{JS}
MASTERY

The Ultimate GSAP Guide

Created by **JS Mastery**

Visit ***jsmastery.pro*** for more

What’s in the guide?

Welcome to the Ultimate GSAP Guide! 👋

Whether you're a beginner dipping your toes into the world of web animation or a seasoned developer looking to elevate your skills, you've arrived at the perfect destination.

This comprehensive guide is your roadmap to mastering the GSAP animation.

From foundational principles to advanced techniques, this guide will take you on a journey through: **step-by-step roadmap, insider tips, examples, and more**.

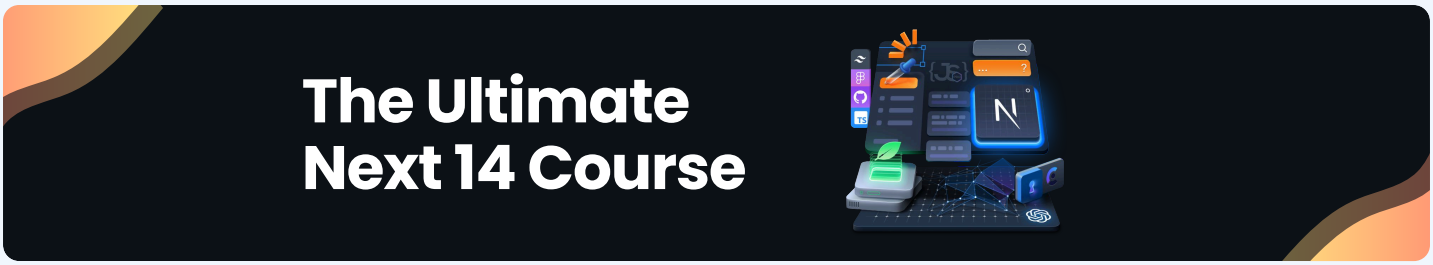
So, dive in and let's begin on an exciting journey to unleash the full potential of GSAP.

Happy animating and coding! 🎨✨

...before you go

While the GSAP Guide equips you with invaluable skills in web animation, imagine taking your skills to the next level by seamlessly integrating these techniques with Next.js.

If you're eager to dive deep into something this specific and build substantial projects, our **special course on Next.js** has got you covered.



It teaches everything from the ground up, providing a hands-on experience that goes beyond just GSAP.

Check it out and take your skills to the next level 🚀

What is GSAP?

GSAP is a framework-agnostic JavaScript animation library that turns devs into animation superheroes.

Build high-performance animations that work in every major browser. Animate CSS, SVG, canvas, React, Vue, WebGL, colors, strings, motion paths, generic objects...anything JavaScript can touch!

GSAP is unmatched in **delivering advanced sequencing**, **reliability**, and **precise control** for animations on over 12 million websites.

It effortlessly handles browser inconsistencies, ensuring your animations work seamlessly.

GSAP is a fast property manipulator, updating values over time with precision, and it's up to 20 times faster than jQuery!

GSAP Setup

To use GSAP, you have multiple options for integrating it into your project.

1. NPM

One common approach is to install **GSAP via npm**, which allows you to manage dependencies efficiently within your project's ecosystem.

This method is particularly useful for larger projects or those using modern build tools like Webpack or Parcel.

2. CDN

Alternatively, you can opt for the quick and easy method of including GSAP directly via a **CDN** (Content Delivery Network) link in your HTML file.

This approach is convenient for smaller projects or when you want to quickly prototype an idea.

GSAP Setup

3. React

If you're working with React, you have the option to use the **@gsap/react library**, which provides seamless integration of GSAP with React components.

This allows you to harness the power of GSAP within your React applications while leveraging the component-based architecture of React.

Overall, the flexibility of GSAP's integration options ensures that you can easily incorporate it into your preferred development workflow, whether you're building a traditional website, a single-page application, or a complex web application with React.

GSAP Setup

3. React

If you're working with React, you have the option to use the **@gsap/react library**, which provides seamless integration of GSAP with React components.

This allows you to harness the power of GSAP within your React applications while leveraging the component-based architecture of React.

Overall, the flexibility of GSAP's integration options ensures that you can easily incorporate it into your preferred development workflow, whether you're building a traditional website, a single-page application, or a complex web application with React.

CDN Setup

To get started with GSAP, we first need to add the GSAP library to our HTML file. You can do this by grabbing the **CDN link** to the GSAP library.

