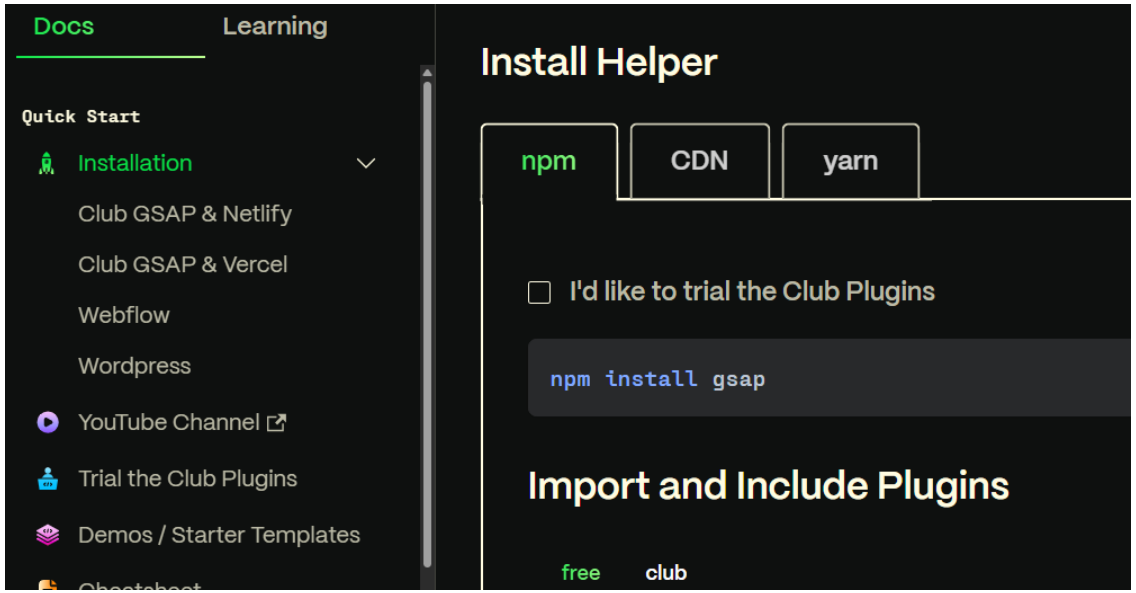
GSAP Introduction

Utilizing JS animation features using GSAP Animation tools

GSAP Installation

There are 4 ways to utilize GSAP into your project:

- ZIP File
- CDN (Content Delivery Network)
- NPM (Package manager for JavaScript Programs)



The preferred way to use GSAP would be the Content delivery network, it loads fast and prevents delays and errors in your project that can happen through file organization/typos/etc.

NPM and CDN

NPM:

NPM is the world's largest Software Registry.

The registry contains over 800,000 code packages.

Open-source developers use NPM to share software.

Using the command line terminal inside VSCode, we can access NPM to download packages remotely directly into our project folders.

CDN (Content Delivery Network):
A Content Delivery Network (CDN) is a version

of the jQuery library in the “Cloud.” Using a CDN saves

us the hassle of downloading/uploading the file ourselves.

JS folder organization

Using the CDN method we covered in our last lecture; we will include the complied code within our projects to use GSAP animation triggers within our project.

Once you've attached GSAP to your project file using the script tag, You should create a JS folder in your project to hold your JavaScript Program (Project Folder-> js -> gsap.js). You can also place your JavaScript directly into the "index.html" inside a <script> tag. We will be making a separate "js" folder to create better organization of our project. Just like how we can add CSS to the "index.html" using the <style> tag but we choose to create a separate stylesheet.

Using GSAP to trigger animations and interactions

Setting up animations in your project:

The image I want to animate is placed in a tag in the “index.html” file. I then configure the animation inside of my “gsap.js” file.

```
<body>
    <div>
        <img class="pic" src="img/mowalola.svg">
    </div>


    <script src="https://cdn.jsdelivr.net/npm/gsap@3.12.7/dist/gsap.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/gsap@3.12.7/dist/ScrollTrigger.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/gsap@3.12.7/dist/ScrollToPlugin.min.js"></script>
    <script src="js/gsap.js"></script>
</body>
```

In this screenshot, you can see that my <script> tags are at the bottom of my <body> tag. The visual elements like images, divs, and other assets will exist above the <script> tag. BOTH should be inside the <body> tag in your “index.html”.

The class applied to the image (class=“pic”), will be used in our JavaScript file to connect the image to the GSAP animation.

Animate using GSAP Functions

There are a wide variety of callback methods used in GSAP to trigger an animation. Just like how we can skew, distort, scale and fade our objects using CSS animation properties. We can use GSAP's premade animation functions to create "tweens".

Are animations are referred to as tweens because it is animated between states.

To begin our animation journey, we will be focusing on gsap.to(), gsap.from(), and gsap.fromTo().

```
gsap.to(".pic", { duration: 2, x: 600});
// move elements with a class of "pic" on the x-axis by 600px
// ("x" is a shortcut for a translateX() transform) over the course of 1 second.

gsap.from(".pic2", { opacity: 0, y: 100, duration: 1 });
// animate ".pic2" from an opacity of 0 and a y position of 100 (like transform: translateY(100px))
// to the current values (an opacity of 1 and y position of 0).

gsap.fromTo(".box", { opacity: 0 }, { opacity: 0.5, duration: 1 });
//animate ".box" from an opacity of 0 to an opacity of 0.5
//fromTo allows you to set the starting and ending values of the animated object
```

Animation Configuration

```
gsap.to(".pic", { duration: 2, x: 600});
```

When GSAP see's ".pic" it uses the browsers query selector all method to find elements that match this class name. The information inside the curly brackets {} is referred to as the configuration object with destination values for any properties we want to animate.

