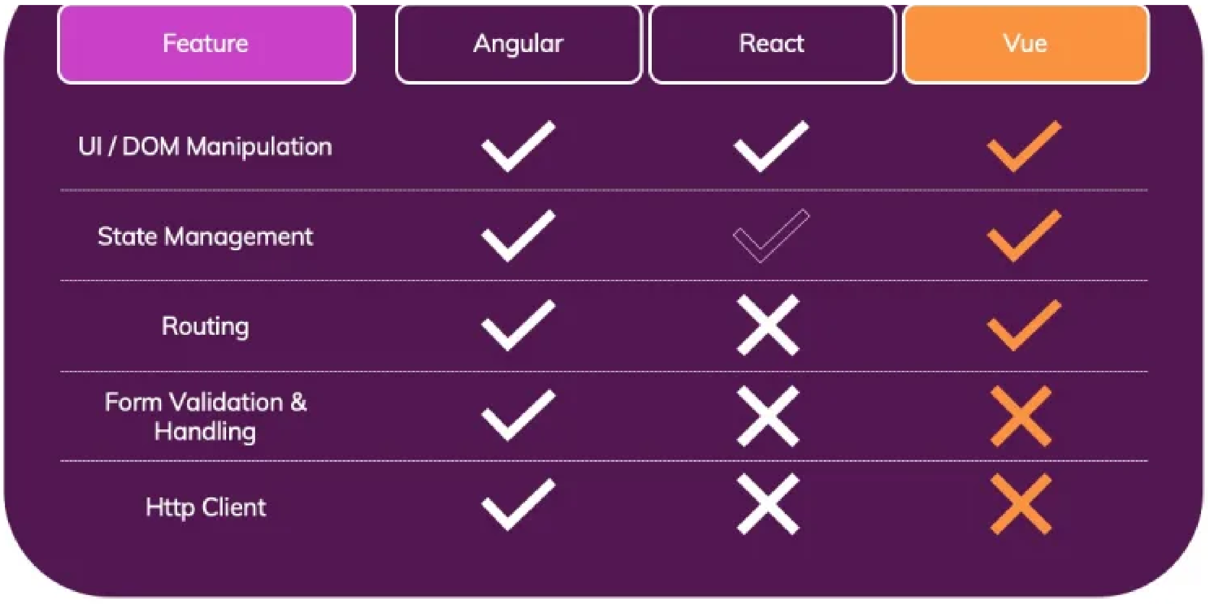
Feature	Angular	React	Vue
UI / DOM Manipulation	✓	✓	
State Management	✓	✓	
Routing	✓	✗	
Form Validation & Handling	✓	✗	
Http Client	✓	✗	

This is not necessarily good or bad (neither is it for Angular) - I'll dive into the pros and cons of those approaches a little bit later. For now, I just want to describe the different philosophies.

# Vue (or Vue.js)

Vue is a framework which kind of sits between React and Angular. It's not as "big" as Angular but it definitely includes more features than React does. Vue does give you built-in state management and it also ships with a built-in router. It does, however, not include form validation or Http client functionalities.

Just like Angular and React, the core of Vue is all about building user interfaces by combining re-usable components. But beyond that, it gives you a bit more than React and a bit less than Angular.



Feature	Angular	React	Vue
UI / DOM Manipulation	✓	✓	✓
State Management	✓	✓	✓
Routing	✓	✗	✓
Form Validation & Handling	✓	✗	✗
Http Client	✓	✗	✗

# Is "More" Automatically "Better"?

The obvious question now is: Isn't Angular much better than the other frameworks? It does offer more features after all - and that gives you, as a developer, more options.

The answer is: No, this is not automatically better. It's also not worse - it simply depends on the project you're working on and on your personal preferences.

There are some features like state management and routing which you'll need in almost every project you'll be working on - no matter how big or small that project is.

Angular and Vue have full built-in support to make that easier for you. React only has some state management support built-in (not ideal for high-frequency changes) and has **no** built-in routing support.

So why wouldn't Angular (or Vue) be better than React?

Because you could also argue that the simplicity of React and its strong focus on components and UI rendering is a big strength. Where Angular needs to connect a lot of things and ensure that they work together smoothly, React doesn't need to do that.

React does one thing and it does that thing really, really well!

For the other "tools", which you'll eventually also need, you got the very active React community. This community developed solutions (i.e. extra packages you can add to your project) like the React router, Redux or Formik.

So you could argue that the React team focuses on giving you the best possible UI-rendering library whereas the community focuses on individual projects that add on to that library. Not a bad approach.

Of course, that does also not mean that Angular's way of doing it is bad. Neither is Vue's.

