GSAP | Docs & Learning

gsap.com/cheatsheet


```
// "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:()=>{
console.log("finished")
},
// other callbacks:
// onStart, onUpdate, onRepeat, onReverseComplete
});

// "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 });

// 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:()=>{
console.log("finished")
},
// other callbacks:
// onStart, onUpdate, onRepeat, onReverseComplete
});

// 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 ^^
('-=50%');// overlap half of inserting animation's duration
('<25%');// 25% into the previous animation (from its start)
```

```
// 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})

// 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)'
//expressive plugin eases
CustomEase,CustomWiggle,CustomBounce;
```

```
scrollTrigger:{
trigger:".selector",// selector or element
start:"top center",// [trigger] [scroller] positions
end:"20px 80%",// [trigger] [scroller] positions
// or relative amount: "+=500"
scrub:true,// or time (in seconds) to catch up
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",
fastScrollEnd:true,// or velocity number
containerAnimation: tween,// linear animation
id:"my-id",
anticipatePin:1,// may help avoid jump
snap:{
snapTo:1/10,// progress increment
// or "labels" or function or Array
duration:0.5,
directional:true,
ease:"power3",
onComplete: callback,
// other callbacks: onStart, onInterrupt
},
pinReparent:true,// moves to documentElement during pin
pinSpacing:false,
pinType:"transform",// or "fixed"
pinnedContainer:".selector",
preventOverlaps:true,// or arbitrary string
once:true,
endTrigger:".selector",// selector or element
horizontal:true,// switches mode
invalidateOnRefresh:true,// clears start values on refresh
refreshPriority:1,// influence refresh order
onEnter: callback
// other callbacks:
// onLeave, onEnterBack, onLeaveBack, onUpdate,
// onToggle, onRefresh, onRefreshInit, onScrubComplete
}

// Register GSAP plugins (once) before using them
gsap.registerPlugin(Draggable,TextPlugin);
// Available plugins
Draggable,DrawSVGPlugin,EaselPlugin,Flip,
GSDevTools,InertiaPlugin,MorphSVGPlugin,
MotionPathPlugin,MotionPathHelper,Observer,
Physics2DPlugin,PhysicsPropsPlugin,PixiPlugin,ScrambleTextPlugin,
ScrollToPlugin,ScrollTrigger,ScrollSmoother,SplitText,TextPlugin
```

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


functionscene1(){
let tl = gsap.timeline();
  tl.to(...).to(...);// build scene 1
return tl;



functionscene2(){
let tl = gsap.timeline();
  tl.to(...).to(...);// build scene 2
return tl;


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

// 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,
trialWarn: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 with gsap.ticker
gsap.ticker.add(myFunction);
functionmyFunction(time, deltaTime, frame){
// Executes on every tick after
// the core engine updates
}
// To remove the listener later...
gsap.ticker.remove(myFunction);
```