

Angular vs React: Which is Better in 2022?

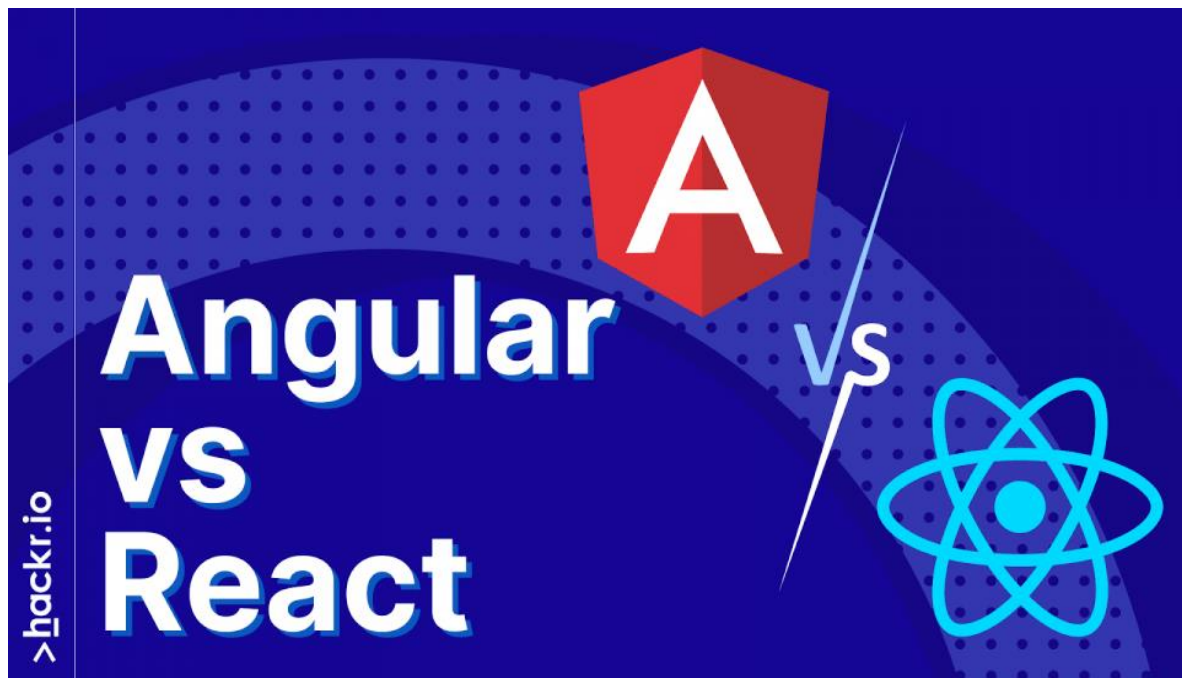


Table of Contents

We're living the years of the Javascript, as the language swiftly climbed to the first rank on Github, and is continuously gaining momentum across different applications, not just front-end application development. Nowadays, Javascript is used in the frontend, the backend, building mobile applications, and even for building AI applications. The chrome V8 engine is pushing its performance on the server-side as it is doing on the browser.

Frameworks make it easier to create applications with Javascript. Two of the most popular are Angular and React.

- Angular is a scalable, component-based framework for Javascript applications. It is a complete rewrite of AngularJS. Angular is used by Google, Microsoft, Forbes, PayPal, and Upwork.
- React is an open-source, front-end JavaScript library for the development of user interfaces. React is used by Meta, Netflix, Uber, Airbnb, and The New York Times.

Today, we'll take a deep dive into both.

Angular vs React: Head to Head Comparison

Technology	Angular	React
Technology type	Component-Based Framework using Typescript	User Interface Library with a component-based architecture using Javascript
Data binding	2-way data binding	1-way data binding
Size	Quite large and since it needs to be shipped to the client side, it increases the initial load time	Quite small in size, especially when compared with Angular
Learning Curve	Quite steep, given the number of features and options you have in Angular	It's easy to pick up and learn
Performance	Comparable to React, Angular 2 and 4 are some	Faster than Angular thanks to the Virtual DOM
Simplicity	Complex	Simple
Scalability	Easy to scale thanks to the power CLI and generation tools, It's also used by many large companies	Fairly easy to scale and is quite testable which facilitates the scaling procedure

Single-Page Application vs. Multi-Page Application?