The Angular team is of course a huge team of highly experienced developers, so we probably don't need to be afraid that they mess things up just because they're working on more than one library. In addition, managing it all under one hood of course also means that you'll never run into any versioning problems or incompatibilities. The different building blocks will always work together smoothly because they're managed by one team.

This is true for both Angular and Vue.

So ultimately, as mentioned earlier, there is no better approach here. And it's up to you (and the kind of projects you're working on) which approach you prefer. However, you can build any project with any of the three big libraries.

Maybe you also thought that it might a problem that Angular and Vue have more built-in features if you don't need a specific feature in a project. You could think that your final app then includes extra code which is never getting used.

This won't be a problem though. Both Vue and Angular strip out the parts which your project does not need. So if you have an Angular project without any forms, you'll not be shipping the form validation part of Angular in your final, optimized code.

# Comparing Syntax

Now that we had a closer look at the different philosophies, let's dive into some code.

Because it matters a lot! If you don't like the way you have to write code with a certain library, you'll not enjoy working with it overall. And in such a case, the library should probably **not** be the library you work with day to day.

# Angular

Angular projects use TypeScript, which is a superset (i.e. an addition) to JavaScript. TypeScript doesn't run in the browser but Angular projects include tools that will compile the TypeScript code down to browser-compatible JavaScript code under the hood.

A typical Angular code snippet would look like this:

```
import { Component, Input, Output, EventEmitter } from '@a

@Component({
  selector: 'app-user-list',
  template: `
    <ul>
      <li *ngFor="let user of users" (click)="onSelectUser
        {{ user.name }}
      </li>
    </ul>
  `,
})
export class UserListComponent {
  @Output() selectUser = new EventEmitter<{ id: string; na
  @Inout() users: { id: string; name: string }[]

  onSelectUser(id: string) {
    this.selectUser.emit(this.users.find(u => u.id === id)
  }
}
```

This is a fairly trivial component (`UserListComponent`) which uses property binding (`@Input()`) to expose data to be set "from outside" (i.e. from inside the component, this component will be used in), event binding

(`@Output()`) to emit events and a directive - `ngFor` - to render a list of data.

As mentioned earlier, I'll not explain Angular's syntax in-depth here - if you want to learn all about Angular, my Angular - The Complete Guide course is the right choice, there, you'll learn it from the ground up.

You can see, that Angular uses a TypeScript feature called "Decorators" (`@Component`) to attach extra data to normal classes (`UserListComponent`). You write code like this and Angular, behind the scenes, takes care about manipulating and "connecting" the real DOM.

So you, as a developer, never have to write any code that directly creates or removes elements to/ from the DOM. You instead create Angular components, with the syntax shown above, to let Angular do the heavy lifting.

You can define inputs (properties) and outputs (events) of components and you also can manage some component-specific or app-wide state with help of Angular (not visible in the above snippet).

# React

Let's explore a similar code snippet in React:

```
import React from 'react'

export function UserList(props) {
  function userSelectHandler(userId) {
    props.onSelectUser(props.users.find(u => u.id === user
  }

  return (
    <ul>
      {props.users.map(user => (
        <li key={user.id} onClick={userSelectHandler.bind(
          {user.name}
        </li>
      ))}
    </ul>
  )
}
```

React typically uses JavaScript (though you also can use it with TypeScript

- for example in my Ionic + React course) and it uses a special JavaScript "feature" which is called "JSX".

JSX isn't really part of the JavaScript language, React projects just are set up such that this "HTML in JavaScript" syntax is supported during development. Just like with TypeScript in Angular projects, JSX gets compiled to regular, browser-friendly JavaScript code behind the scenes once you build your project.

Just as in the Angular snippet, we built a `UserList` component here - however without decorators but instead as a simple JavaScript function which uses a concept called `props` to receive input and invoke events (`props.onSelectUser()`).

JSX is something you can love or hate - it's definitely not regular JavaScript. I initially didn't really like it but over time, I've changed my opinion. Mixing HTML and JavaScript can be confusing and for some developers, it can also be almost a sin.

I'm not as extreme but I think that it can be confusing if you're relatively new to JavaScript. Ultimately, it is just something to get used to though.

# Vue

Let's now have a look at the same component implemented in Vue:

```
<template>
  <ul>
    <li v-for="user in users" :key="user.id" @click="selec
      {{ user.name }}
    </li>
  </ul>
</template>
<script>
  export default {
    props: ['users'],
    methods: {
      selectUser: function(userId) {
        this.$emit(
          'selectUser',
          this.users.find(u => u.id === userId)
        )
      },
    },
  }
</script>
```

Vue uses regular JavaScript (though, optionally, you can also use TypeScript) and it typically utilizes something which is called "Single File Components". It is a framework all about components after all (just like the other two) but like Angular, it splits template (i.e. HTML) and JavaScript logic apart.