Here's how it will look, but with the full link.

```
<script
  src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.12....
  integrity="sha512-EZI2cBcGPnmR89wTgVnN3602Yyi7muWo8say...
  crossorigin="anonymous"
  referrerpolicy="no-referrer"
></script>
```

Make sure to place this script tag before the script tag containing your local JS file. This ensures that your local file has access to the GSAP library and its functionality.

Also, if you're using any GSAP plugins, ensure that their script tags are placed after the GSAP script tag.

GSAP Basics

GSAP is incredibly flexible; you can use it anywhere you like, and it has zero dependencies.

If you've used any version of GSAP in the past couple of years or are familiar with tools like **TweenLite, TweenMax, TimelineLite, and TimelineMax**, you'll notice that in the new version, GSAP 3.0, they have all been replaced by the **GSAP object**.

Think of the GSAP object as the main hub for everything in GSAP.

It's like a toolbox filled with all the tools you need to create and control Tweens and Timelines, which are the main things you'll be working with in GSAP.

To really get the hang of GSAP, it's important to understand Tweens and Timelines:

GSAP Basics

Understanding Tween

Think of a Tween as the magic that makes things move smoothly—it's like a super-efficient property setter. You give it targets (the objects you want to animate), set a duration, and specify which properties you want to change.

Then, as the Tween progresses, it calculates and applies the property values at each step, creating seamless animation.

Here are some common methods for creating a Tween:

- **`gsap.to()`**
- **`gsap.from()`**
- **`gsap.fromTo()`**

GSAP Basics

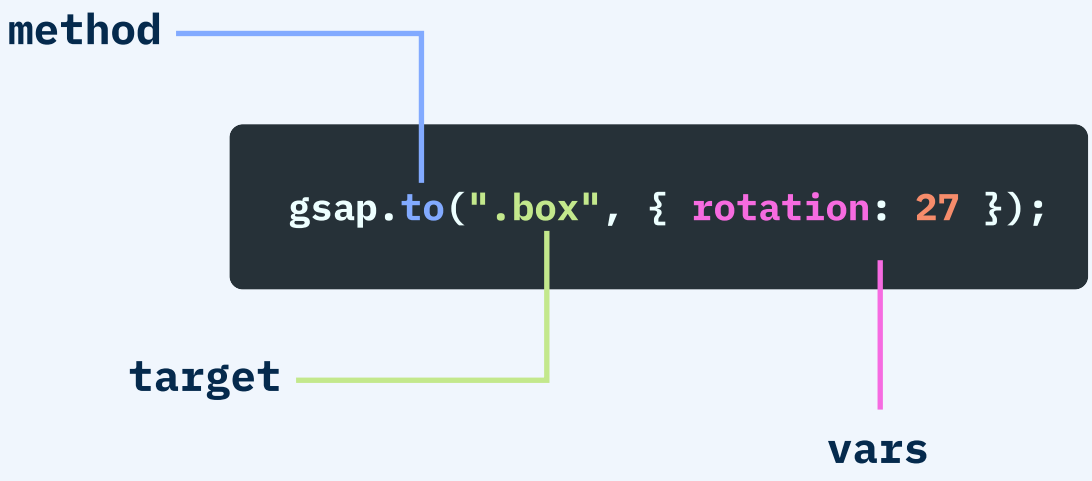
For simple animations (no fancy sequencing), the methods above are all you need! For example:

```
gsap.to(".box", { rotation: 27, x: 100, duration: 1 });
```

rotate and move elements with a class of "box" ("x" is a shortcut for a translateX() transform) over the duration of 1 second.

Basic sequencing can be achieved by utilizing the **delay** special property.

Let's take a closer look at the syntax.



GSAP Basics

We've got a method, a target and a vars object which all contain information about the animation

The method(s)

There are four types of tweens:

`gsap.to()`

his is the most common type of tween. A `to()` tween starts at the element's current state and **animates "to" the values defined in the tween.**

`gsap.from()`

This is similar to a backwards `to()` tween. It animates "from" the values defined in the tween and ends at the element's current state.

GSAP Basics

`gsap.fromTo()`

With this method, you define both the starting and ending values for the tween.

`gsap.set()`

This method immediately sets properties without any animation. It's essentially a zero-duration `to()` tween.

These methods provide flexibility in how you create tweens and allow you to achieve various effects in your animations.

GSAP Basics

The target(s)

In GSAP, you need to specify what you want to animate, known as the target or targets. GSAP internally utilizes **`document.querySelectorAll()`**, allowing you to use selector text like ".class" or "#id" for HTML or SVG targets. Alternatively, you can pass in a variable or an Array.