Application development is shifting from the desktop to the browser since it provides a better UX and it's shareable across many devices. There are two main paradigms or patterns to choose from when developing a web-based application.

These two patterns are Single-page applications or SPA for short, and Multi-page applications (MPA). Now, this doesn't mean that you can't have both on the same website, in fact, in most cases, you'd want to mix your pages between both to achieve the best UX, depending on the situation.

An SPA is one that doesn't require reloading which makes for a smoother and faster performance since all the components are shipped to the client-side. Gmail is an excellent example of a SPA. The SPA requests the application shell or the Markup, and the data independently, which makes development easier.

It's also easy to debug using Chrome. However, [SEO](#) is a little bit complex when it comes to the SPA, as it might be required to perform server-side rendering or use

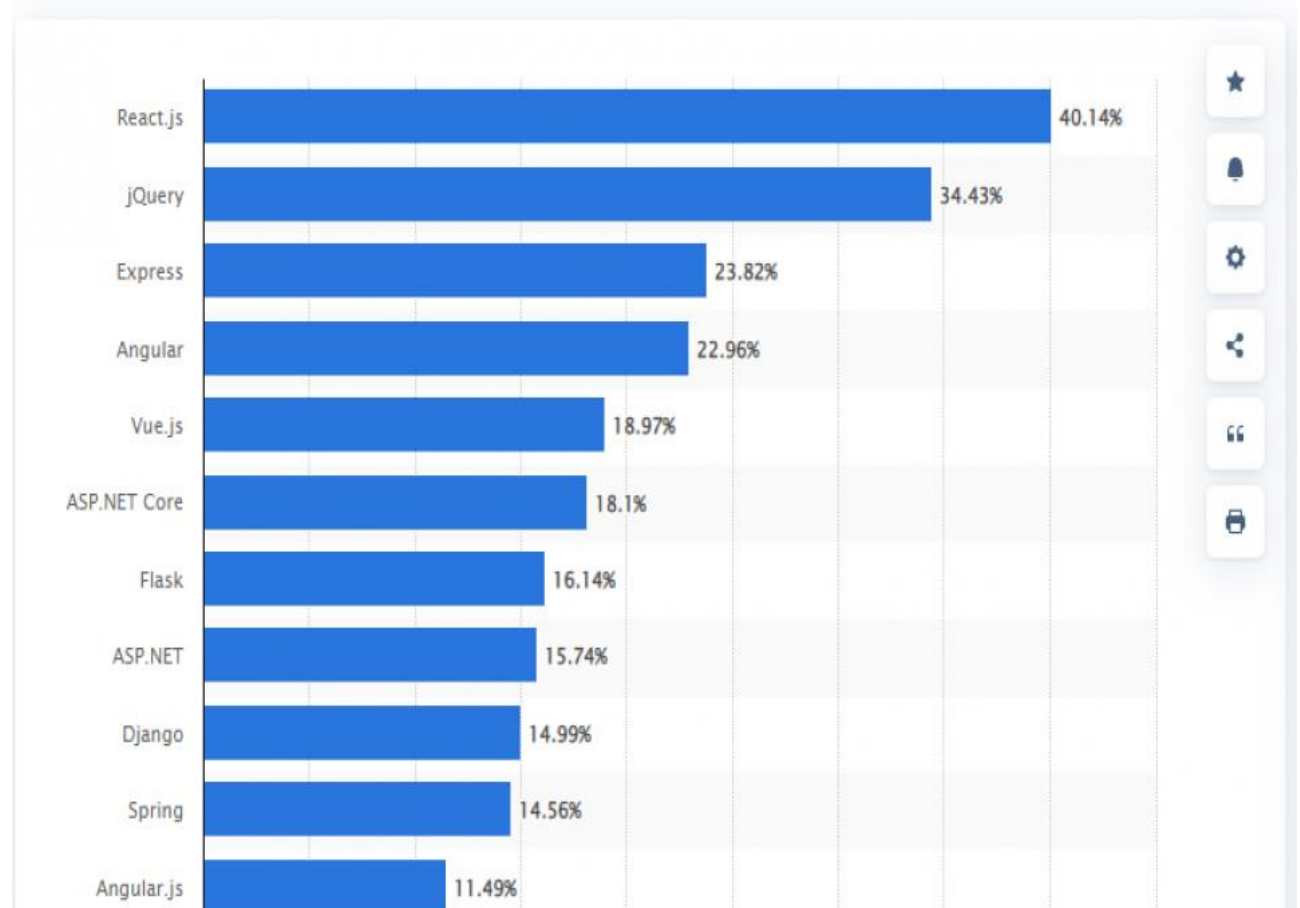
some rendering service to avoid confusing the crawler. Also, if the user has his JS turned off, you'd also need to render the pages on the server-side, which would be totally inefficient and mess up the whole SPA idea.