Vue also has support for JSX but in the vast majority of projects, you'll see code as written above.

This separation can be a bit easier to grasp for newcomers (similar to Angular) since we don't mix JavaScript with HTML here. But that's of course something highly individual.

# Which Syntax Is Best?

You might already be guessing it: There is **no best syntax**.

Some people prefer Angular's way of structuring code and using TypeScript. Other developers love JSX and having "just JavaScript" as in React's case. Or you're a fan of splitting code but you want to use JavaScript and hence you go for Vue.

It's 100% subjective and there is no objective argument here. No syntax is faster or slower than the others - it totally comes down to your preferences.

# Comparing "Ease of Learning" / Complexity

We know the philosophy, we know how the code looks like - how easy is it to learn and use those frameworks now?

That matters a lot! On bigger projects, you might need to train new developers to work with Angular/ React/ Vue. Or you personally are considering learning one of those libraries.

Of the three, Vue is probably the framework that is easiest to learn. That has two core reasons:

# 1) No Special Setup Needed

You can get started by **just importing the Vue library into a HTML file** and adding some JS to that file. No special project setup or "behind the scenes" compilation is required (you DO need a more complex setup for the "Single File Component" approach shown above though).

```
<div id="app">{{ message }}</div>
<script src="vue.js"></script>
<script>
  new Vue({
    el: '#app',
    data: {
      message: 'Hello Vue!',
    },
  })
</script>
```

# 2) Just JavaScript & HTML

Vue uses just JavaScript and HTML - no big special syntax you need to learn first. You "enhance" HTML with directives like `v-for` but for the most part, those enhancements are pretty self-explanatory.

Both Angular and React are definitely a bit more difficult to get into than Vue is.

Both require a more complex project setup with advanced developer tooling (like Webpack) to (realistically) get started. Though, to be fair, bigger Vue projects (where you want to use the above-mentioned "Single File Component" syntax) also require a more complex project setup.

Creating projects with fitting setups is made easy though - you do have the Angular CLI for Angular projects and create-react-app for React projects. In case you need a more complex Vue project setup, you can easily get it with help of the Vue CLI.

In Angular's case, you then also need to **learn TypeScript**, for React you need to get into that **JSX syntax** and understand how you write React components with "just JavaScript".

Angular at least still differentiates between HTML and JavaScript (TypeScript) logic - an approach which often is easier to grasp for new developers.

On the other hand, Angular has way more features and special "commands" built-in than React. For Angular (and also a bit for Vue), you have to learn special DOM instructions (directives) like `ngFor` (Vue: `v-for`), `ngIf` (Vue: `v-if`) etc.

I **personally** would probably **rank Angular and React equally** - though if you already have some Vue experience, Angular will probably be easier to learn than React (and vice versa). **Vue definitely is easiest** to learn amongst the three.

# Comparing Performance

Performance matters! But it's also easy to put too much emphasize on it or to run the wrong benchmarks and hence draw the wrong conclusions. Just because framework A renders 10,000 rows of `<div>` s faster than framework B, does **not** mean that framework A is overall and always better than B!

There also are different kinds of performance:

Startup performance: How fast does your web app (web page) load and become interactive (= usable by the user)?
Runtime performance: How fast is your app once it's up and running (i.e. how well does it re-render, update the UI, react to user input etc)?

(1) Startup performance is primarily influenced by the size of the created code bundle - i.e. your own code + the framework code combined.

(2) Runtime performance is primarily influenced by the internals of the chosen framework and how it approaches DOM manipulation and updating.

# Startup Performance

The overall bundle size, i.e. the code generated and uploaded to your deployment server, is what matters most here.

Often, a basic "Hello World" app is taken to compare bundle sizes of the three main players.

But I don't like that approach too much since you're typically not building just a "Hello World" app. Whilst it does give a basic idea about the size of a framework, it does not really reflect how big more realistic apps would be.

In such "Hello World" comparisons, **Angular tends to be a bit bigger** than Vue and React apps - with **Vue then being a bit smaller than React**. But again, you shouldn't interpret too much into this. It is worth noting that Angular has made a lot of progress here and that the Angular team is hard at work at **drastically decreasing the size of Angular code bundles** (source).

For bigger apps, all three frameworks should produce roughly equally-sized code bundles - assuming you are using optimization techniques like "Lazy Loading" which all three frameworks support.

# Runtime Performance

Ultimately, what counts, is the user experience. If a page "feels" fast to the end user, it does not matter if it technically is a bit slower under the hood.