Here are some examples:

```
  // Using a class or ID
★ gsap.to(".box", { x: 500 });

  // Using a complex CSS selector
★ gsap.to("section > .box", { x: 900 });

  // Using a variable
  let box = document.querySelector(".box");
★ gsap.to(box, { x: 200 });

  // Using an Array of elements
  let square = document.querySelector(".square");
  let circle = document.querySelector(".circle");
★ gsap.to([square, circle], { x: 200 });
```

GSAP Basics

These examples showcase how you can specify different targets for your animations in GSAP, allowing you to animate various elements with different properties and values.

The variables

The vars object holds all the details about the animation. It includes properties you wish to animate, as well as special properties that control the animation's behavior, such as **`duration`**, or **`repeat`**.

Here's how you might use it in a **`gsap.to()`** tween:

```
gsap.to(target, { // This is the vars object
  // It contains properties to animate
  x: 200,
  rotation: 360,
  // Along with special properties
  duration: 2,
});
```

GSAP Basics

What properties can we animate?

With GSAP, you have the flexibility to animate **almost anything you can imagine**. There's no predefined list of properties you can animate because GSAP is incredibly versatile.

You can animate CSS properties like width, height, color, and font-size, as well as custom object properties. GSAP even allows you to animate CSS variables and complex strings!

While you can animate virtually any property, some of the most commonly animated properties include transforms (such as **translate, rotate, scale**), opacity, and position. These properties are frequently used to create smooth and visually appealing animations across various elements on your webpage.

GSAP Basics

However, for more advanced sequencing and complex choreography, **Timelines** are the way to go. They make the process much easier & more intuitive.

Understanding Timeline

A Timeline serves as a container for Tweens, making it the ultimate tool for sequencing animations. With a Timeline, you have the power to position animations in time exactly where you want them.

You can effortlessly control the entire sequence using methods like **`pause()`, `play()`, `progress()`, `reverse()`**, and **`timeScale()`**, among others.

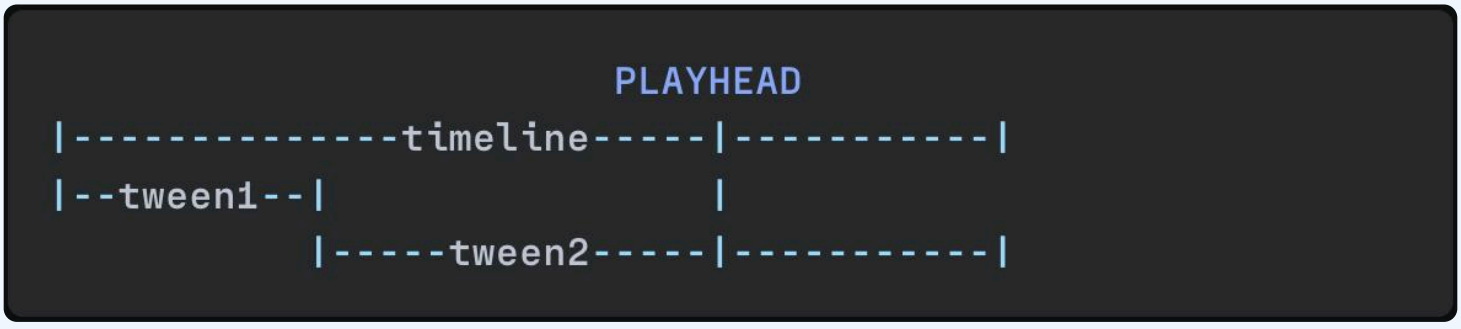
The beauty of Timelines is that you can create as many as you need, and even nest them, which is great for organizing your animation code into manageable modules.

GSAP Basics

Here's the cool part: every animation (whether it's a Tween or another Timeline) is placed onto a parent timeline (which is usually the globalTimeline by default).

This means that when you move a Timeline's playhead, it cascades down through its children, ensuring that all the animations stay perfectly synchronized.

It's important to note that a Timeline is all about grouping & coordinating animations in time, it doesn't actually set properties on targets like Tweens do.



GSAP Basics

To create a Timeline in GSAP, you simply use the method:

```
gsap.timeline()
```

With GSAP's API, you have the power to control virtually anything on-the-fly.

You can manipulate the playhead position, adjust the **startTime** of any child, play, pause, or reverse animations, alter the timeScale, and much more.

GSAP Basics

Sequencing things in a Timeline

To create a Timeline, you first initialize it like this:

```
var tl = gsap.timeline();
```

Then, you can add tweens using one of the convenience methods like `to()`, `from()`, or `fromTo()`:

```
tl.to(".box", { duration: 2, x: 100, opacity: 0.5 });
```

You can repeat this process as many times as needed. Notice that we're calling .`to()` on the timeline instance (in this case, the variable `tl`), not on the gsap object.

