**Web Animation for Your React App: Framer Motion vs GSAP**

[[](https://semaphoreci.medium.com/?source=post_page-----dd6208440281--------------------------------)](https://semaphoreci.medium.com/?source=post_page-----dd6208440281--------------------------------)

[Semaphore](https://semaphoreci.medium.com/?source=post_page-----dd6208440281--------------------------------)

·

Follow

11 min read

·

Oct 24, 2023

68

2

Whether you’re working on a personal project or working on a large-scale application, we can agree web animations are very crucial in our websites, especially with the increasing complexities in UI and UX. With [React](https://react.dev/) being a so prominent web framework, developers must know what’s the best animation library for their use case: [Framer Motion](https://www.framer.com/motion/) or [GSAP](https://greensock.com/gsap/).

So, after reading this article, you will know about the benefits and drawbacks of each one, the key differences, how to set them up in your project, best practices and the how to pick the best animation library for your React project.

**Prerequisite**

To properly understand this article, you need to have a basic knowledge of :

* [HTML](https://www.w3schools.com/html/)
* [CSS](https://www.w3schools.com/html/)
* [JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript)
* [React](https://react.dev/).

**What is Framer Motion?**

[Framer Motion](https://www.framer.com/motion/) is a web animation [library](https://www.npmjs.com/package/framer-motion) built particularly for [React](https://react.dev/). The library contains everything needed for implementing motions in your React App. Being a React library, its basic implementation is its syntax which makes it quite easy to use for seasoned developers. It provides animation instances for :

* **Layout**: which manipulates the structure of the webpage and positioning of elements.
* **Gestures**: with the help of event listeners which manipulate the motion of elements according to the actions of the user, for instance, when the user hovers on, taps, or drags an element.
* **Scroll**: which manipulates scroll-linked and scroll-triggered animations.
* **Transitions**: which specify how values animate from one state to another.

Implementing these animation sequences without a library is time-consuming. So, having ready-made animation sequences in Framer Motion also saves a ton of time.

**What is GSAP?**

The [GreenSock Animation Platform(GSAP)](https://greensock.com/gsap/" \t "_blank) is a web animation library for JavaScript-powered tools or sites in general. Unlike Framer Motion which is used for animation in React, GSAP provides support for a variety of other frameworks like [Vue](https://vuejs.org/) and [Angular](https://angular.io/). If you’re not making use of a framework, GSAP is still your guy. In short, **it’s capable of adding animation to any site powered by JavaScript**.

Since GSAP was not made particularly for React, the library does not contain built-in React animation sequences to use. It gives the flexibility to create custom animation sequences and manipulate elements using JavaScript.

**Differences between Framer Motion and GSAP**

The major difference between Framer Motion and GSAP is its range of support. While Framer Motion supports functionality for only React applications, GSAP goes a further mile in supporting other frameworks and tools.

Another distinctive difference is that while Framer Motion has a limited range, GSAP offers flexibility in creating sequences which include:

* Creating SVG animations.
* Manipulating canvas animations.
* Creating animation for WebGL projects

**Comparison between Framer Motion and GSAP**

This section will help you in choosing the perfect animation library for your next project. It will cover a comparison between Framer Motion and GSAP across various factors like their syntax, performance, size, optimization, working principle, ease of learning and predicted relevance and active community. Let’s see how they compare.

**Syntax**

For Framer Motion, it consists of the motion component and the animation property, the animate property is often used to trigger animations. The basic syntax is shown below :

<motion.div animate={{ x: 100 }} />

For GSAP, the syntax consists of a **method**, a **target**, and a **variable**. Let’s see how we would write the above code snippet using GSAP.

gsap.to(".box", { x: 100 });

From the above syntax, gsap.to() is the method, ".box" is the target, which specifies the class of the element we want to animate, and x:100 is the variable which holds the value and specifies how our animation behaves. The manner in which our animation behaves depends on values defined in a tween. A **tween** is an animation property setter in GSAP which accepts animation parameters that define the target(element you want to animate), the duration, and elements you may want to animate at a new position. Methods in GSAP are made up of four tweens :

* gsap.to(): animates an element from its current state to its new state, which is the value in the tween.
* `gsap.from(): animates an element in a manner that makes it move from a position defined in the tween to its current state.
* gsap.fromTo(): which combines the functionality of the .to and .from methods. It gives the flexibility of defining both start and end positions.
* gsap.set(): is like gsap.to() only that there is no animation sequence. It goes to the value set in the tween immediately.

