Welcome!

So glad you signed up for this course. I think you're really going to like what I have to show you. I know you can learn a lot just from watching the videos but I really want you to get your hands dirty.

This document contains some top-level notes and brief instructions for some practice exercises. I urge you to type as much as possible and NOT copy and paste code. You will need to know how to type this stuff and taking the time to do it right at the beginning will pay off.

Also, coding is about experimentation and play. Please tweak the code, play with the values, add extra properties and see what happens! Everything here is just provided as a starting point.

If you have any questions about the course just add a comment in the course community section.

This is the first release of this course, so please forgive any errors or hiccups. If you see something that needs fixing please let me know. carl@snorkl.tv

Happy Animating!

Carl

Setting Up Dev Environment

For this course all your work will be done on CodePen. CodePen makes it super easy to create and share web demos.

After watching the GSAP + CodePen video please follow the steps below to create a basic GSAP demo with a run button.

Create GSAP + CodePen Hello World

1. Visit CodePen.io and create a free account.
2. Visit GreenSock.com homepage
3. Copy the CDN URL displayed in the Get GSAP section

4. Visit CodePen.io and click “Start Coding” button.
5. In the HTML panel type **`<h1>hello world!</h1>`**
6. Click on the little gear icon in the JS panel to go to the JS settings.
7. Paste the CDN URL you copied from GreenSock.com into the field for external scripts.
8. In the JS panel type the code: **`gsap.to("h1", {x:400});`**
9. Click on the Settings button in the main nav.
10. Click on the Behavior tab.
11. Under Auto-Updating Preview make sure the checkbox next to Enabled is **de-selected**.
12. Click the Close button.

Basic Tween

Create a tween using the gsap.to() method.

The basic syntax for a to() tween is as follows:

```
gsap.to(".fred", {x:400}); // animates the element with a class of “fred” to an x position of 400.
```

If you do not specify a duration, gsap will use the default which is 0.5 seconds (500ms).

You can change the default duration using:

```
gsap.defaults({
  duration: 1
});
```

Behind the scenes gsap changes the target’s inline style during the animation.

For best performance animate CSS Transform values and opacity:

1. x
2. y
3. rotation
4. rotationX
5. rotationY

6. skewX and skewY
7. scaleX, scaleY, or just scale

GSAP can animate any numeric property you throw at it.

1. width and height
2. backgroundColor *hyphenated values need to be camelCase
3. color
4. padding
5. left and top (must set position to relative, absolute, or fixed)
6. vh and vw

Changing values that are not CSS Transforms or opacity can cause the browser to re-do its layout of the page which in extreme situations can hinder performance. For a few tweens, it's not the end of the world as some purists make it out to be.

Basic Tween Exercise

1: Open: https://codepen.io/snorkltv/pen/qBBxPme

2: Type the following code:

```
gsap.to(".fred", {x:700, y:400, scale:3, rotation:360,
duration:3});
```

Links and More

gsap.to() docs: https://greensock.com/docs/v3/GSAP/gsap.to()
gsap defaults docs: https://greensock.com/docs/v3/GSAP/gsap.defaults()

from() and fromTo()

gsap.from() animates from the values you specify to the object's natural values.

To animate from x and y values of 400, use:

```
gsap.from(".fred", {x:400, y:400});
```

gsap.fromTo() animates **from** the values you specify **to** the values you specify.

gsap.fromTo(".fred", {x:400, y:400}, {x:200, y:200});

For best results make sure the **from vars** and **to vars** have the same properties.

fromTo() Exercise

1: Open: https://codepen.io/snorkltv/pen/qBBxPme

2: Type the following code:

```
gsap.fromTo(".fred",
            {x:700, y:400, opacity:0},
            {x:400, y:200, scale:3, opacity:1, duration:3});
```

3: Experiment with other properties

Links and More

gsap.from() docs: https://greensock.com/docs/v3/GSAP/gsap.from()
gsap.fromTo() docs: https://greensock.com/docs/v3/GSAP/gsap.fromTo()

Special Properties: Delay and Repeat

Special properties define how the animation should run and what it should do. Special properties are not animated.

delay: how much time should transpire before animation begins
repeat: how many times the animation should repeat
yoyo: when set to true the animation will play back and forth
repeatDelay: how much time should transpire between each repeat

An animation will repeat indefinitely if you set repeat:-1

Practice

1: Open: https://codepen.io/snorkltv/pen/qBBxPme

2: Type the following code:

```
gsap.to(".fred", {x:300, repeat:-1, yoyo:true, repeatDelay:1});
```

Special Property: Ease and Using the Ease Visualizer

An ease controls the rate of change as your animation plays.