This creates a tween and immediately adds it to that specific Timeline.

GSAP Basics

By default, the animations will be sequenced one-after-the-other. You can even use method chaining to simplify your code:

```
// Sequenced one-after-the-other
tl.to(".box1", { duration: 2, x: 100 })
  .to(".box2", { duration: 1, y: 200 })
  .to(".box3", { duration: 3, rotation: 360 });
```

It's worth noting that while you could create individual tween instances with **`gsap.to()`** and then use **`timeline.add()`** to add each one, it's much easier to call `.to()`, `.from()`, or `.fromTo()` directly on the Timeline instance.

This approach accomplishes the same thing in fewer steps, keeping your code clean and concise.

React GSAP

Why choose GSAP with React?

While React-specific libraries provide a declarative approach to animation, GSAP offers unique advantages.

Animating imperatively with GSAP empowers you with greater **control**, **flexibility**, and **creativity**.

Whether you're animating DOM elements, SVGs, three.js, canvas, or WebGL, GSAP allows you to unleash your imagination without limits.

What sets GSAP apart is its framework-agnostic nature. This means that your animation skills seamlessly transfer to any project, be it Vanilla JS, React, Vue, Angular, or Webflow.

React GSAP

With GSAP, you won't need to switch between different libraries for different projects. It becomes your reliable toolkit, ensuring consistency and efficiency across all your goals.

React Setup

Setup for experimenting with React and GSAP.

```
npm create vite@latest my-react-app -- --template react
```

Once the project is set up we can install GSAP and the special GSAP/React package through npm,

```
# Install the GSAP library
npm install gsap

# Install the GSAP React package
npm install @gsap/react

# Start the project
npm start
```

React Setup

Then import it into the app.

```
import { useRef } from "react";

import gsap from "gsap"; // <-- import GSAP
import { useGSAP } from "@gsap/react";

gsap.registerPlugin(useGSAP);

export default function App() {
  const container = useRef();

  useGSAP(
    () => {
      // gsap code here...
      gsap.to(".box", { rotation: 180 });
    },
    { scope: container }
  );

  return (
    <div ref={container} className="app">
      <div className="box">Hello</div>
    </div>
  );
}
```

React GSAP

What is that useGSAP() Hook?

GSAP works seamlessly with any JavaScript framework, without needing any special adjustments.

However, this hook is specifically designed to smooth out some React-specific challenges, letting you focus on the fun parts.

`useGSAP()` is a convenient replacement for **`useEffect()`** or **`useLayoutEffect()`**. It automatically manages cleanup using **`gsap.context()`**. Proper cleanup is important in React, and using Context makes it easy.

Simply import the **`useGSAP()`** hook from **`@gsap/react`**, and you're ready to roll!

Any GSAP animations, ScrollTriggers, Draggables, or SplitText instances created with the **`useGSAP()`** hook

React GSAP

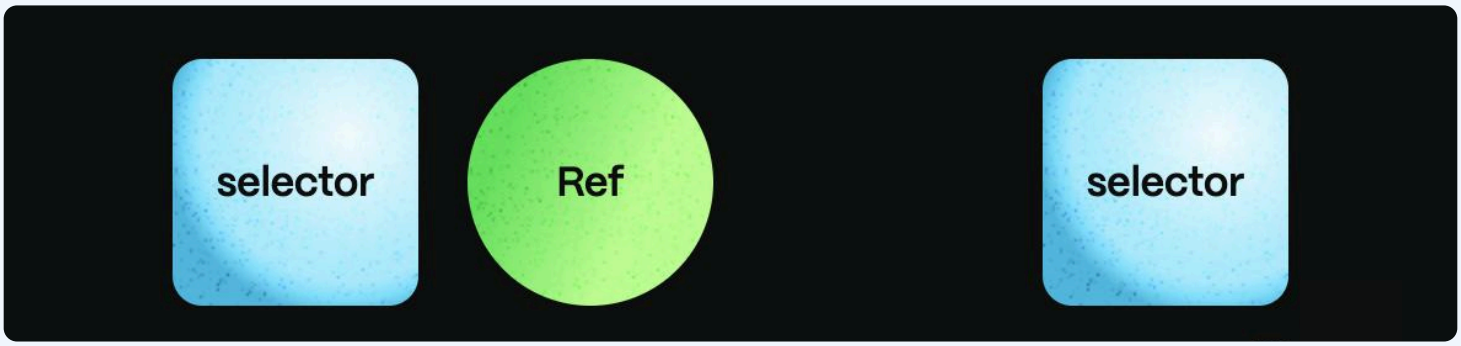
will be cleaned up automatically when the component is unmounted and the hook is removed.

```
import { useRef } from "react";
import gsap from "gsap";
import { useGSAP } from "@gsap/react";

gsap.registerPlugin(useGSAP);

const container = useRef();

useGSAP(
  () => {
    // gsap code here...
    gsap.to(".box", { x: 360 }); // <-- automatically reverted
  },
  { scope: container }
); // <-- scope is for selector text (optional)
```



