□□ GSAP Cheat Sheet (WordPress/Bricks-Friendly)

A dense, practical reference for daily use: syntax, patterns, gotchas, and copy-paste snippets. □

---

1) Quick Setup (CDN) + Plugin Registration □□

Minimal GSAP Core (recommended starting point)

```
<script src="https://cdn.jsdelivr.net/npm/gsap@3/dist/gsap.min.js"></script>
```

ScrollTrigger (very common)

```
<script src="https://cdn.jsdelivr.net/npm/gsap@3/dist/ScrollTrigger.min.js"></script>
<script>
  gsap.registerPlugin(ScrollTrigger);
</script>
```

Pro/Club plugins (e.g., SplitText, Flip, ScrollSmoother)

- These typically **cannot** be loaded from public CDNs. Use the official files and load them via WordPress enqueue or a reliable asset pipeline.

---

2) Core Concepts (What GSAP *is*)

Targets

GSAP animates **targets** (elements, arrays of elements, selector strings, objects).

```
gsap.to(".card", { opacity: 1 }); // selector string
gsap.to(document.querySelector(".card"), { x: 50 }); // element
gsap.to(document.querySelectorAll(".card"), { stagger: 0.1, y: 20 }); // NodeList works
```

Tween vs Timeline

1. **Tween** = animates *something* over time.
2. **Timeline** = sequences multiple tweens into a *choreography*.

---

3) The Big 3: `to`, `from`, `fromTo`

`gsap.to(target, vars)`

Animates **from current state → to vars**.

```
gsap.to(".box", { x: 200, duration: 0.8, ease: "power2.out" });
```

`gsap.from(target, vars)`

Animates **from vars → to current state** (great for entrances).

```
gsap.from(".box", { y: 30, opacity: 0, duration: 0.6, ease: "power2.out" });
```

`gsap.fromTo(target, fromVars, toVars)`

Use when you want **full control** of both ends.

```
gsap.fromTo(".box",
  { y: 30, opacity: 0 },
  { y: 0, opacity: 1, duration: 0.6, ease: "power2.out" }
);
```

---

4) Most-Used Properties (and what they map to) □□

Transforms (fast, preferred)

- `x`, `y`, `z` → `translate3d(...)`
- `scale`, `scaleX`, `scaleY`
- `rotation` (degrees), `rotationX`, `rotationY`
- `skewX`, `skewY`
- `transformOrigin`

```
gsap.to(".img", { scale: 1.05, rotation: 1.5, duration: 0.7 });
```

Visibility

- `opacity` (0–1)
- `autoAlpha` = `opacity` + toggles `visibility:hidden` when 0 □ (great for accessibility/interaction)

```
gsap.to(".panel", { autoAlpha: 1, duration: 0.4 });
```

Layout properties (use cautiously; can cause reflow)

- `top/left/width/height/margin/padding` etc.
  Prefer transforms for motion; animate layout only if necessary.

CSS variables (powerful)

```
gsap.to(".theme", { "--accent": "#ff3b30", duration: 0.5 });
```

Filters (expensive on many devices)

```
gsap.to(".hero", { filter: "blur(0px)", duration: 0.8 });
```

---

5) Timing & Control: `duration`, `delay`, `repeat`, `yoyo`

```
gsap.to(".dot", {
  y: -12,
  duration: 0.35,
  repeat: 5,      // number of repeats; -1 = infinite
  yoyo: true,     // ping-pong back and forth
  repeatDelay: 0.1,
  delay: 0.2
});
```

`repeatRefresh` (recomputes start values each repeat)

Useful when values are function-based/random:

```
gsap.to(".spark", {
  x: () => gsap.utils.random(-80, 80),
  y: () => gsap.utils.random(-80, 80),
  repeat: -1,
  repeatRefresh: true,
  duration: 0.6
});
```

6) Easing Cheat Sheet (what to reach for) □□

GSAP ease strings:

- **UI/snappy:** `"power2.out"`, `"power3.out"`
- **soft/floaty:** `"sine.out"`, `"power1.out"`
- **overshoot:** `"back.out(1.7)"`
- **bouncy:** `"elastic.out(1, 0.3)"` (use sparingly)
- **linear/constant:** `"none"`

```
gsap.to(".cta", { y: 0, ease: "back.out(1.4)", duration: 0.7 });
```

7) Staggers (instant “pro” look) □

Basic stagger

```
gsap.from(".item", {
  y: 18,
  autoAlpha: 0,
  duration: 0.55,
  stagger: 0.06,
  ease: "power2.out"
});
```