In simple uses an ease will control whether your animation slows down or speeds up.

An ease can be applied on the way out (default), on the way in, or both directions.

The steeper the curve the faster change is taking place.

ease:"bounce" will bounce on the way out
ease:"bounce.in" will bounce on the way in
ease:"bounce.inOut" will bounce on the way in and out

Some eases can be configured

ease:"back.config(6)" will have a stronger overshoot

Practice

1: Visit the GreenSock Ease Visualizer: https://greensock.com/docs/v3/Eases

2: Spend some time playing with it and choosing different eases.

3: Open the pen at: https://codepen.io/snorkltv/pen/oNNjZGY

4: Change the code to use eases from the Ease Visualizer

Special Property: Stagger

The stagger property allows you to offset the start time of multiple targets in a single tween.

In GSAP3 you no longer need the staggerTo(), staggerFrom(), and staggerFromTo() methods of GSAP2.

```
// each image will start 0.2 seconds after the previous one starts.
gsap.to("#freds img", {y:-100, stagger:0.2});
```

A stagger object gives you greater control over where the staggers start from and how the timing is dispersed.

```
gsap.to("#freds img", {y:-50, stagger:{
  each:0.2,
  from:"end"
  }
});
```

each:0.2 means there will be 0.2 seconds between the start of each animation.
amount:0.2 means all animations will start within 0.2 seconds.

Practice

1: Open the pen at: https://codepen.io/snorkltv/pen/bGGMMgz

2: Experiment with different stagger options

```
gsap.to("#freds img", {y:-50, stagger:{
  each:0.2,
  from:"edges"
  }
});
```

Try changing **each** to **amount**.
Try changing **“edges”** to **“center”** or **“end”**

Tween Control

Tween’s have a number of methods for controlling playback.

In order to control a tween you need have way to reference it.

```
var tween = gsap.to("#fred", {x:600});
```

To prevent a tween from playing automatically you can set its paused special property to true.

```
var tween = gsap.to("#fred", {x:600, paused:true});
```

To play that tween you can later call:

```
tween.play();
```

Practice

1: Open the pen at: https://codepen.io/snorkltv/pen/PoojYPV
2: Study the code. Yes, I’m giving you a little break here;)

GSAP Docs

Visit the GSAP Docs https://greensock.com/docs/v3/GSAP/Tween and be sure to bookmark them! Study the methods and properties of the GSAP object and Tween class.

Basic Timeline

A timeline is created with gsap.timeline();
All tweens in a timeline naturally play one after the other.

Practice

1: open the demo at: https://codepen.io/snorkltv/pen/wvvzGEX
2: Type out the first 2 lines and then run the pen after each additional line is added.

```
gsap.timeline()
  .from("#demo", {autoAlpha:0})
  .from("#title", {opacity:0, scale:0, ease:"back"})
  .from("#freds img", {y:160, stagger:0.1, duration:0.8,
ease:"back"})
  .from("#time", {xPercent:100, duration:0.2})
```

3: SAVE the pen so you can use it in the next practice session (you need to have a FREE CodePen account in order to save).

Position Parameter

The position parameter allows you to offset the start time of tweens in a timeline.

```
var tl = gsap.timeline();
  tl.to(object, {y:300}, "+=1")  // start 1 second after previous tween ends
  tl.to(object, {x:300}, "-=1")  // start 1 second before previous tween ends
  tl.to(object, {rotation:90}, "<")  // start when previous tween begins
  tl.to(object, {opacity:0.5}, "<1") // start 1 second after previous tween
begins
  tl.to(object2, {x:200}, 1) // start exactly at a time of 1
```

Read gsap.timeline() docs: https://greensock.com/docs/v3/GSAP/gsap.timeline()

Basic Timeline Position Parameter

Go back to the demo you saved from the previous practice session.
Add some position parameters and experiment with how they work.

Challenge

Using the file from the previous lesson:

- make the freds start 1 second after the #title comes in
- make the #time element start exactly when the freds start animating

Timeline Control and Labels

Timelines have the exact same control methods as tween. Since you already know how to play() a tween you already know how to play() a timeline.

You must first create a reference to your timeline like

```
var animation = gsap.timeline()

// later on you can do
animation.play();
animation.pause();
animation.restart();
animation.reverse();
//etc
```

Labels allow you mark a specific point in time in your timeline.