Why am I writing this?

Because especially React is **not just focusing** on using all possible JavaScript tricks to detect and perform all necessary UI updates efficiently. Instead, it also utilizes techniques where it priorizies certain tasks to make the loaded page feel faster to the end user. It might, for example, handle user input with priority and delay the update of some text being output somewhere on the screen. Of course, when I write "delay" here, we're typically only talking about (a few hundred) milliseconds.

Angular and Vue might do something similar in the future, at the moment, they instead follow a different route. Both frameworks are using highly advanced techniques under the hood to really only do work when it's needed - and to then do it in the most efficient way possible.

This does **not mean** that React is doing sloppy work by the way. Angular and Vue are just implemented differently - they do use HTML templates

combined with JavaScript for example. This gives them extra room for optimizations React does not have.

Diving into the full technical details and differences is beyond the scope of this article, it's just important to understand that there are different approaches taken by the different frameworks.

Overall, all three frameworks are delivering great runtime performance and Angular, React and Vue are all getting used in production on big websites with a lot of traffic and complex user interfaces:

- Angular: Delta Airlines
- React: Facebook
- Vue: Alibaba

# Benchmarks

As mentioned, you should be careful about reading too much into benchmarks but you can check out this bechmark table to get a detailed comparison of tons of JavaScript libraries and frameworks.

# Summary

All three frameworks are very fast - both when it comes to starting and at runtime.

Angular apps can be a bit bigger than React or Vue apps - especially when we're talking about smaller applications. In most projects, this shouldn't really matter - it's also just a "snapshot" since the Angular team (but of course also the other teams) are constantly working on improving both startup and runtime performance.

Ultimately, **performance will probably not be your decision factor**.

# Comparing Adoption & Popularity

The popularity of a framework often matters - in general in life, we tend to choose things that are liked by other people, too.

When it comes to programming frameworks and libraries, there are good reasons to follow the herd:

Popular frameworks are **less likely to disappear** (i.e. they will probably be maintained by the core team in the future)

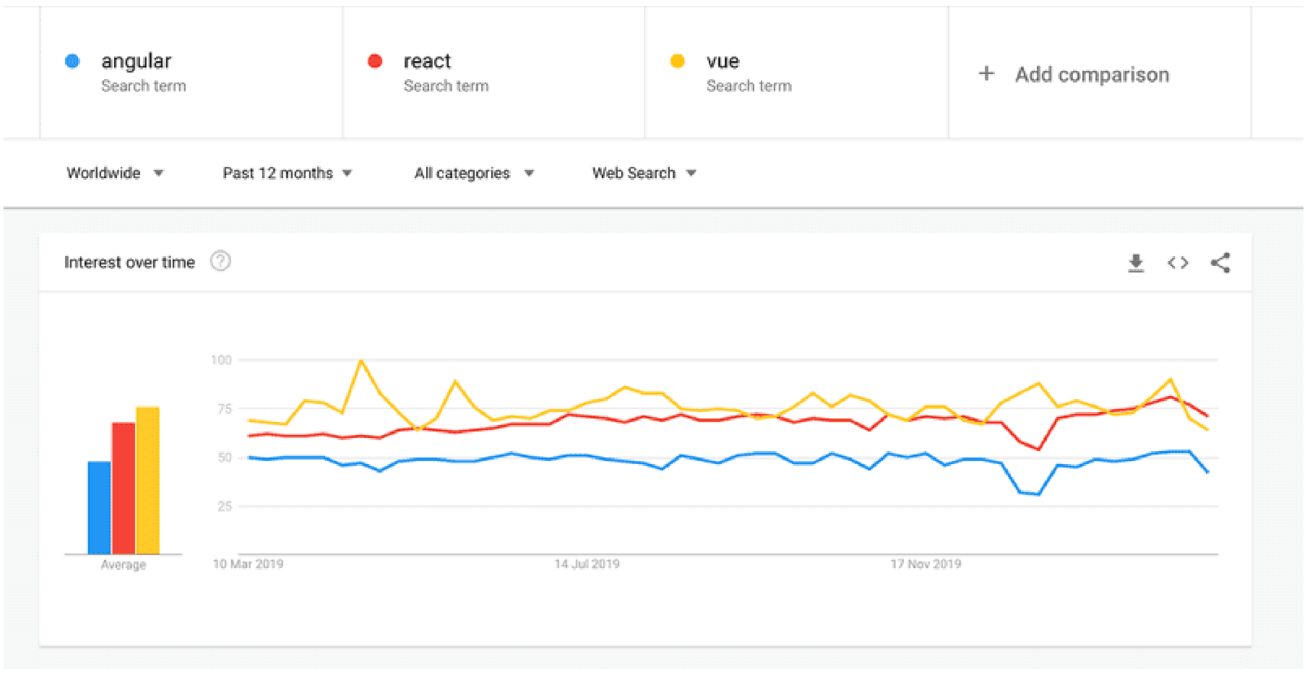
Popular frameworks tend to have **big communities** that produce complimentary libraries and tooling
You find way **more help** (tutorials, forum threads, docs) on popular libraries and frameworks
Finding a job is easier because companies typically go for mainstream solutions