**Working Principles**

To create animations in Framer Motion, you start by wrapping the element in the <motion/> component, which means you have enabled that particular element for animation. For example, a <div>No Animation</div> element now becomes <motion.div>Animation</motion.div>. Then you can pass in animation properties which specify how you want the element to behave. Suppose we want to move this particular element to move 100 pixels to the right, we do this :

<motion.div animate={{ x: 100 }} />

In GSAP, there’s no need to wrap an element because we can target it directly. We can add animations to an element using the class or id of that element. For example, to add animation to an element with a class of "text-container", we do it like this:

gsap.to(".text-container", { x: 200 });

This means we’re telling GSAP to move all the elements targeted by the “text-container” class to a new position, which is 200 pixels away from its current state.

**Performance and Speed**

[Frame Per Second](https://en.wikipedia.org/wiki/Frame_rate) (FPS) measures the animation speed; it refers to the number of images (or frames) that is displayed per second on the screen. The higher the FPS, the smoother the motion is. Both Framer Motion and GSAP have a default frame rate of 60FPS, which means your animations will perform well and run smoothly.

The speed depends on several reasons which include the type of browser, the speed of your device, the number of animations being loaded into the Document Object Model (DOM), and your internet connection. You may want to test your animation across different browsers if you’re working on a large-scale application.

**Size and Optimization**

The total size of a library which contains all the assets needed to run optimally is called the bundle size. Bundle sizes differ across various libraries and it is normal to expect that a library with a smaller bundle size will run faster, so libraries and software packages go through **minification**. Minification is a process where extra or unnecessary lines of code are removed while maintaining its functionality so we can get the least possible bundle size of the library.

Both Framer Motion and GSAP are lightweight libraries and should run in an optimized manner irrespective of your RAM size. The size of Framer Motion when minified is about 119kb, while the size of GSAP when minified is about 69kb. That’s about a 50kb difference in bundle size. Under similar conditions, although negligible, GSAP will run faster since it weighs lesser.

**Learning Curve**

It is not easy to measure the learning curve for both libraries because it depends on your prior knowledge of React and JavaScript and how fast you grasp concepts based on both languages. We can say Framer Motion is built on the foundation of React and GSAP is built on the foundation of JavaScript.

So, if you’re looking to add animation to your React application, it’ll be easier for React developers to understand the concept. Framer Motion and GSAP provide explanatory and easy-to-use documentation to get you started even without prior knowledge of animation.

**Active Community**

In case you’re wondering why popularity or the number of active users in a community is a factor to consider, it becomes easier in fixing bugs you may run to. The higher the number of users in a community, the higher the chance they may have come across a bug and found a fix to it. It is faster to get a pool of answers to questions you may have about a particular library when there are many users.

At the time of this writing, Framer Motion has over 19,000 stars on Github with about 78 contributors while GSAP has over 17,000 stars on Github with 2 contributors. We can conclude Framer Motion has the more active community.

**Predicted Relevance**

Especially on a commercial basis, knowing if a language, tool or library will stand the test of time is important in acquiring the skill. We cannot make conclusions on this but we can make predictions given some metrics.

According to NPM, Framer Motion has weekly downloads of about 1.7 million while GSAP has 315k weekly downloads. With these numbers, we can say both libraries are relevant.

**Benefits and Shortcomings of Framer Motion and GSAP**

As great as Framer Motion and GSAP are in implementing web animations, there are also some shortcomings. We will be discussing them in this section.

**Benefits of Framer Motion**

* Being built on the foundation of React, it provides easy integration with your React application.
* The syntax is easier to understand and use. It also implements animation with fewer lines of code.
* With a highly active community, fixing bugs is easier.
* It provides a more advanced set of event listeners, this makes implementation easier for React developers and gives a better experience of animations.
* Being compatible with [Framer X](https://www.framer.com/), a design animation tool, Framer Motion bridges the gap between design and development by providing a seamless translation of animation sequences into codes.

**Shortcomings of Framer Motion**

* It has a higher bundle size, which makes it less optimal when it has to load heavy assets.
* Complex animations lag on low-end browsers.

