**TimeLine** — это контейнер для анимации, который позволяет размещать анимацию в реальном времени. Этого можно добиться с помощью различных методов GSAP. Ниже приведен пример типичной timeline в GSAP:

**let tl = gsap.timeline();**

**tl.to(".box-1", {duration: 1, opacity: 0 })**

**.to(".box-2", {duration: 1, opacity: 0,5 })**

**.to(".box-3", {duration: 1, opacity: 0,7 });**

**to()**

Это наиболее распространенный метод GSAP, используемый для установки конечных значений данной анимации. При использовании этого метода GSAP автоматически устанавливает начальные значения на основе заданных свойств CSS данного элемента.

**gsap.to(".box" , {borderRadius: 50, x: 100, duration: 1})**

**from()**

Метод from() лучше всего использовать для обратной анимации. Когда вы устанавливаете начальные свойства CSS, GSAP вернется к значениям по умолчанию. Например, если непрозрачность установлена на 0, тогда GSAP начинает анимацию с непрозрачности 0 до тех пор, пока не достигнет первоначально установленной непрозрачности (в нашем примере — 1). Ниже приведен пример анимации с использованием метода from().

**gsap.from(".box" , {y:80, opacity:0, duration:2})**

**fromTo()**

С помощью метода fromTo () разработчики могут определять, как начальные, так и конечные свойства данной анимации. Синтаксис этого метода немного отличается от синтаксиса двух других методов, упомянутых выше, поскольку он принимает два разных объекта, используемых для указания начального и конечного значения:

**gsap.fromTo(".box", {x: 20 , y: 10}, { x: 200 , y : 100, duration: 1});**

**set()**

Метод set() — это метод нулевого направления, используемый для установки свойств данного элемента. Этот метод пригодится, когда вы хотите установить свойства, которые позже вы будете анимировать с помощью GSAP.

**gsap.set(".box", {x: 10 , y: 20});**

Из вышесказанного, координаты x и y элемента с классом «.box» установлены равными 10 и 20 соответственно. Затем вы можете анимировать элемент позже с уже установленными начальными значениями.

**Взаимодействие методов и базовая последовательность**

Чтобы добиться более лаконичной анимации, в которой мы точно определяем, какая анимация идет первой, нам нужно привязать метод to() и перечислить анимации в том порядке, в котором мы хотим их видеть. Для этого мы сначала создадим временную шкалу (timeline):

**var tl = gsap.timeline({repeat: 30, repeatDelay: 1});**

Затем мы можем приступить к добавлению каждой из наших анимаций движения на уже созданную временную шкалу:

**tl.add( gsap.to("#box", {duration: 1, x: 100}) );**

**tl.add( gsap.to("#box2", {duration: 2, x: 100 , scale: 1.1}) );**

**tl.add( gsap.to("#box3", {duration: 3, x: 100 , scale: 2}) );**

**Функции управления анимацией**

Функции управления анимацией дают вам полный контроль над анимацией, позволяя выполнять такие действия, как пауза и реверс. Функции управления включают, среди прочего, play(), pause(), reverse(), kill() и resume().

const instance = gsap.to(".box", {x: 100 , y: 100});

document.querySelector(".pause").onclick = () => instance.pause();

**Приоритет (stagger)**

Приоритет используются для анимации группы элементов. Он позволяет указать время начала анимации каждого элемента в группе.

gsap.from(".box", { duration: 2, scale: 0.5, opacity: 0, delay: 0.5, **stagger: 1**, ease: "elastic", force3D: true});

**Контролируйте размещение с помощью параметра position**

Точно определите, где вы хотите разместить свои анимации на временной шкале, используя необязательный параметр **position** .

Число указывает абсолютное время (в секундах), или строка с префиксом "+=" или "-=" указывает смещение относительно КОНЦА временной шкалы. Например, "+=2" будет через 2 секунды после окончания, создавая 2-секундный промежуток. "-=2" создаст 2-секундное перекрытие.

**tl.to(..., 1.5)**

**.to(..., "-=0.75") //overlaps by 0.75 seconds**

**.to(..., "+=1") //adds a 1-second gap before**

**Свойства CSS**

GSAP может анимировать любое анимируемое свойство CSS, а также многие из тех, которые официально не являются анимируемыми с помощью CSS.

**gsap.to(element, {**

**backgroundColor: "red", // background-color**

**fontSize: 12, // font-size**

**boxShadow: "0px 0px 20px 20px red", // animate complex strings**

**borderRadius: "50% 50%",**

**height: "auto", // animate between auto and a px value 🪄**

**});**

|  |  |
| --- | --- |
| **GSAP** | **Description or equivalent CSS** |
| x: 100 | transform: translateX(100px) |
| y: 100 | transform: translateY(100px) |
| xPercent: 50 | transform: translateX(50%) |
| yPercent: 50 | transform: translateY(50%) |
| scale: 2 | transform: scale(2) |
| scaleX: 2 | transform: scaleX(2) |
| scaleY: 2 | transform: scaleY(2) |
| rotation: 90 | transform: rotate(90deg) |
| rotation: "1.25rad" | Using Radians - no CSS alternative |
| skew: 30 | transform: skew(30deg) |
| skewX: 30 | transform: skewX(30deg) |
| skewY: "1.23rad" | Using Radians - no CSS alternative |
| transformOrigin: "center 40%" | transform-origin: center 40% |
| opacity: 0 | adjust the elements opacity |
| autoAlpha: 0 | shorthand for opacity & visibility |
| duration: 1 | animation-duration: 1s |
| repeat: -1 | animation-iteration-count: infinite |
| repeat: 2 | animation-iteration-count: 2 |
| delay: 2 | animation-delay: 2 |
| yoyo: true | animation-direction: alternate |