React GSAP

Why is cleanup is crutial?

Cleanup is crucial in animation, particularly with frameworks like React. In React 18, there's a default strict mode setting that can cause Effects to run twice locally. This double execution can result in duplicate or conflicting animations, as well as logic issues with tweens if not properly reverted.

The **`useGSAP()`** hook adheres to React's recommended practices for animation cleanup, ensuring that your animations are handled correctly and efficiently.

You can safely use this hook in **Next.js** or similar server-side rendering environments.

GSAP Easing

The GSAP official documentation defines easing as the primary method to adjust the timing of your Tweens.

Easing determines how an object transitions between positions at different points during an animation.

It controls the rate of change of animation and sets the style of an object's movement.

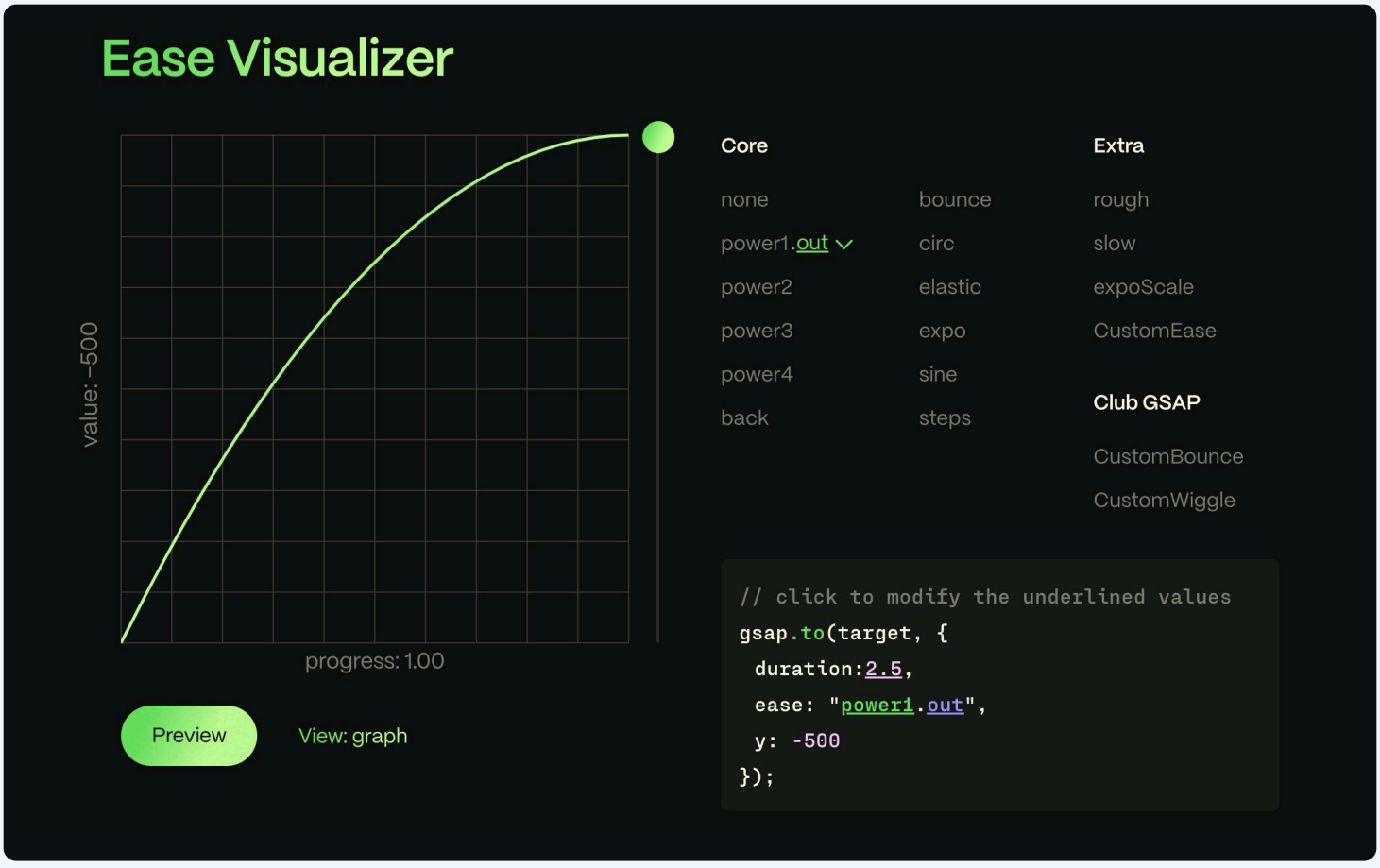
GSAP offers various types of eases and options to provide you with greater control over your animation behavior.