MPA is the traditional pages we're used to seeing, where most actions requests either data through ajax, or even entire page reloads, it's much more complex, the front-end and the back-end are usually tightly coupled, however, they are more SEO efficient.

The Basics of App Development: React vs. Angular

Now that we understand what SPA and MPA are, we can start to wrap our heads around what React and Angular are, and choose the one that best fits our application.

Frameworks and libraries accelerate the development process by leveraging the power of the tools created by other professional developers and their solutions to common problems. So, to use these tools in a much more efficient way, we have to understand how they work and what they have to offer, to choose only the ones that best fit our functional and non-functional software requirements.



Source: [Statista](#)

Today, [React](#) is the most popular framework. But jQuery and Angular are close behind.

Angular started out as an MV* framework, i.e. it could be used as MVC or an MVVM architecture. It's quite flexible, and many people regard this flexibility as a degree of difficulty since there's not one specific way of doing things. It was created by Google and is used by many companies such as Autodesk, Apple, and Microsoft. Angular focuses on building rich SPA, and when it was first released back around 2010, it shocked the world with many new concepts that it introduced.

React on the other hand is a Javascript library for building rich user interfaces.

Although React feels like a framework for many developers for the way it does things, it's not a framework. React is created and maintained by Facebook, and they use it a lot in their applications. React serves as the view in an MVC architecture, but beyond that, it has introduced a new way of doing things with its component-based architecture.

Integrating Angular and React with Other Technologies

Here's an important point to mention, in case you're not considering migrating from your already set architecture: integration.

If you're running something like a full Ruby on Rails MVC server, you can use either Angular or React. But Angular may be more difficult.

There are gems for Rails Angular, but they are outdated and not quite well maintained, the reason, of course, is that Angular was not really made to be run and developed for the server-side, although there are methods and techniques for using Angular on the server-side, we won't get into those here.

On the other hand, It's fairly easy to [integrate React](#) with your backend technology, even a strict MVC one, React works well for both, server-side rendering and client-side rendering. For rails, the react-rails gem is well maintained and gets updated regularly, so if you want to keep the entire backend functionalities and just tweak the views, React is definitely your guy.

The good news, however, is that most of the backend frameworks nowadays, including Rails, can be run and created in an API mode, and easier than before, with the aid of many packages or libraries. This indeed facilitates building whichever architecture you want for your website, and you're most probably familiar with the MEAN and MERN stacks since they're quite popular today.

Angular: An In-Depth Overview

At its core, Angular extends the browser's functionality, by extending and intercepting the event loop and enhancing the HTML parsing done by the processor. First of all, let's look at a simple Angular application, for this demonstration I'll be using the Angular 6.