This leaves only one question: **How do you measure the "Popularity of a framework"?**

I got three main measures which I like to look at:

Google Trends
npm Package Downloads
Open Job Offerings

# Google Trends

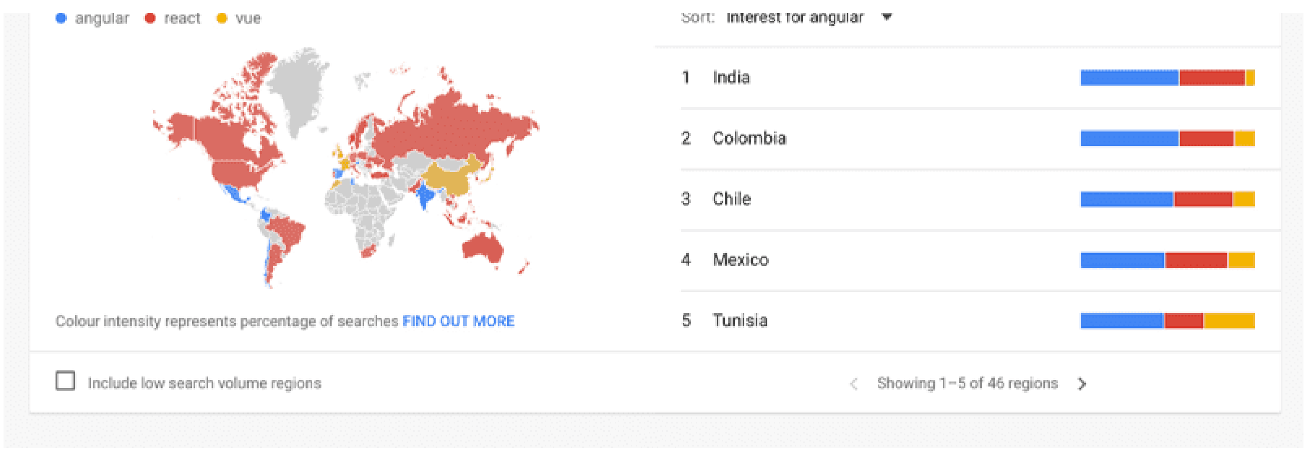


Okay, this seems to give us a clear picture!

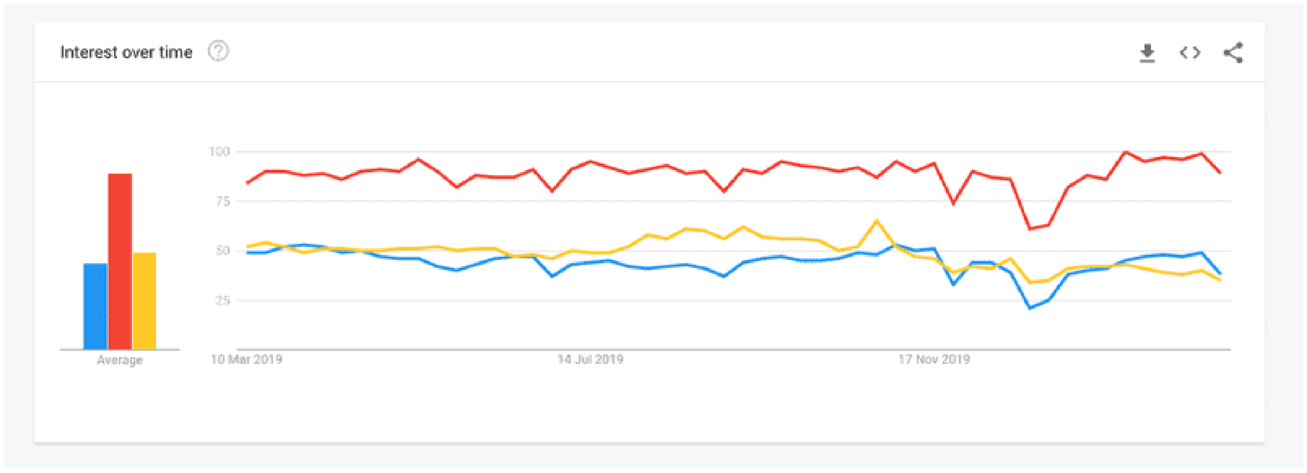
Vue and React are equally popular and they both are a bit more popular than Angular!