**Benefits of GSAP**

* It provides a wide range of support for frameworks.
* It weighs less and runs faster than Framer Motion.
* It offers flexibility in creating complex animations.
* It handles animation responsiveness under the hood for desktop, laptop, and mobile while providing solutions for browser inconsistencies.

**Shortcomings of GSAP**

* Compared to Framer Motion, the syntax is not so easy to grasp.
* Implementing complex animations takes time and requires high skill.

**Installation/Setup**

In this section, we will cover the installation of both libraries and how we will set them up for usage in our projects. In both cases, we will be running the installation commands through our terminal.

**Framer Motion**

To install Framer Motion in your project, open your terminal and run the line of code:

npm install framer-motion  
npm start

This installs the packages from NPM. We run the start command after the installation of a package. After this, we configure it by importing it for use in our file. In your App.js, run the following line of code:

import React from "react";  
import { motion } from "framer-motion";

And you’re set, you can now use Framer Motion in your project.

**GSAP**

Installing GSAP and configuring it for use is like the Framer Motion installation. Open your terminal and run the installation command:

npm install gsap  
npm start

This installs the GSAP package for use and runs our start command. To configure your project, open your App.js file and import it.

import React from "react";  
import { gsap } from "gsap";

You’re now set to use GSAP for animations in your project.

**A Deeper Dive into Syntax**

A recap of what we have learnt so far in this article, we will be looking at a syntax explanation of both Framer Motion and GSAP using a sample code. We want to implement an animation where two elements, one from the left, and the other from the right, appear 3 seconds after you enter the viewport. Let’s see how we would do this in Framer Motion.

**Framer Motion**

import React from "react";  
import { motion } from "framer-motion";

const App = () => {  
 const leftAnimation = {  
 visible: { x: 0, transition: { duration: 3 } },  
 hidden: { x: "-100vw" },  
 }; const rightAnimation = {  
 visible: { x: 0, transition: { duration: 3 } },  
 hidden: { x: "100vw" },  
 }; return (  
 <motion.div>  
 <div initial="hidden" animate="visible" variants={leftAnimation}>  
 Left Animation  
 </div> <div initial="hidden" animate="visible" variants={rightAnimation}>  
 Right Animation  
 </div>  
 </motion.div>  
 );  
};

In this sample code snippet, we start by importing Framer Motion into our project. Then we create two object instances, leftAnimation and rightAnimation which contain our animation properties. In the leftAnimation object, we create two custom properties called visible and hidden. The visible class represents values when the element is in the viewport, while the hidden class represents values when the element is out of the viewport.

We do a similar implementation for the rightAnimation object, the only difference is that it has a value of 100vw for the visible value, which means it comes from the right as opposed to the -100vw for the leftAnimation object which comes from the left. In both, we apply a duration of 3 seconds, which represents the transition sequence.

In our motion.div element, we pass in three props, initial, animate, and variants. The initial prop represents the start position, that is when the element gets loaded in the viewport. The animate prop represents how we want our element to behave after it is loaded in the viewport, and the variants property is to specify the particular animation property to run.

Let’s see how we would do this in GSAP.

**GSAP**

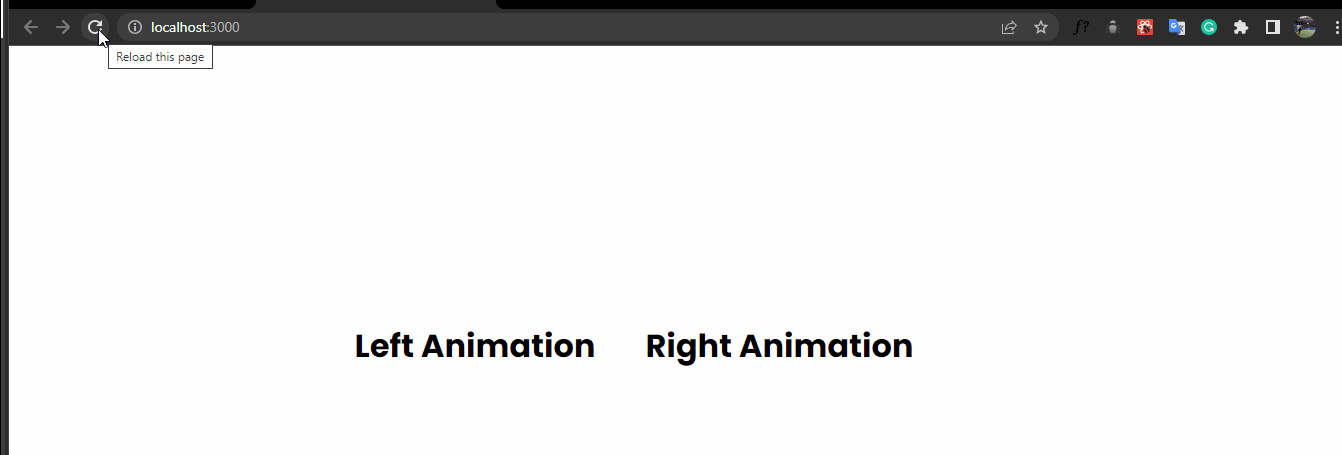
import { gsap } from "gsap";  
import { useRef, useEffect } from "react";