Stagger object

```
gsap.from(".item", {
  autoAlpha: 0,
  y: 16,
  stagger: {
    each: 0.05,
```

```
    from: "start",   // "center", "end", "edges", or index number
    grid: "auto"
  }
});
```

Random order

```
gsap.from(".item", { autoAlpha: 0, stagger: { each: 0.04, from: "random" } });
```

---

8) Callbacks & Lifecycle Hooks □□

```
gsap.to(".progress", {
  scaleX: 1,
  duration: 1.2,
  onStart: () => console.log("start"),
  onUpdate: () => console.log("updating", gsap.getProperty(".progress", "scaleX")),
  onComplete: () => console.log("done")
});
```

Common ones:

- `onStart`, `onUpdate`, `onComplete`
- `onReverseComplete`
- `onRepeat`

---

9) Getting & Setting Values (indispensable) □□

`gsap.set()`

Set instantly (no tween).

```
gsap.set(".modal", { autoAlpha: 0, y: 20 });
```

`gsap.getProperty(target, property)`

```
const x = gsap.getProperty(".box", "x");
```

`gsap.utils` helpers (gold)

```
const clamp = gsap.utils.clamp(0, 1);
const snap = gsap.utils.snap(10);
const mapRange = gsap.utils.mapRange(0, 100, 0, 1);

const q = gsap.utils.selector(document.querySelector(".section"));
q(".item"); // scoped selection within a section 
```

---

10) Timelines

Create timeline + add tweens

```
const tl = gsap.timeline({ defaults: { duration: 0.6, ease: "power2.out" } });

tl.from(".h", { y: 20, autoAlpha: 0 })
  .from(".sub", { y: 16, autoAlpha: 0 }, "-=0.35")
  .from(".btn", { y: 12, autoAlpha: 0 }, "-=0.25");
```

Position parameter (the “when”)

1. **Absolute time** (seconds)

   ```
   tl.to(".a", { x: 100 }, 0.5);
   ```

2. **Relative**

```
tl.to(".a", { x: 100 }, "+=0.2"); // start 0.2s after previous ends
tl.to(".b", { x: 100 }, "-=0.3"); // overlap previous by 0.3s
```

3. **Labels**

```
tl.addLabel("intro")
  .from(".a", { autoAlpha: 0 }, "intro")
  .from(".b", { autoAlpha: 0 }, "intro+=0.15");
```

Timeline controls

```
tl.play();
tl.pause();
tl.reverse();
tl.restart();
tl.timeScale(1.2);     // speed up
tl.progress(0.5);      // jump to 50%
tl.seek("intro+=0.2"); // jump to label
```

---

11) Common UI Patterns (Copy/Paste) □□

Hover “lift” (with overwrite to prevent jitter)

```
const btn = document.querySelector(".btn");

btn.addEventListener("mouseenter", () => {
  gsap.to(btn, { y: -3, duration: 0.2, ease: "power2.out", overwrite: "auto" });
});

btn.addEventListener("mouseleave", () => {
```

```
  gsap.to(btn, { y: 0, duration: 0.2, ease: "power2.out", overwrite: "auto" });
});
```

Toggle open/close with a paused timeline

```
const panel = document.querySelector(".panel");
const tl = gsap.timeline({ paused: true })
  .set(panel, { autoAlpha: 1 })
  .fromTo(panel, { y: 12 }, { y: 0, duration: 0.25, ease: "power2.out" });

let open = false;
document.querySelector(".toggle").addEventListener("click", () => {
  open ? tl.reverse() : tl.play();
  open = !open;
});
```

---

12) ScrollTrigger Essentials □□

Basic reveal on scroll

```
gsap.from(".reveal", {
  y: 20,
  autoAlpha: 0,
  duration: 0.7,
  ease: "power2.out",
  scrollTrigger: {
    trigger: ".reveal",
    start: "top 85%",
    toggleActions: "play none none reverse"
    // markers: true
  }
});
```

Scrub: tie animation to scroll

```
gsap.to(".bar", {
  scaleX: 1,
  ease: "none",
  scrollTrigger: {
    trigger: ".section",
    start: "top top",
    end: "bottom top",
    scrub: true
  }
});
```

Pinning a section

```
ScrollTrigger.create({
  trigger: ".pinned",
  start: "top top",
  end: "+=800",
  pin: true,
  pinSpacing: true
});
```