Well, it's not that easy.

Actually, Vue is especially popular in China.


● angular ● react ● vue
Sort: Interest for angular
1 India
2 Colombia
3 Chile
4 Mexico
5 Tunisia
Colour intensity represents percentage of searches FIND OUT MORE
Include low search volume regions
Showing 1–5 of 46 regions

If we narrow down the search to the US, we get this picture:

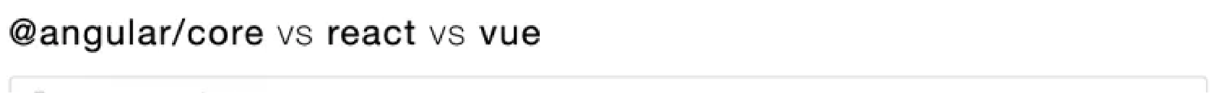


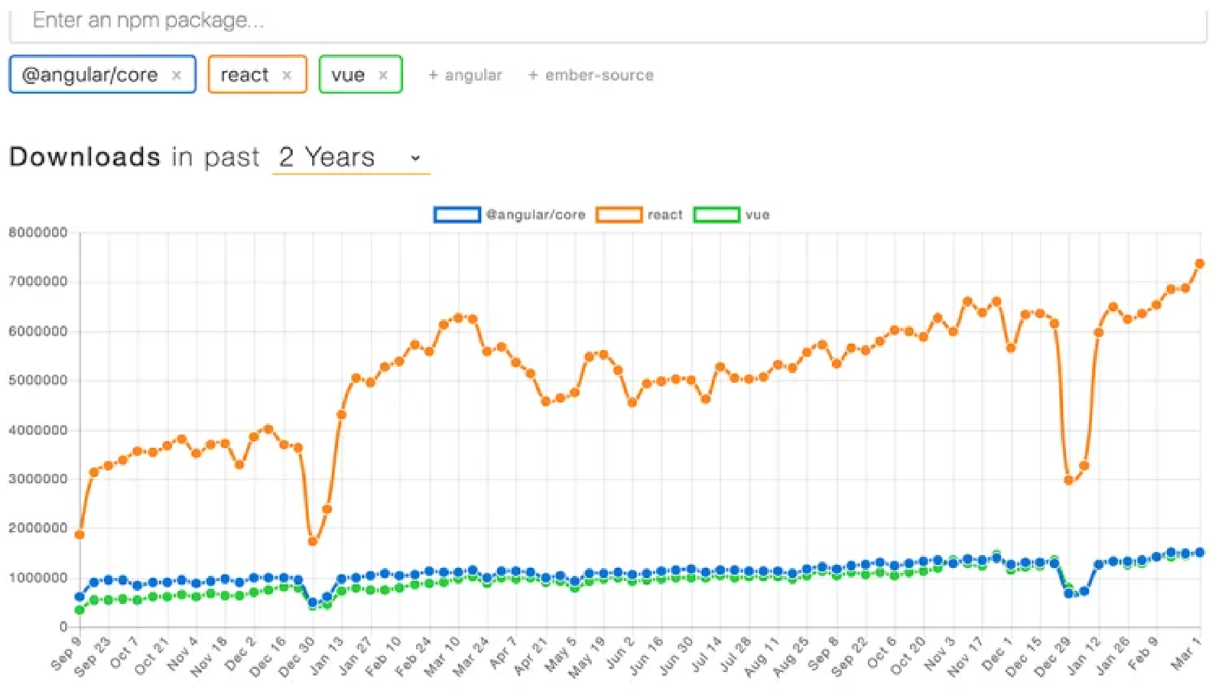
So that's a different picture! Now, React wins, with Vue and Angular being equally popular.

BUT here's the thing: **This just measures search popularity**! It doesn't say too much about the actual usage - especially not when we think about big, enterprise-size applications.

We can definitely say that there is quite some interest in all three frameworks though. We definitely have **strong and big communities for all three frameworks**!

# npm Package Downloads





Okay, this confirms the Google Trends picture, doesn't it? At least the US one. React is definitely quite a bit ahead of Angular and Vue which in turn are on the same level.

For this, you have to know one thing: Angular does also distribute cached packages to enterprise customers - i.e. those customers will not count towards the npm downloads. And Angular has a lot of enterprise customers - so the actual download numbers would be a bit higher than what we see here.

Nonetheless, it's probably safe to say that React gets downloaded more often than the other two frameworks. Indeed, it currently seems to be the most popular framework (or library), albeit only by a small margin in my opinion.

