Notes / Cheat Sheet

####OPEN YOUR BROWSER CONSOLE

***NB* - Always Import The libraries JS files before your own JS Link (GSAP ScrollMagic) It goes in logical order BEWARE**

Tween

Simple Tween

```
TweenMax.to(elem, time, { vars });
```

- `.to()` / `.from()`
- `elem` is the element you want to animate *(Target this with its ID or ClassName)*
- `time` is the duration of the Animation
- `vars` is an JS Object of vars you want to Animate *(CSS)*

Stagger Tween

```
TweenMax.staggerTo(elem, time, { vars }, timeBetween);
```

- `.staggerTo()` / `.staggerFrom()`
- `elem` is the element you want to animate *(Target this with its ID or ClassName)*
- `time` is the duration of the Animation
- `vars` is and JS Object of vars you want to Animate (CSS Properties *(See Further Down)*)

`timeBetween` is a number value of time between the stagger animation

Additional

- `.add()` Used to Add A Label to a Timeline
- `.set()` Sets the initial css values of element

Timeline *(Chaining Tweens)*

Creating a New GSAP timeline

```
var tl = new TimelineLite();
```

A GSAP Timeline is very similar to Tween but we *chain* then together

Example of Simple Timeline

```
tl.to(element1, 1, { x: 50, y: 0 }).to(element2, 1, { x: 50, y: 0 });
```

####Verbose Explanation

When the Timeline named `tl` is called it will move ***element1*** for one second then ***element2*** will be moved for one second

Adjust Time

```
tl.to(element1, 1, { x: 50, y: 0 }).to(element2, 1, { x: 50, y: 0 },
TIME);
```

- `TIME` is a var that uses Relative Numbers
- `TIME = 0.5` will animate element2 one 0.5 seconds into timeline
- `TIME = "-=0.5"` will Overlap element2's animation 0.5 seconds before the previous animation ends
- `TIME = "+=0.5"` will Delay element2's animation 0.5 seconds after the previous animation ends

Label

Label work simulate that adjusting the time abut you are naming it

```
tl.add(LABEL)
  .to(element1, 1, { x: 50, y: 0 })
  .to(element2, 1, { x: 50, y: 0 }, LABEL);
```

CSS Cheat Sheet (origin here)

Standard CSS properties

...are all supported, with `hyphenated-names` becoming `camelCaseNames`. Non-animatable properties are also supported but they will be set at the beginning of the tween.

Special mentions:

- `opacity`/`autoAlpha`: can be used interchangeably but when autoAlpha hits 0 it also sets `visibility: hidden`
- `className`: animates class changes by determining all the rule differences automatically. Overwrites the class by default but can also add/remove if using the `+=` or `-=` prefixes.

- `clearProps`: a comma-delimited list of properties that you want to clear from element's inline styles when tween is over. Allows element to fall back to the stylesheet rules.
- `autoRound: true`: rounds pixel values and zIndex to the closest integer during the tween, for browser performance. Can be disabled with `autoRound: false`. You can still use the RoundPropsPlugin for specific properties.
- `bezier`: animate a property along a bezier path. See BezierPlugin for more info

2D Transform properties

- `rotation`: equivalent of `rotationZ`. uses degrees but also supports radians if specified, e.g. `rotation: '3rad'`
- `directionalRotation`: a suffix to any type of `rotation` value, to enforce the direction (_cw, _ccw, or _short). Can be combined with the "+=" or "-=" prefixes for relative values
- `scale`: takes a decimal number value or percentage value as string (e.g. `0.5` or `'50%'`)—also relative values (e.g. `'+=0.2'` or `'-=10%'`)
- `scaleX`: same format as `scale`
- `scaleY`: same format as `scale`
- `skewX`
- `skewY`: defaults to greensock's 'compensated' skew which is more like what graphics apps produce; for css-native skew (more distorted) set `CSSPlugin.defaultSkewType = 'simple'` or use extra prop `skewType:'simple'`
- `x`: pixel-based `translatex()`
- `y`: pixel-based `translatey()`
- `xPercent`: percent-based `translatex()`
- `yPercent`: percent-based `translatey()` nb. px (`x`) and % (`xPercent`) can be combined in one tween/set

3D Transform properties

- `rotationX`
- `rotationY`
- `rotationZ`: identical to regular rotation
- `z`: pixel-based `translatez()`
- `zPercent`: percent-based `translatez()`
- `perspective`
- `transformPerspective` set `perspective()` property of the parent element or the special transformPerspective`prop of the element or global`CSSPlugin.defaultTransformPerspective`
- `transformOrigin`: as with CSS, can be percentage ("50% 50%") or keyword("top", "left", "right", or "bottom")

Control Your Timeline with Playback Functions

- `tl.play(1.5)` Play from 1.5s
- `tl.play(-1);` Play 1s from end
- `tl.pause();` Pause timeline
- `tl.seek(1.5);` Go to 1.5s or 'label'
- `tl.resume();` Continue playback
- `tl.reverse();` Reverse playback anytime

- `tl.timeScale(2);` Speed up timeline
- `tl.tweenTo('LABEL');` Skips To That Label in the Timeline
- `tl.progress(0.5);` Skip to halfway

JS events to Control Animation *(Self Study)*

Most JS events can Trigger An Animation

Full List Can be Found Here

Example

```
button.addEventListener('mouseenter', function() {
  tl.play();
});
```

Animation Callback Functions

`onStart` `onComplete` `onUpdate` `onRepeat` `onRepeatParams` `onReverseComplete`

Bezier Plugin

Passed as an array of objects the values define the path the element will follow.

Read More In Docs HERE

Example

```
var white1svgPath = {
  curviness: 1.5,
  autoRotate: true,
  type: 'soft',
  values: [
    { left: '10%', bottom: '15%', rotation: 0 },
    { left: '15%', bottom: '25%', rotation: 90 },
    { bottom: '55%', left: '15%', rotation: 180 },
    { bottom: '55%', left: '40%', rotation: 180 },
    { bottom: '40%', left: '40%', rotation: 360 }
  ]
};
```

and then that object is passed to a tween as a css property

```
{
  bezier: white1svgPath,
  ...other css props
}
```

Scroll Control of Animation

ADD-ON for GSAP called Scroll Magic CDN HERE

Boilerplate

```
var controller = new ScrollMagic.Controller();
```

Setting up Scrollmagic Scene

```
var sceneOne = new ScrollMagic.Scene({
  triggerElement: '#trigger',
  duration: '100%',
  triggerHook: 0
})
  .setTween(tl)
  .addIndicators()
  .addTo(controller);
```

- `triggerElement` where the trigger is in the dom *(Set In HTML)*
- `duration` is how long the scroll animation will be - if removed animation will trigger on scroll and behave normally
- `triggerHook` Where the Animation Will be Triggered *(0 -1 Value Range)*

Chain Elements

- `.setTween()` is the Tween or Timeline to trigger on scroll.
- `.addIndicators()` Only For Dev to see where the triggers are *(optional)*
- `.setPin('WHERE_TO_PIN')` Pins Scene and releases it after. *(optional)*
- `.addTo(controller)` Boilerplate work *(Don't Worry)*