1. 기본 애니메이션

JS

```
//애니메이션 효과를 적용할 요소 선택
const box1 = document.querySelector("#section1 .parallax__item__img");
const box2 = document.querySelector("#section2 .parallax__item__img");
const box3 = document.querySelector("#section3 .parallax__item__img");
const box4 = document.querySelector("#section4 .parallax__item__img");
const box5 = document.querySelector("#section5 .parallax__item__img");
const box6 = document.querySelector("#section6 .parallax__item__img");
const box7 = document.querySelector("#section7 .parallax__item__img");
const box8 = document.querySelector("#section8 .parallax__item__img");
const box9 = document.querySelector("#section9 .parallax__item__img");
```

```
// box1 요소
// gsap의.to 메서드를 사용하면 선택한 요소는 움직입니다.
// gsap의 가장 기본 애니메이션입니다.
gsap.to(box1, {
    duration: 2, //2초동안
    x: 500, //오른쪽으로 500px 이동
    borderRadius: 100, //모서리를 반지름 100만큼 둥글게
    rotation: 360, //360도 회전
});
```

trigger

trigger를 설정하면 스크롤을 내릴때 움직이기 시작합니다

```
// box2 요소
gsap.to(box2, {
    duration: 2,
    x: 500,
    rotation: 360,
    borderRadius: 100,
    scrollTrigger: {
        trigger: box2,  //트리거를 box2로 설정 box2가 보이는 영역에 오면 애니메이션이 실행 됩니다.
    }
});
```

toggleActions

애니메이션의 행동을 4가지로 설정할 수 있습니다

- 애니메이션이 시작했을 때 **(onEnter)**
- 애니메이션이 끝났을 때 **(onLeave)**
- 애니메이션이 시작하고 화면에 보이지 않을 때 **(onEnterBack)**
- 애니메이션이 끝나고 화면에 보이지 않을 때 **(onLeaveBack)**

여기에는 `play`, `pause`, `resume`, `reset`, `restart`, `complete`, `reverse`, `none` 요소 값을 설정할 수 있습니다

```
gsap.to(box3, {
    duration: 1,
    x: 500,
    rotation: 360,
```

```
    borderRadius: 100,

    scrollTrigger: {
        trigger: box3,
        toggleActions: "play pause reverse none"
    }
});
```

start, end

> trigger는 애니메이션의 기준점 역할을 하고, start는 시작점을 의미합니다

start와 end는 두가지 값을 설정합니다.

- 첫 번째는 요소의 시작점을 의미하고, 두 번째는 브라우저의 시작점을 의미합니다.
- 요소의 시작점과 브라우저의 시작점이 만나면 애니메이션이 작동되는 원리입니다.
- 여기에는top,bottom,left,right,center를 사용할 수 있으며, px이나% 사용도 가능합니다.
- 여기에서 markers: true로 설정하면 마커의 위치를 확인할 수 있습니다.

```
gsap.to(box4, {
    duration: 1,
    x: 500,
    rotation: 360,
    borderRadius: 100,

    scrollTrigger: {
        trigger: box4,
        start: "top 50%",
        end: "bottom 20%",
        toggleActions: "play pause reverse pause",
        markers: true,
```

```
    }
});
```

scrub

> 이 속성은 스크롤을 내리면 같이 움직이게 설정할 수 있습니다
>
> 스크롤 위치와 애니메이션을 동기화 시킴
>
> 이 속성에는 `true` 및 정수 값을 넣을 수 있습니다

```
gsap.to(box5, {
    duration: 1,
    x: 500,
    rotation: 360,
    borderRadius: 100,

    scrollTrigger: {
        trigger: box5,
        start: "top 50%",
        end: "bottom 20%",
        scrub: 0.5,    //true, 1, 2,....
        markers: false,
    }
});
```

pin

> pin 속성은 고정시키는 역할을 합니다.
>
> 위치한 영역에 고정시키기 위해서는pin: true를 설정합니다

```
gsap.to(box6, {
    duration: 2,
    x: 500,
    rotation: 360,
    borderRadius: 100,

    scrollTrigger: {
        trigger: box6,
        start: "top 50%",
        end: "top 100px",
        pin: true,
        scrub: true,
        markers: true,
    }
});
```

toggleClass

> 시작점에 됐을 때 애니메이션도 줄 수 있지만class도 추가할 수 있습니다

```
gsap.to(box7, {
    duration: 2,
    x: 500,
    rotation: 360,
    borderRadius: 100,

    scrollTrigger: {
        trigger: box7,
        start: "top center",
        end: "bottom top",
        scrub: true,
        markers: true,
        toggleClass: "active", //active클래스를 토글 합니다
```

```
        id: "box7"
    }
});
```

callback

> 하나의 함수를 실행하고 그 다음 함수를 실행하는 함수
>
> `toggleActions` 처럼 `onEnter`, `onLeave`, `onEnterBack`, `onLeaveBack` 메서드를 제공하며, `onUpdate` 이나 `onToggle` 같은 메서드도 제공합니다

```
gsap.to(box8, {
    duration: 2,
    x: 500,
    rotation: 360,
    borderRadius: 100,

    scrollTrigger: {
        trigger: box8,
        start: "top center",
        end: "bottom 30%",
        scrub: true,
        markers: false,
        // 애니메이션 시작할 때 호출
        // onEnter : () => {console.log("onEnter")},

        // 애니메이션이 끝났을 때 호출
        // onLeave : () => {console.log("onLeave")},

        // 애니메이션이 시작하고 화면에 보이지 않을 때 호출
        // onEnterBack : () => {console.log("onEnterBac
k")},

        // 애니메이션이 끝나고 화면에 보이지 않을 때 호출
        // onLeaveBack : () => {console.log("onLeaveBac
```

```
k")},

        // 애니메이션 진행 상태가 업데이트될 때 호출
        // self.progress를 통해 애니메이션의 진행 상태를 소수점 3자
리까지 콘솔에 출력
        // onUpdate : (self) => {console.log("onUpdate", se
lf.progress.toFixed(3))},

        // 애니메이션의 활성 상태가 변경될 때 호출
        // self.isActive를 통해 애니메이션이 활성화 또는 비활성 되
었는지 출력
        onToggle : (self) => {console.log("onToggle", self.
isActive)},
    }
});
```