# Jobs

We of course have to **"split" popularity into "community popularity" and "company popularity"**. At least if you're looking for a job, you can't buy much if most employers want framework A but the community loves framework B.

So let's see some job offerings stats.

- Searching for **"Angular" jobs** (exact query) on March 5th 2020 gave me **16,237 job listings** on indeed.com.

- Searching for **"React" jobs** (exact query) on March 5th 2020 gave me **60,595 job listings**
- Searching for **"Vue" jobs** (exact query) on March 5th 2020 gave me **3,914 job listings**

We can clearly see that **React currently has way more job listings** than Angular or Vue. Though the distance between Angular and Vue also is quite noticable.

React indeed is getting used by big companies like AirBnB, Facebook or Uber. Angular on the other hand also has many enterprise customers and Google uses it (or migrates to it) for all of its big applications.

# Summary

Vue definitely is **very popular amongst developers** and it also has a pretty active community. On the job market, it's definitely a technology that finds you jobs but you'll have **less choice than you have for Angular or React**. This can, of course, also be an opportunity though since companies might also have fewer developers to choose from!

Angular is **popular both amongst developers as well as employers**. It's getting used for some big applications and you have plenty of jobs to choose from.

The **winner clearly is React**. It's extremely popular and it also seems to have a more fans than Angular. You find **extremely many jobs** that require React and therefore it's definitely a safe pick.

# Comparing Framework Evolution

All three frameworks are **under active development**.

This also means, that they change over time. New features are added, old features are deprecated and eventually removed. New patterns emerge.

This can be annoying but **it's a vital part of programming**. As a developer, you have to be willing to stay up-to-date with new developments since this ultimately allows you to build better and faster (web) apps that can offer more features to your end users.

# Angular

For Angular, a **new major version is released around every six months**. This does **not** mean that Angular changes every six months though. It just means that there **can** be new features (or deprecated features) every six months. Angular 9, for example, completely changed the internal runtime (no effect on the code you write though), yielding much smaller and faster bundles.

# React

React.js follows **no strict release schedule** but we see new versions being released, too. React 16 has been around for quite some time but the minor releases (16.1, 16.2 etc) also sometimes added groundbreaking new features (without removing old features though). For example, React 16.8 added "React Hooks", which allows you to change how you write stateful React components.

# Vue

Vue also is under active developent. There have been some new features that were introduced since the initial release of Vue 2 but at the moment, the Vue team is working on the next big major version: Vue 3. Vue 3 will partly be backwards-compatible and it will generally follow the same philosophy as Vue 2. But it will also add some new features to Vue. **At the moment, it's hard to tell when Vue 3 will be released though**.

# My Personal Opinion

So … what's my opinion?

Okay, that's boring: **I really like all three frameworks.**

Do I say that because I have courses on all three frameworks?

No - I do have courses on all three because **I like all three**, that's the relation.

From my own statistics (enrolments etc) I can clearly see that Angular and React are extremely popular - actually, they're way closer to each other than indicated by some of the above graphs.

I can confirm a "gap" between Angular/ React and Vue though. Vue is still extremely popular and my students are loving it, but I do definitely have less students in my Vue courses than I do in my Angular or React courses.

But these are just the numbers: **What do I like (and not like) about the frameworks?**

# Angular

I like how many features Angular brings to the table. I like that you don't have to look for third-party packages to add core features like form validation. I'm a big fan of TypeScript and I like the general Angular syntax.

At times, it can be feel a bit "big" though. There are quite a few features built-into Angular which I've (almost) never needed. This probably gives developers a lot of "power" to allow them to finetune exactly as needed but it also can be a bit overwhelming.

# React

React is really fun to use! I like it's simplicity (don't confuse this with "simple" though - writing proper React code can be challenging). I like that it's really clean and that I know when to use which built-in feature. I'm okay with JSX now (I used to not like it) and I'm also happy to use "just JavaScript" (I could use React with TypeScript, too).

Having to bring in third-party/ community-managed libraries for basic tasks like form validation is something I don't like too much though. Yes, it gives me (as a developer) more flexibility as I can finetune everything as needed. But it also makes basic tasks a bit more complicated.

# Vue

Vue is awesome! I like its syntax, how it approaches components and how easy it is to use. It feels like it has a solution for most pain points you encounter during development. I personally like using it with just JavaScript but that's of course up to you. I'm glad that it includes a router and a state management system.

I'm a bit afraid that it might become a bit more complex to learn once Vue 3 is released but it's too early to tell. In some places (e.g. emitting events)

Vue also offers you multiple possible approaches which can make it harder for newcomers to decide which approach is best.

