GSAP 3 Cheat Sheet

Most code is linked to the appropriate page in the Docs
Links: Get started | Install | Forums | Tips | Learning | CodePen | Club

Basics

```
// "to" tween (animate to provided values)
gsap.to(".selector", { // selector text, Array, or object
  x: 100, // any properties (not limited to CSS)
  backgroundColor: "red", // camelCase
  duration: 1, // seconds
  delay: 0.5,
  ease: "power2.inOut",
  stagger: 0.1, // stagger start times
  paused: true, // default is false
  overwrite: "auto", // default is false
  repeat: 2, // number of repeats (-1 for infinite)
  repeatDelay: 1, // seconds between repeats
  repeatRefresh: true, // invalidates on each repeat
  yoyo: true, // if true > A-B-B-A, if false > A-B-A-B
  yoyoEase: true, // or ease like "power2"
  immediateRender: false,
  onComplete: myFunc,
  // other callbacks:
  // onStart, onUpdate, onRepeat, onReverseComplete
  // Each callback has a params property as well
  // i.e. onUpdateParams (Array)
});

// "from" tween (animate from provided values)
gsap.from(".selector", {fromVars});

// "fromTo" tween (define both start and end values)
gsap.fromTo(".selector", {fromVars}, {toVars});
// special properties (duration, ease, etc.) go in toVars

// set values immediately (no animation)
gsap.set(".selector", {toVars});
```

Timelines

```
// Create a timeline
let tl = gsap.timeline({
  delay: 0.5,
  paused: true, // default is false
  repeat: 2, // number of repeats (-1 for infinite)
  repeatDelay: 1, // seconds between repeats
  repeatRefresh: true, // invalidates on each repeat
  yoyo: true, // if true > A-B-B-A, if false > A-B-A-B
  defaults: { // children inherit these defaults
    duration: 1,
    ease: "none"
  },
  smoothChildTiming: true,
  autoRemoveChildren: true,
  onComplete: myFunc,
  // other callbacks:
  // onStart, onUpdate, onRepeat, onReverseComplete
  // Each callback has a params property as well
  // i.e. onUpdateParams (Array)
});

// Sequence multiple tweens
tl.to(".selector", {duration: 1, x: 50, y: 0})
  .to("#id", {autoAlpha: 0})
  .to(elem, {duration: 1, backgroundColor: "red"})
  .to([elem, elem2], {duration: 3, x: 100});

// position parameter (controls placement)
tl.to(target, {toVars}, positionParameter);

0.7 // exactly 0.7 seconds into the timeline (absolute)
"-=0.7" // overlap with previous by 0.7 sec
"myLabel" // insert at "myLabel" position
"myLabel+=0.2" // 0.2 seconds after "myLabel"
"<" // align with start of most recently-added child
"<0.2" // 0.2 seconds after ^^
```

Control methods

```
// retain animation reference to control later
let anim = gsap.to(...); // or gsap.timeline(...);
// most methods can be used as getters or setters
anim.play() // plays forward
  .pause()
  .resume() // respects direction
  .reverse()
  .restart()
  .timeScale(2) // 2 = double speed, 0.5 = half speed
  .seek(1.5) // jump to a time (in seconds) or label
  .progress(0.5) // jump to halfway
  .totalProgress(0.8) // includes repeats
  // when used as setter, returns animation (chaining)

  // other useful methods (tween and timeline)
  .kill() // immediately destroy
  .isActive() // true if currently animating
  .then() // Promise
  .invalidate() // clear recorded start/end values
  .eventCallback() // get/set an event callback

  // timeline-specific methods
  // add label, tween, timeline, or callback
  .add(thing, position)
  // calls function at given point
  .call(func, params, position)
  // get an Array of the timeline's children
  .getChildren()
  // empties the timeline
  .clear()
  // animate playhead to a position linearly
  .tweenTo(timeOrLabel, {vars})
  // ^^ with both start and end positions
  .tweenFromTo(from, to, {vars})
```

Eases

```
// see greensock.com/ease-visualizer
ease: "none" // no ease (same as "linear")

// basic core eases
"power1", "power2", "power3", "power4",
"circ", "expo", "sine"
// each has .in, .out, and .inOut extensions
// i.e. "power1.inOut"

// expressive core eases
"elastic", "back", "bounce", "steps(n)"

// in EasePack plugin (not core)
"rough", "slow", "expoScale(1, 2)"

// members-only expressive plugins
CustomEase, CustomWiggle, CustomBounce
```