Add a label to a timeline using the add() method

```
.from("#freds img", {y:160, stagger:0.5, duration:0.8, ease:"back"}, "+=0.5")
.add("test")
.from("#time", {xPercent:100, duration:1, ease:"bounce"});
```

Practice

1: review the code at: https://codepen.io/snorkltv/pen/oNNwNBr

Final Project Basic Animation

In this lesson the objective is to set up a simple sequence of animations. In future lessons we'll tweak the timing and refine things along the way.

Practice

1: Open the demo at: https://codepen.io/snorkltv/pen/ZEEoPPd

2: Add the following code (I suggest typing it line-by-line and running it often so you can see which each line does)

```
var tl = gsap.timeline()
    tl.from("#demo", {opacity:0})
      .from("h1", {x:80, opacity:0})
      .from("h2", {x:-80, opacity:0})
      .from("p", {y:30, opacity:0})
      .from("button", {y:50, opacity:0})
      .from("#items > g", {
                            transformOrigin:"50% 50%",
                            scale:0,
                            opacity:0,
                            stagger:0.1,
                           });
```

3: Save the demo so that you can continue working on it in future lessons.

Final Project Timeline Defaults

Timeline defaults allow us to apply the same property values to every tween in a timeline.

Practice

1: Open the demo you were working on in the last practice session.
2: Remove all instances of opacity:0 from the code as shown below in red (and commas too).

```
var tl = gsap.timeline()
   tl.from("#demo", {opacity:0})
     .from("h1", {x:80, opacity:0})
     .from("h2", {x:-80, opacity:0})
     .from("p", {y:30, opacity:0})
     .from("button", {y:50, opacity:0})
     .from("#items > g", {
                        transformOrigin:"50% 50%",
                        scale:0,
                        opacity:0,
                        stagger:0.1,
                       })
```

3: add a **defaults** object to the timeline function

var tl = gsap.timeline(**{defaults:{ease:"back", opacity:0}}**)

4: add a linear ease to the #demo tween

tl.from("#demo", {**ease:"linear"**})

5: save your project for the next practice session

Final Project GSDevTools

GSDevTools gives you an awesome controller bar that makes it super easy to scrub through your animations, change the speed, loop them, and more.

GSDevTools is bonus tool for Club GreenSock members, but you can use it for free on CodePen.

Read about GSDevTools: https://greensock.com/gsdevtools
Get CodePen-friendly urls for all Club GreenSock plugins:
https://codepen.io/GreenSock/full/23d3979528b262cb07da37f6a7c7dd76

Practice

1: return to your demo from the previous practice session
2: add code following code below your timeline code:

```
GSDevTools.create();
```

3: Run your pen. You should see GSDevTools appear!

4: Save your pen for the next lesson.

Yes, I was a bit sneaky. GSDevTools is already loaded ;)

Final Project Tweak Timing

This lesson was all about tweaking the timing of our tweens and their position. With animation, “timing is everything” and the timing can be very subjective. I’d love for you to experiment on your own now.

Practice

1: Return to your demo from the previous practice session
2: Change the duration of the tweens and add position parameters to suit your liking. Be creative!

If you want to use my file as a reference have a peak here:
https://codepen.io/snorkltv/pen/a50120d3bf20691bc941028417979f77?editors=0010

Final Project Remove Flash of Unstyled Content (FOUC)

The Flash of Unstyled Content occurs when elements appear before they are styled properly. Most commonly this happens when a page renders before the proper font has loaded. You'll see text for a brief second with the wrong font and then it will change.

In order to give users the quickest loading experience its recommended to load your custom scripts after the closing body tag. For our purposes we are using GSAP to animate things in from an opacity of 0, however there is always going to be a brief duration of time after the page is rendered and your JavaScript runs; this is the flash of unstyled content.

To avoid the flash there are a few steps to take:

1. Hide the that contains all your elements by giving it a css style of visibility:hidden
2. Create gsap animation code that fades in from autoAlpha:0
3. Wrap your animation code in an init function().
4. Call the init function after the page loads using a load event listener

Practice

Study the code structure of the finished file here:
https://codepen.io/snorkltv/pen/84cf29bc5d301d5e70e57997dd40dd5b

Credits:
vector artwork for final project from:
https://www.freepik.com/free-vector/gradient-creative-landing-page_5289087.htm#page=1&query=landing%20page&position=17