Additionally, GSAP provides an **Ease Visualizer tool** to assist in selecting preferred ease settings.

There are three main types of eases, each operating differently:

GSAP Easing

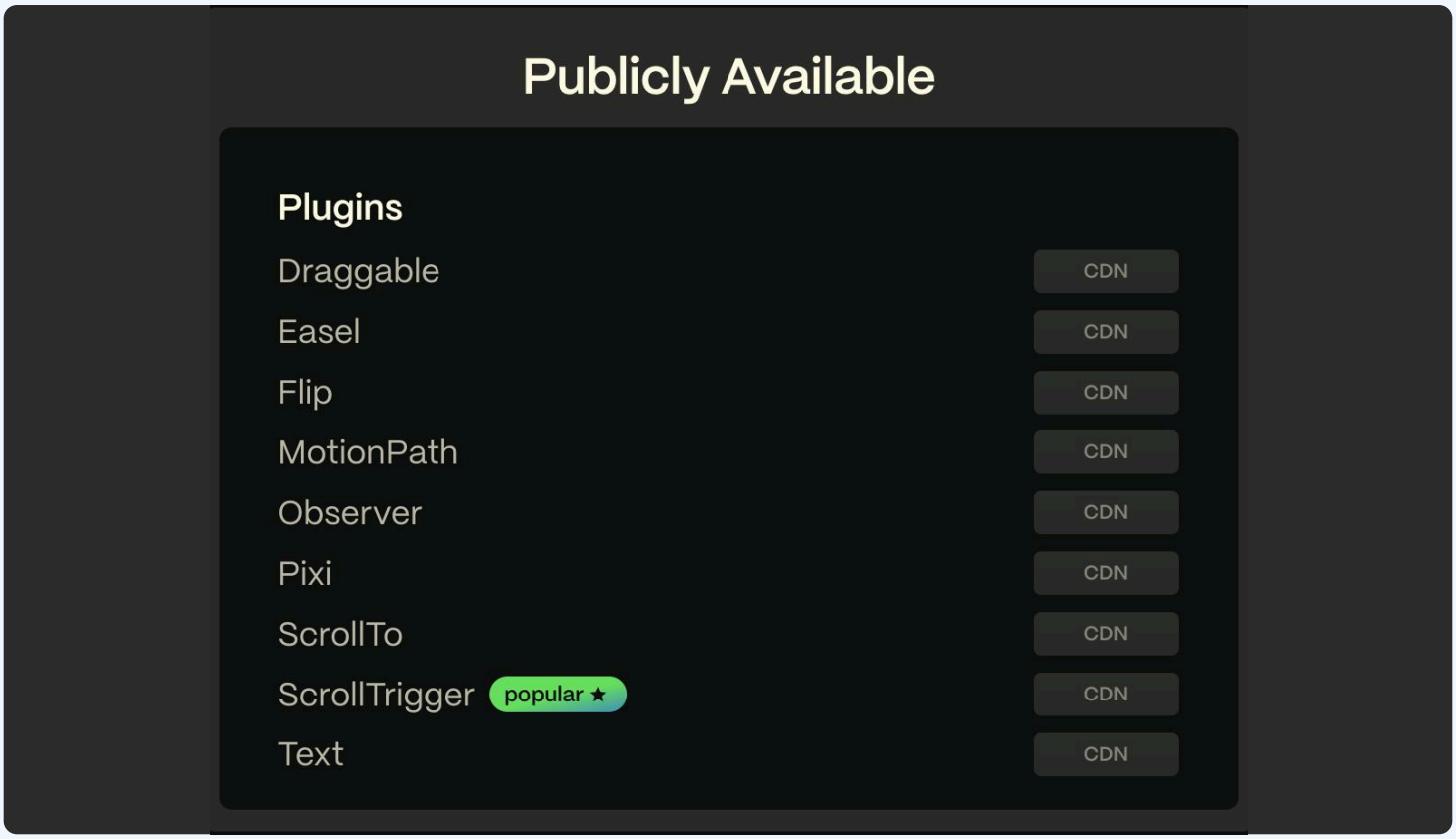
- **`in()`**: Animation begins slowly and accelerates towards the end.
- **`out()`**: Animation starts quickly and decelerates towards the end.
- **`inOut()`**: Animation starts slowly, accelerates in the middle, and then decelerates towards the end.