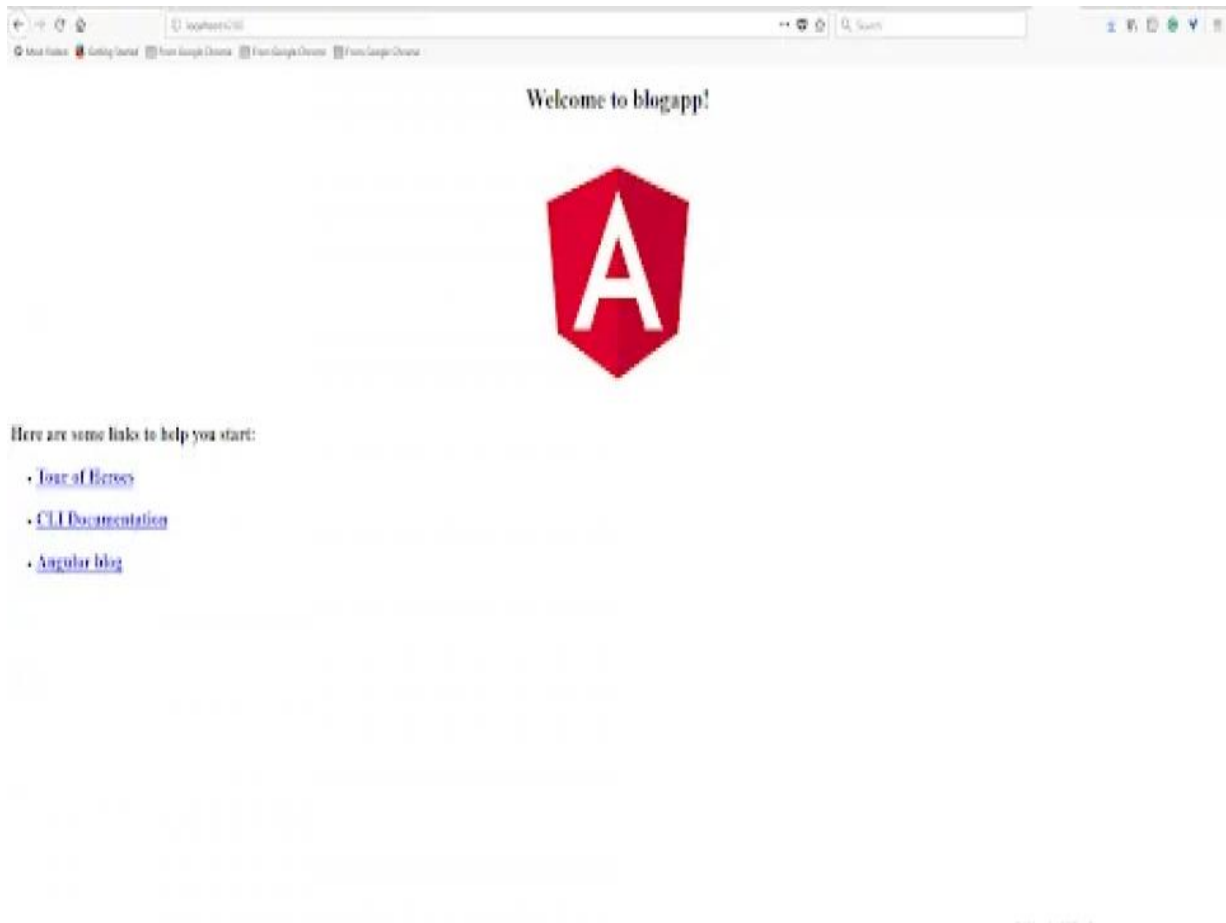
To get started with Angular, we'll install the Angular command line through npm. For this, we'll run:

```
npm install -g @angular/cli
```

Next, we'll create a new application using:

```
ng new [applicationname]
```

Angular uses a component-based architecture, basically, each component is made of 4 files, a typescript file that defines and implements the component, an HTML file that presents the view, a CSS file for style and a spec file for testing. Before we make any changes to the application, run the command `ng serve` and you'll see the default created application being run on port 4200.



Now, inside of `src/app`, clear the contents of `app.component.html`. And let's create our own new component.