const Home = () => {  
 const leftAnimation = useRef();  
 const rightAnimation = useRef();  
 useEffect(() => {  
 gsap.fromTo(leftAnimation.current, { x: -100 }, { x: 0, duration: 3 });  
 gsap.fromTo(rightAnimation.current, { x: 100 }, { x: 0, duration: 3 });  
 }); return (  
 <div>  
 <h1 ref={leftAnimation}>Left Animation</h1> <h1 ref={rightAnimation}>Right Animation</h1>  
 </div>  
 );  
};

As we can see, this is a different syntax from Framer Motion. First, we have a major difference in importing two React hooks; useRef and useEffect. The useRef hook plays the role of the variants prop in Framer Motion by integrating the animation properties with the element, while the useEffect hook handles rendering the animation when the application is loaded.

We see that instead of defining the visible and hidden values as we did in Framer Motion, we made use of the gsap.FromTo()tween which lets us state the start and end positions for the leftAnimation and rightAnimation. Then we pass the ref as a prop which specifies the animation property we want that element to follow.

The result of both animation examples is the same. Here’s the result:



**Knowing the Perfect Use Case**

Choosing between Framer Motion and GSAP for your next project depends on the scale of the project, and the nature of animations you want to implement.

If you are new to animations in React and want to make basic animation sequences for your personal React project, Framer Motion is recommended since it is more user-friendly than GSAP.

If you want to animate more than HTML and CSS elements and looking to make more complex animations to [SVG](https://developer.mozilla.org/en-US/docs/Web/SVG), [Canvas](https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial), [Three.js](https://threejs.org/), or [WebGL](https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API), then GSAP is recommended.

Also, if you want an animation library that runs faster, GSAP is recommended since it weighs less and is more performant.

**Best Practices**

While the usage of animation may be fun, you need to aim for maximum optimization while at it. The following tips will help.

* On a commercial scale, only apply animations where necessary. You cannot tell what device the user loads the application on so you may want to make it as minimal as possible.
* Test on different browsers to ensure cross-browser compatibility.
* Use moderate animations on a page. Too many objects moving on the screen may deplete the user’s experience.

**Conclusion**

In this article, you have learnt about [Framer Motion](https://www.framer.com/motion/) and [GSAP](https://greensock.com/gsap/), how they compare, the benefits and shortcomings, and knowing the perfect use case for both libraries. It is difficult to choose a clear winner for this comparison as it hugely depends on the use case. You can now decide whether to use Framer Motion or GSAP for your next article based on what you have read. Have fun with your animations!

*Originally published at*[*https://semaphoreci.com*](https://semaphoreci.com/blog/react-framer-motion-gsap)*on October 24, 2023.*

[React](https://medium.com/tag/react?source=post_page-----dd6208440281---------------react-----------------)

[Reactjs](https://medium.com/tag/reactjs?source=post_page-----dd6208440281---------------reactjs-----------------)

[Framer Motion](https://medium.com/tag/framer-motion?source=post_page-----dd6208440281---------------framer_motion-----------------)

[Gsap](https://medium.com/tag/gsap?source=post_page-----dd6208440281---------------gsap-----------------)

[Web Animation](https://medium.com/tag/web-animation?source=post_page-----dd6208440281---------------web_animation-----------------)