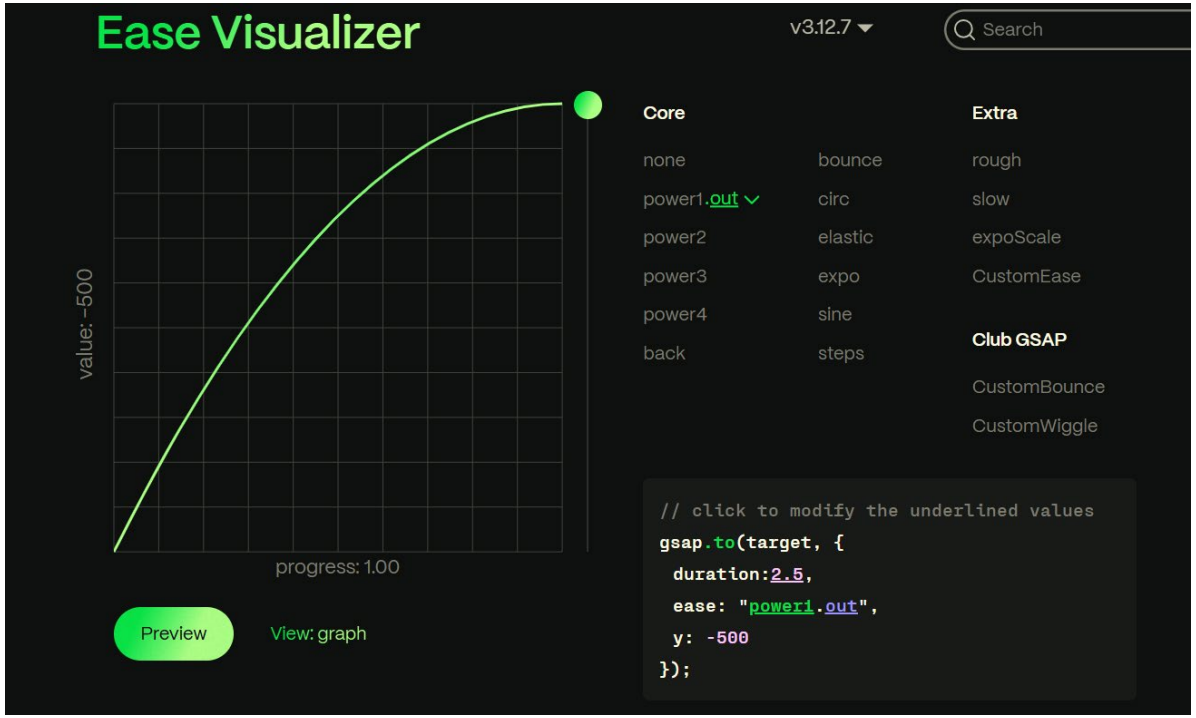
We can also layer values in a single configuration in this callback,

ex:

```
gsap.to(".pic", { duration: 2, x: 600, opacity: 0, borderRadius: "20%", border: "5px solid black"});
```

GSAP Ease Visualizer

When creating tween animations, easing is a convenient way to change the timing of your tweens. There are a list of eases to try out available to you on GSAP's resources page. There is also an Ease Visualizer that helps you understand how each function works and what would fit the needs of your current animation.



Control Methods

There are functions that can manipulate the playback of our tweens.

By default, GSAP animations play automatically on page load. You can use control methods to have more control of your animation, like having an animation play, repeat, or reverse based on user interaction such as button click.

Control methods work for tweens and also timelines.

You can find a full list on the GSAP Docs page listed under methods.

```
<button id="play">play()</button>
<button id="pause">pause()</button>
<button id="resume">resume()</button>
<button id="reverse">reverse()</button>
<button id="restart">restart()</button>
```

```
document.querySelector("#play").onclick = () =>
tween.play();
document.querySelector("#pause").onclick = () =>
tween.pause();
document.querySelector("#resume").onclick = () =>
tween.resume();
document.querySelector("#reverse").onclick = () =>
tween.reverse();
document.querySelector("#restart").onclick = () =>
tween.restart();
```

Sequencing in GSAP

Since our animations are automatically set to start on page load, we can create a timeline visualizer and add sequencing to our GSAP code to alter timing, delays, and transitions between our animations.

The timeline is a container for our animated objects or “tweens”. By placing our “tweens” under the same sequence, we can control them together instead of manipulating the load time of each individual object in relation to each others animation runtime.

The most important use of sequencing using a timeline allows our tweens to automatically adjust their time and delay based on the previous animation.

Sequencing in GSAP cont.

Creating a timeline in GSAP:

Add the timeline to your project a variable at the beginning of your JS file.

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

When using the timeline variable, you will change the “gsap.to()” animator to “tl.to()”. All objects that use the “tl” property will be included in this new timeline in order of appearance in our set up.

Ex:

```
tl.to(".pic", { duration: 2, x: 600, opacity: 0});
tl.to(".pic2", { duration: 4, x: 700, opacity: 0});
```