One timeline controlled by scroll

```
const tl = gsap.timeline({
  scrollTrigger: {
    trigger: ".story",
    start: "top top",
    end: "+=1200",
    scrub: 0.6,
    pin: true
  }
});

tl.to(".img", { scale: 1.1 })
  .to(".txt", { autoAlpha: 1, y: 0 }, "<");
```

Responsive triggers with `matchMedia`

```
const mm = gsap.matchMedia();

mm.add("(min-width: 992px)", () => {
  // desktop triggers
});

mm.add("(max-width: 991px)", () => {
  // mobile triggers
});
```

---

13) Preventing Conflicts: `overwrite` + killing animations □□

Overwrite (crucial for hover/drag/rapid interaction)

```
gsap.to(".box", { x: 100, overwrite: "auto" });
```

Kill tweens / timelines

```
gsap.killTweensOf(".box");
tl.kill();
```

ScrollTrigger cleanup

```
ScrollTrigger.getAll().forEach(st => st.kill());
```

---

14) Defaults & Reuse (make it scalable) □□

Global defaults

```
gsap.defaults({ duration: 0.7, ease: "power2.out" });
```

Reusable function that returns a timeline

```
function heroIntro(root) {
  const q = gsap.utils.selector(root);
  return gsap.timeline()
    .from(q(".h"), { y: 24, autoAlpha: 0 })
    .from(q(".sub"), { y: 16, autoAlpha: 0 }, "-=0.35");
}

document.querySelectorAll(".hero").forEach(hero => heroIntro(hero));
```

---

15) Performance Rules (fast checklist) ⚙

1. Prefer animating:
   1. `transform` (`x`, `y`, `scale`, `rotation`)
   2. `opacity` / `autoAlpha`
2. Avoid heavy use of:
   1. layout props (`top/left/width/height`)
   2. `filter` (blur) on large areas
3. Use `will-change` sparingly (don’t blanket everything).
4. For many elements, use:
   1. stagger
   2. ScrollTrigger batching
   3. fewer simultaneous tweens

16) BricksBuilder + WordPress Specific Notes □□□

Targeting: prefer **classes/data attributes** over auto-generated IDs

- Add your own class like `.js-reveal` or attribute `data-anim="reveal"`.

Avoid duplicate initialization (common in WP builders)

A safe pattern:

```
document.querySelectorAll("[data-anim='reveal']").forEach(el => {
  if (el.dataset.gsapInit) return;
  el.dataset.gsapInit = "1";

  gsap.from(el, {
    y: 18,
    autoAlpha: 0,
    duration: 0.65,
    scrollTrigger: { trigger: el, start: "top 85%" }
  });
});
```

When content height changes (images/fonts, accordions)

- Call `ScrollTrigger.refresh()` after significant layout changes.

17) Troubleshooting (most common “why isn’t it working?”) □□

1. **Selector matches multiple things**
   - Confirm with `document.querySelectorAll(".x").length`.
2. **Element starts hidden due to CSS**
   - Remove conflicting CSS transitions/opacity rules or use `gsap.set(...)`.
3. **ScrollTrigger feels “off”**
   - Enable `markers: true`, verify `start/end`, then check layout shifts (images loading).
4. **Hover animations stutter**
   - Add `overwrite: "auto"` and keep durations short.
5. **Transforms look weird**
   - Check `transform-origin`, existing transforms, and parent transforms.

---

18) Handy Reference Table □□

1. **Tweens**
   1. `gsap.to(target, vars)`
   2. `gsap.from(target, vars)`
   3. `gsap.fromTo(target, fromVars, toVars)`
   4. `gsap.set(target, vars)`
2. **Timelines**
   1. `gsap.timeline({ paused, defaults, onComplete... })`
   2. `.to() .from() .fromTo()`
   3. `.addLabel(name)` / position params: `"+=0.2"`, `"-=0.3"`, `"<"`, `">"`
3. **Utilities**
   1. `gsap.utils.selector(root)`
   2. `gsap.utils.clamp(min,max)`
   3. `gsap.utils.snap(increment)`
   4. `gsap.utils.random(min,max)`
   5. `gsap.matchMedia()`
4. **ScrollTrigger**
   1. `scrollTrigger: { trigger, start, end, scrub, pin, markers, toggleActions }`
   2. `ScrollTrigger.create({...})`
   3. `ScrollTrigger.refresh()`

Revision #1
Created 2026-04-19 18:16:05 UTC by art10m
Updated 2026-04-19 18:17:15 UTC by art10m