ScrollTrigger

```
scrollTrigger: {
  trigger: ".selector", // selector or element
  start: "top center",  // [trigger] [scroller] positions
  end: "20px 80%", // [trigger] [scroller] positions
  scrub: true, // or time (in seconds) to delay
  pin: true, // or selector or element to pin
  markers: true, // only during development!
  toggleActions: "play pause resume reset",
  // other actions: complete reverse none
  toggleClass: "active",
  id: "my-id",
  anticipatePin: 1, // can help avoid flash
  snap: 1 / 10, // progress increment
  pinReparent: true, // moves to documentElement during pin
  once: true,
  endTrigger: ".selector", // selector or element
  horizontal: true, // switches mode
  onEnter: callback
  invalidateOnRefresh: true, // clears start values on refresh
  // other callbacks:
  // onLeave, onEnterBack, onLeaveBack, onUpdate,
  // onToggle, onRefresh, onRefreshInit, onScrubComplete
}
```

Other Plugins

```
// Register GSAP plugins (once) before using them
gsap.registerPlugin(Draggable, TextPlugin);

// Available plugins
Draggable, DrawSVGPlugin*, EaselPlugin,
GSDevTools*, IntertiaPlugin*, MorphSVGPlugin*,
MotionPathPlugin, MotionPathHelper*, Physics2DPlugin*,
PhysicsPropsPlugin*, PixiPlugin, ScrambleTextPlugin*,
ScrollToPlugin, ScrollTrigger, SplitText*, TextPlugin
// * available to Club GreenSock members. greensock.com/club
```

Installation

```
// Import and register GSAP (ES Modules)
import { gsap } from "gsap";
import { DrawSVGPlugin } from "gsap/DrawSVGPlugin";
gsap.registerPlugin(DrawSVGPlugin);

// Import and register GSAP (UMD/CommonJS)
import { gsap } from "gsap/dist/gsap";
import { DrawSVGPlugin } from "gsap/dist/DrawSVGPlugin";
gsap.registerPlugin(DrawSVGPlugin);
```

Utility methods

```
// accessible through gsap.utils.foo()
checkPrefix() // get relevant browser prefix for property
clamp() // clamp value to range
distribute() // distribute value among and array
getUnit() // get unit of string
interpolate() // interpolate between values
mapRange() // map one range to another
normalize() // map a range to the 0-1 range
pipe() // sequence function calls
random() // generates a random value
shuffle() // shuffles an array in-place
snap() // snap a value to either increment or array
splitColor() // splits color into RGB array
toArray() // convert array-like thing to array
unitize() // adds specified unit to function results
wrap() // place number in range, wrapping to start
wrapYoyo() // place number in range, wrapping in reverse
```

Nesting Timelines

```
function scene1() {
  let tl = gsap.timeline();
  tl.to(...).to(...); // build scene 1
  return tl;
}

function scene2() {
  let tl = gsap.timeline();
  tl.to(...).to(...); // build scene 2
  return tl;
}

let master = gsap.timeline()
  .add(scene1())
  .add(scene2(), "-=0.5") // overlap slightly
```

Miscellaneous

```
// Get the current value of a property
gsap.getProperty("#id", "x");       // 20
gsap.getProperty("#id", "x", "px"); // "20px"

// Set GSAP's global tween defaults
gsap.defaults({ease: "power2.in", duration: 1});

// Configure GSAP's non-tween-related settings
gsap.config({
  autoSleep: 60,
  force3D: false,
  nullTargetWarn: false,
  units: {left: "%", top: "%", rotation: "rad"}
});

// Register an effect for reuse
gsap.registerEffect({
  name: "fade",
  effect: (targets, config) => {
    return gsap.to(targets, {
      duration: config.duration,
      opacity: 0
    });
  },
  defaults: {duration: 2},
  extendTimeline: true
});

// Now we can use it like this
gsap.effects.fade(".box");
// Or directly on timelines
tl.fade(".box", {duration: 3})

// Add listener
gsap.ticker.add(myFunction);
function myFunction() {
  // Executes on every tick after
  // the core engine updates
}
// To remove the listener later...
gsap.ticker.remove(myFunction);

// faster way to repeatedly set property than .set()
const setX = gsap.quickSetter("#id", "x", "px");
document.addEventListener("pointermove", e => setX(e.clientX) );
```