GSAP Plugins

A plugin extends the capabilities of GSAP's core functionality. By using plugins, the GSAP core remains lightweight while providing additional features that can be added as needed.

This modular approach allows you to customize your GSAP setup by including only the features that are required for your specific project, keeping your codebase efficient and focused.



GSAP Plugins

To install or load a plugin, you have several options. Plugins are JavaScript files, similar to the core GSAP library. You can install them using **script tags, npm, yarn**, or even a **tgz file**.

Here's how you can register a plugin

```
//list as many as you'd like
gsap.registerPlugin(MotionPathPlugin, ScrollTrigger);
```

Remember, you need to load the plugin file before registering it. Registering a plugin with GSAP ensures seamless integration and avoids issues with build tools and bundlers.

You only need to register a plugin once before using it. It's okay to register the same plugin multiple times, but it doesn't offer any additional benefits.

GSAP Plugins

ScrollTrigger Plugin Example

ScrollTrigger empowers users to effortlessly create nice scroll-based animations with minimal code.

It offers unparalleled flexibility, allowing you to easily implement scroll-triggered effects such as scrubbing, pinning, snapping, or triggering any scroll-related action, regardless of whether it involves animation or not.

Overall, ScrollTrigger provides a flexible and intuitive way to create engaging scroll-based animations and interactions, making it a valuable tool for enhancing user experiences on the web.

GSAP Plugins

Here's a simple example using ScrollTrigger to animate a box:

```
gsap.to(".box", {
  scrollTrigger: ".box",
  x: 500,
});
```

In this example, when the element with the class "box" enters the viewport, GSAP will animate it by moving it 500 pixels to the right.

ScrollTriggers can perform an actions on an animation (play, pause, resume, restart, reverse, complete, reset) when entering/leaving the defined area or link it directly to the scrollbar so that it acts like a scrubber (`scrub: true`).

GSAP Examples

ScrollTrigger Showcase

https://codepen.io/collection/DkvGzg

GSAP Animation Websites

https://www.awwwards.com/websites/?aw…

Spilt Text Examples

https://codepen.io/collection/XMoeqD

Filer

https://www.filer.dev/3d-models/1

GSAP Examples

Popular Demos

https://gsap.com/demos/

MorphSVG Showcase

https://codepen.io/collection/naMaNQ

DrawSVG Showcase

https://codepen.io/collection/DYmKKD

GSAP UI Design Examples

https://codemyui.com/tag/gsap/

Talented coders that feature a ton of GSAP

Cassie Evans

https://codepen.io/cassie-codes

Blake Bowen

https://codepen.io/osublake

Craig Roblewsky

https://codepen.io/PointC/

Darin Senneff

https://codepen.io/dsenneff

Talented coders that feature a ton of GSAP

Chris Gannon

https://codepen.io/chrisgannon

Carl Schooff

https://codepen.io/snorkltv

Pete Barr

https://codepen.io/petebarr

Steve Gardner

https://codepen.io/ste-vg

Talented coders that feature a ton of GSAP

Ryan Mulligan

https://codepen.io/hexagoncircle

Cameron Knight

https://codepen.io/cameronknight

Tom Miller

https://codepen.io/creativeocean

Tips

The most important tip

Don't go overboard with animations. Not everything needs to move, and too many animations can make things confusing for users. Focus on animating things that make the user experience better.

Properties like transformations (such as translate, rotate, scale) and opacity typically offer smooth performance across browsers.

On the other hand, properties like filter or boxShadow can be computationally expensive and may cause performance issues, especially on low-end devices.

Ensuring that your animations perform well on low-end devices is crucial for delivering a consistent user experience across different devices and platforms.

Tips

Easing functions

Experiment with different easing functions to add smoothness and character to your animations. GSAP has options like "**easeIn**" (starts slow and speeds up), "**easeOut**" (starts fast and slows down), and others you can experiment with.