The Angular CLI provides a powerful generator, to create a component we'll use `ng g component componentes/demo`. Now our application structure will look like this:



Inside `demo.component.ts`, we'll start implementing our simple two-way binding application. We'll define a simple message that should start with a default message and later on be changed and displayed on the view. The `@component` defines the metadata of the component. The selector is the custom HTML name we'll use to include this component. We'll add a message variable inside of the demo component class and define its type as a string. Inside of the `ngOnInit` method, we'll give it a default value. `ngOnInit` is a life cycle method that executes before the component is mounted, we can also use the constructor for this purpose, they will both give the same result for this application.

```
import { Component, OnInit } from '@angular/core';
```

```

@Component({
  selector: 'app-demo',
  templateUrl: './demo.component.html',
  styleUrls: ['./demo.component.scss']
})

export class DemoComponent implements OnInit {
  message: string,

  constructor: () {}

  ngOnInit {
    this.message = "Hello"
  }
}

```

Next, we'll create the HTML, we'll use an H1 tag to display the current value of the message variable. The double braces are used to interpolate the value of the message. We'll also create a form without an action, which consists of a label and an input field of type text. To create a two-way binding, we need to use ngModel which will allow us to change the value of the message variable and then the name property will display the value of the variable inside of the input field. So before we type anything, it should have the default value "Hello" written into it.

```

<h1>{{ message }}</h1>

<form>

  <div>

    <label for="message">message: </label>

    <input type="text" [(ngModel)]="message" name="message">

  </div>

</form>

```

To finalize this mini-app, we need to include the app-demo component inside of the app.component.html which is the main component of our application, and actually, include the FormsModule inside of app.module.ts to be able to use ngModel.

Source: [Github](#)

```
import { BrowserModule } from '@angular/platform-browser';

import { NgModule } from '@angular/core';

import { FormsModule } from '@angular/forms';

import { AppModule } from './app/component';

import { DemoComponent } from './components/demo/demo.component';

@NgModule({
  declarations: [
    AppComponent,
    DemoComponent
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})

export class AppModule { }
```

Now, when we save and head back to the browser, we'll find our application running with two-way binding.

Source: [Github](#)

Data-Binding in Angular

Angular, in fact, offers 3 methods of binding, not just one. Each one is demonstrated in one line of this snippet.

The first is the one we talked about earlier, the two-way data binding that reflects any change in the data immediately, if the user changes the data then the application sees it right away and vice-versa.

The second method is the one-way data-binding, in which only the application is able to change the data, and it's reflected immediately as well.

The last method is the one-time data binding, in which the data attribute is only watched changing one time throughout its life cycle and only that change is seen, if it changes otherwise we won't pick it up on the application.

Advanced Angular Features

Angular will introduce you to many other concepts and there's actually a lot you can do with Angular, for instance, we've mentioned that Angular has been extended to be more component-driven like React is, you can create custom filters in Angular, use form-validations and perform unit testing, along with many other cool things that you can check here on the official documentation.

Angular also introduced Dependency Injection which was groundbreaking at the time it was released and changed the way many things are done in development, as Dependency Injection is a useful pattern for increasing code-reusability and enhancing the development process.


Angular 5 or Angular 6?

Angular history is quite confusing. There are only 5 major releases ending with Angular 6, introduced on May 3rd, 2018.

There's Angular 1.x which is not compatible with any of the later versions since Angular 2 was a complete rewrite of the framework. Components and directives replaced controllers and scope to keep up with the uptake that React caused. Also, starting with Angular 2, the framework started to use Typescript, which is a superset of Javascript that provides static typing and a few compelling features that are not present in the regular Javascript.

It's worth mentioning that in Angular 1.5+ components were introduced to the framework as well. Angular 3 was skipped due to certain compatibility issues. Angular 4 came afterward, followed by Angular 5 which introduced the angular service workers, and then came Angular 6. It's also not recommended to make jumps from Angular 2 to Angular 6, though it's possible. We'll stick with Angular 6.