# How Many Frameworks Should You Learn?

This was a pretty long article I guess.

I want to mention one last thing: **You can learn more than just one framework!**

You don't have to pick just one! I would indeed encourage you to at least dive into more than just one. You don't need to master all three but knowing their basics, or at least understanding two of the three frameworks will probably make you a better developer. It can also help you write better code since you draw inspirations from different framework philosophies.

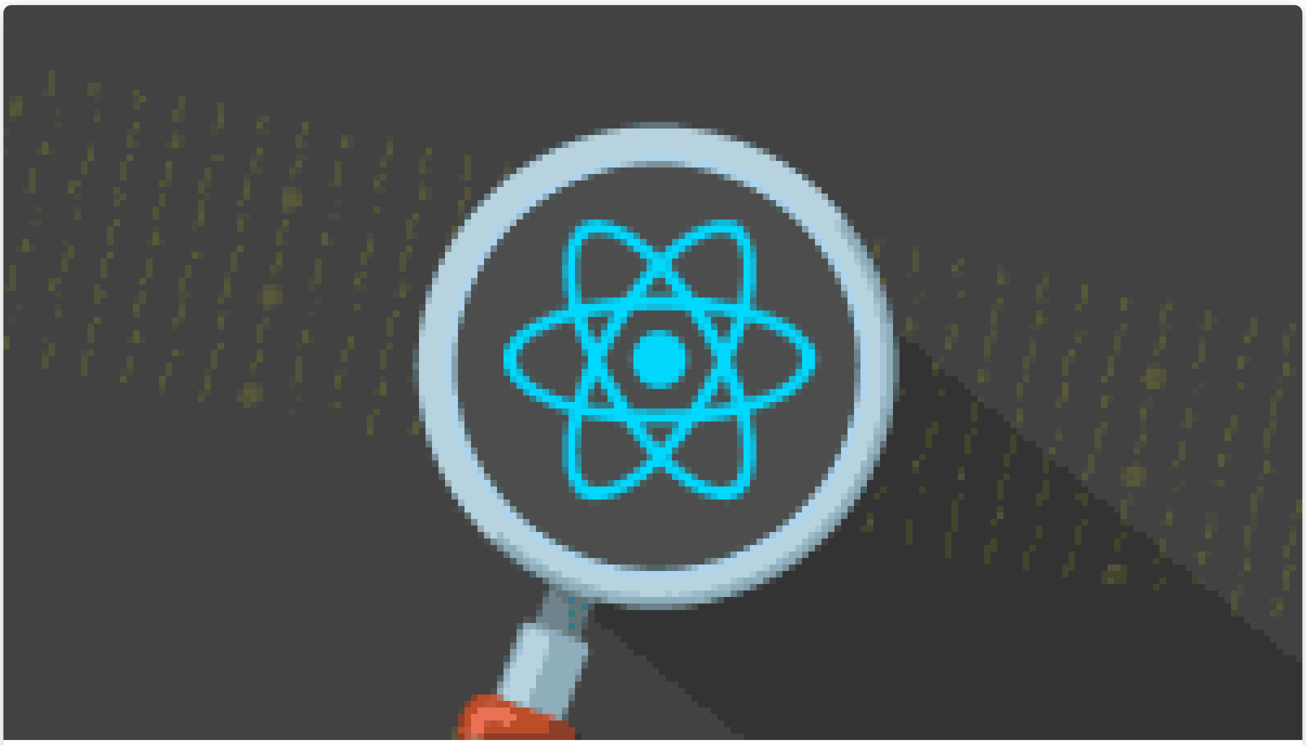
You don't have to build super-big projects just to learn a framework. You don't have to dedicate 100s of hours.

You can check out the official docs (Angular, React, Vue) to get started for free. Or you take my full courses (Angular, React, Vue) to get an in-depth walkthrough (or just take the first few sections to get that brief introduction).

Related Premium Courses

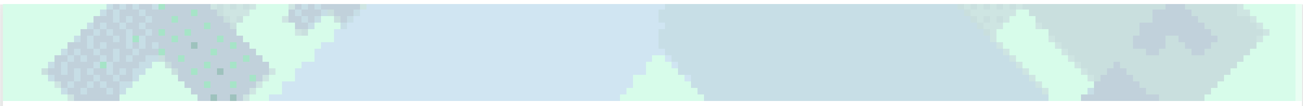
Angular - The Complete Guide

Learn Angular from A - Z with this best-selling, 5* rated complete course!



React - The Complete Guide

In this course, you'll learn React from scratch. And then even further - including React Hooks, Redux & more!



VueJS - The Complete Guide

Learn Vue.js from from the ground-up and join thousands of other happy students!