Explore plugins

GSAP has extra tools you can add to make your animations even cooler. For example, SplitText lets you animate text in unique ways, and ScrollTrigger lets you create animations that happen when you scroll down a webpage. These plugins can add extra flair to your animations without much extra work.

Tips

Preload assets

Preload any assets, such as images or videos, that will be used in your animations to ensure smoother playback and prevent delays.

Practice with examples

One of the best ways to learn is by trying things out yourself. Look for examples of GSAP animations online and try to recreate them in your own projects.

Experimenting with different techniques will help you get better at creating your own unique animations.

Tips

Consider accessibility

Some people might have trouble with certain types of animations, like ones that flash or move around a lot.

Avoid animations that may cause discomfort or motion sickness, provide alternative content for screen readers, and allow users to pause or disable animations if needed.

Additionally, ensure that important information is still accessible even if someone can't see the animations.

Ensure that your animations are accessible to all users, including those with disabilities.

Tips

Test across devices

While GSAP is excellent in performance and provides a robust framework for creating animations, it's essential to remember that different devices have varying capabilities and screen sizes.

Therefore, thorough testing is necessary to guarantee a consistent and optimal UX across the board.

Not all devices are the same, so it's important to make sure your animations work well on all of them. Test your animations on different phones, tablets, and computers to make sure they look good and run smoothly everywhere.

Conclusion

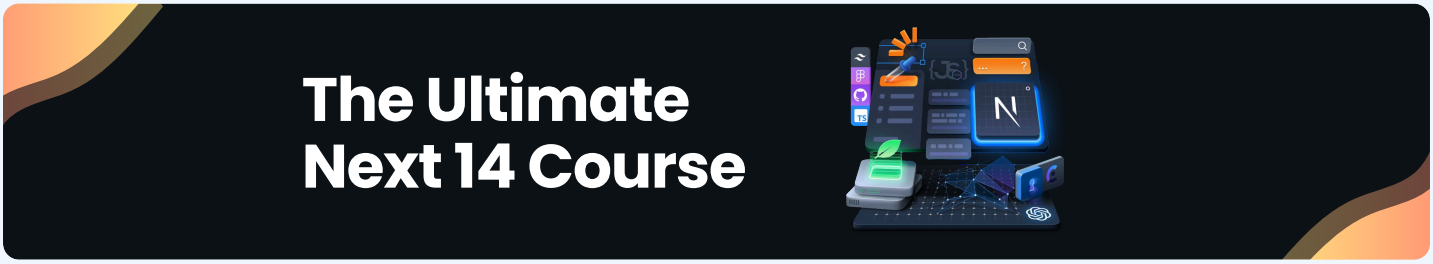
The GSAP library is undeniably fascinating, and I'm glad I could assist you in understanding how to use it while adhering to good practices and tips. 🙌

For a better understanding and access to more features, I suggest reading the **documentation** and experimenting with different animation techniques.

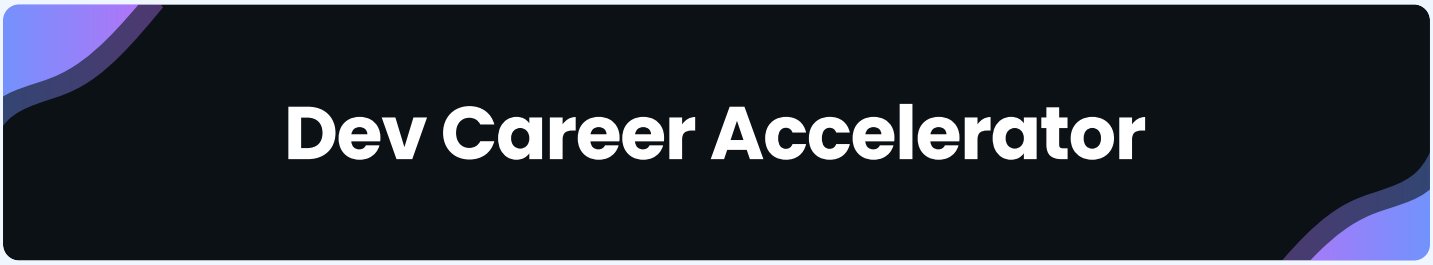
The End

Congratulations on reaching the end of our guide! But hey, learning doesn't have to stop here.

If you're eager to dive deep into something this specific and build substantial projects, our **special course on Next.js** has got you covered.



If you're craving a more personalized learning experience with the guidance of expert mentors, we have something for you — [Dev Career Accelerator](#).



If this sounds like something you need, then don't stop yourself from leveling up your skills from junior to senior.

Keep the learning momentum going. Cheers! 🚀