Pros and Cons of the Angular JavaScript Framework



Angular

Pros:

- ✓ Performance improvements
- ✓ Integration
- ✓ Scalability

Cons:

- ✗ Difficult syntax
- ✗ Waning in popularity
- ✗ Limited options

>hackr.io

Pros:

1. Performance Improvements. Angular offers a number of performance improvements over other similar frameworks, making it ideal for developing large and complex web applications. These include faster rendering times, smaller application sizes, and improved overall speed and responsiveness.
2. Integration. Angular integrates seamlessly with other popular tools and plugins, making it easy to add new features and functionality to your web applications. This includes tools like TypeScript, React, and Node.js.
3. Scalability. Angular is extremely scalable for large websites and web applications, which is what has made it the technology of choice for companies like Google and Microsoft.

Cons:

1. Difficult Syntax. The syntax of Angular can be difficult to learn compared to other options, such as React. This is because Angular uses a lot of complex code and terms that might be unfamiliar to developers who are new to this framework.

2. Waning in Popularity. Angular is no longer as widely used as some of the other popular JavaScript frameworks, like React and JQuery.
3. Limited Options. Angular only offers a limited number of options and tools when compared to other frameworks. This can be limiting for developers who want to use this framework to its full potential. For example, Angular does not offer a lot of flexibility when it comes to routing or animations. As a result, developers may need to look for other tools or frameworks if they want more options and flexibility when working with this framework.

Overall, while Angular does have some drawbacks, it is still an excellent choice for developing complex web applications.

React

React is an interface-making library created and maintained by Facebook. It's one of their open source projects that has been adopted by many other companies like Instagram, Yahoo, WordPress, Walmart, and the list goes on.

React brings a component-based architecture to the game, which revolutionized web development and influenced other frameworks like Angular to follow its lead.

Component-based architecture is more maintainable and even easier to maintain than other architectures in web development. It also speeds up the development process by creating reusable components that let the developer focus on each and every detail on the page.

This is how React sees your application's page and how you should think in React.

☐ Only show products in stock

Name	Price
Sporting Goods	
Football	\$49.99
Baseball	\$9.99
Basketball	\$29.99
Electronics	
iPod Touch	\$99.99
iPhone 5	\$399.99
Nexus 7	\$199.99

That being said, you can configure React as the V in your MV* architecture. Let's look at a simple Hello World program to see the component-based architecture in Action

Creating a Simple Component in React.

I've created a simple component called Hello, with default function render that simply returns the header tag "Hello World".

```
import React, { Component } from 'react';

import './App.css';

class Hello extends Component {

  render() {

    return (

      <h3>Hello World!</h3>

    );

  }

}

export default Hello;
```

In the Index.js, we merely call that component with the ReactDOM.render() method and pass in the component and where it should insert it, in this case, it will be inserted in the root div in the index.html page.

Source: [Github](#)

```
import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import Hello from './App'

import registrationServiceWorker from './registrationServiceWorker';

ReactDOM.render('Hello', document.getElementById('root'));

registrationServiceWorker();
```

When we hit npm start, the component is rendered and the magic takes place.



Hello, World

However, it's not the component-based architecture that made React all this famous; React's virtual DOM is one of the best features it has to offer. Simply said, the Document Object Model is the representation of the web page to the browser, React possesses its own virtual DOM that manipulates the actual DOM of the browser, but since it's much faster than the browser's DOM, it boosts up the performance a lot. React's Virtual DOM is able to create more than 200,000 nodes per second, which is more than a wide majority of websites would need.

Not only that, but it recreates the DOM for every change, and with The Diffing algorithm it uses, it's able to cut down the difference calculation from a complexity of $O(n^3)$ to $O(n)$. The diffing algorithm is a standalone rich subject that unveils some of React's magic, here's the [full documentation](#) from React's official website.

JSX and React

One of React's beautiful features is the introduction of JSX, the Javascript syntax extension. It's illustrated in this weird and funny snippet.

```
const element = <h1>Hello, world!</h1>;
```

While this is not HTML, it's not Javascript either. The beauty of JSX is that it helps the developer visualize the content of the page, and it's much easier to write than traditional Javascript, note that you can ignore the JSX and write traditional JS and

React would work just fine, but here's a little comparison of how the code would look like with and without JSX.

With JSX:


```
ReactDOM.render(  
  
  <h3>  
  
    Hello World!  
  
  </h3>  
  
  , document.getElementById('root') );
```

Without JSX:

```
ReactDOM.render(React.createElement(  
  
  
  'h1',  
  
  
  null,  
  
  
  'Hello World!',  
  
  
), document.getElementById('root'));
```

React is also a very rich library, we've yet to mention the props and events and forms and the whole bunch of amazing features that React brings along and that you can check in-depth on their [official documentation page](#).

Pros and Cons of the React JavaScript Framework



hackr.io

Pros:	Cons:
✓ Simplicity	✗ Comparatively poor documentation
✓ Performance	✗ Rigid architecture
✓ Easy integration	✗ Size
✓ Code reusability	

Pros:

- **Simplicity.** React is a simple and easy-to-understand framework.
- **Performance.** React uses Virtual DOM which helps in improving the performance of your web application by enabling faster updates to the DOM.
- **Easy Integration.** React can be easily used for developing large web applications without hassles.
- **Code Reusability.** React components can be reused which helps in saving development time and effort.

Cons:

- **Comparatively Poor Documentation.** One of the major drawbacks of React is that it has poor documentation when compared to other popular frameworks like Angular and Meteor.
- **Rigid Architecture.** The way in which data flows in React is quite different compared to other frameworks like Angular or Meteor. You need to learn a new way of doing things when working with React which adds to the learning curve.
- **Size.** The size of the library itself is quite large and can be problematic if you are developing an application that requires only a few functionalities from React. It is not recommended to use React without committing yourself to its extensive usage.

That being said, React is now the most popular JavaScript framework today. In part, this is because of the low barrier to entry and ease of use.

Conclusion

Web development would be much much easier if we had winners, but we sadly don't. It really depends on what you want to do and how you want to do it.

Use React if:

- You need a simple framework for building user interfaces.
- You want to build fast, lightweight web applications.
- You're primarily focused on front-end design.

Use Angular if:

- You aren't intimidated by the complexity of Angular.
- You need a more in-depth framework and model.
- You have to build robust, scalable sites.

Whether you choose Angular or React, Hackr.io has the best community-recommended programming tutorials for you.

Frequently Asked Questions

1. Is React taking over Angular?

There is no clear winner between React and Angular. While React has become more popular in recent years, Angular is still used by many large companies, such as Google and Microsoft. In terms of which framework is more popular, it depends on who you ask. Some developers prefer React because of its simplicity and easy-to-understand syntax. This makes it better for small devs and one-man-shops but doesn't necessarily make it better for large, scalable, enterprise-grade systems.

2. Is Angular or React better in 2022?

Both React and Angular have their pros and cons, and both are likely to continue evolving over the next few years. Ultimately, which framework you choose will depend on your specific project needs, as well as your own preferences and experience with different frameworks. For a JavaScript dev, it's probably worth it to understand both.

3. Is React faster than Angular?

React tends to be faster in terms of rendering speed and application size due to its lightweight nature, whereas Angular offers a number of performance improvements that make it ideal for large and complex web applications.

4. How React is different from Angular?

While they share some similarities, such as their focus on performance and scalability, React is known for its simplicity and easy-to-understand syntax, whereas

Angular offers a broader range of functionality and flexibility. Additionally, React tends to be faster in terms of rendering pages and more focused on front-end development.

5. Should I learn React or Angular?

Why not both? Both are powerful frameworks that can be used to build complex web applications, so your choice will likely depend on your specific project needs and experience with different technologies. If you are just starting out as a JavaScript developer, it might be worth it to start with React, as it is easier.

6. Do companies use React or Angular?

Companies today use both React and Angular, although React is a little more popular. Take a look at the job listings that interest you and note which one is surfacing more. It's possible one or the other may be more popular in your industry niche.