If We Want Our Input To Take Full Size Container, We Can Change The Attributes Of position, top, width, height:

```
.input {
  width: 100%;
  height: 100%;
  position: absolute;
  top: 0;
  border: none;
}

.input-container {
  position: relative;
}

.placeholder {
  position: relative;
  z-index: 1;
}
```

****

If We Want Our Made Placeholder To Make The Focus For Our Input, We Can Use pointer-event:

```
.placeholder {
  position: relative;
  z-index: 1;
  pointer-events: none;
}
```

****

In This Way We Can Make Our SVG Line Appear:

```
.line-svg {
   position: absolute;
   left: 0;
   bottom: 0;
}
```

****

If We Want Our Animation To Start Only Once When Clicked, We Can Use Focus-Event Instead Of Click-Event.

****

To Animate The Path Of SVG We Can Use **attr** **And** **d** Attribute Of GSAP TimeLine.

First, We Define The Start And End Of SVG Path:

```
const start = 'M0 0.999512C0 0.999512 60.5 0.999512 150 0.999512C239.5 0.999512 300 0.999512 300 0.999512';

const end   = 'M1 0.999512C1 0.999512 61.5 7.5 151 7.5C240.5 7.5 301 0.999512 301 0.999512';
```

Then We Animate The Line:

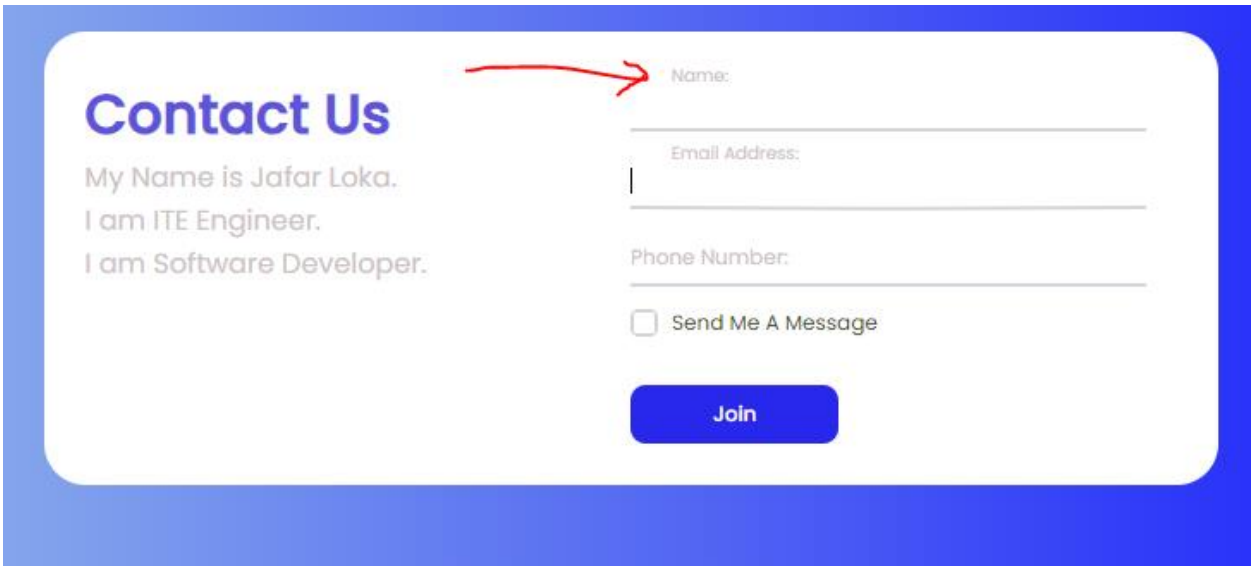
```
if(!input.value) {
          tl.fromTo(line,
            { attr: { d: start } },
            { attr: { d: end },  ease: "power2.in", duration: 0.5, }
          );

          tl.to(line, { attr: { d: start }, ease: "elastic.out(2,0.3)", });
        }
```

****

Always When We Scale, transform, ...etc Any Thing It Happened From Middle Point:

And To Change This Behavior We Can Change transform-origin to left:

```
.placeholder {
  position: relative;
  z-index: 1;
  pointer-events: none;
  padding: 0.5rem 0;
  font-size: 0.75rem;
  opacity: 0.75rem;
  transform-origin: left;
}
```



****

Note (To Remember): If We Want Our Animation To Start With Previous One At The Same Time, Just We Add '<' To The Option:

```
tl2.to('.checkbox-label', { color: '#000095'}, '<');

tl2.to('.checkbox-label', { color: '#999494' }, '<');
```

****

Note: If We Have Cursor-problem With Any Input Element, May The Solution By Adding Big Value For **z-index** Attribute.

****

Note (To Remember): If We Want Our Transform Point To Change We Can Change The **transform-origin** Property To Any Valid Value.

```
#character{
  position: absolute;
  transform: scale(0.95) rotateY(40deg);
  transform-origin: bottom;
  top: 40%;
  left: -5%;
}
```

****

May In Some Situations We Need To Change The Transform Origin To Specific Part Of SVG-Element, And Delay The Repeat Of GSAP:

```
gsap.set('#eye', { transformOrigin: 'center' });
gsap.fromTo('#eye', { scale: 1 }, {
        scale: 0.5, repeat: -1, duration: 1.5, yoyo: true, ease: "power3.out", repeatDelay: 0.25
});
```

****

Note (To Remember): If The Width, Height, Top, Left, ...etc, y, x Not Worked, Then Just Change The Display To inline-block.

****

To Init The Barba We Can Do The Following:

Note 1: To Make The Page Loader Wait Until Finish From Animation, We Use **this.async()**, And **onComplete()-callback**.

```
barba.init({
        transitions:[
            // Showcase Transitions
            {
                name: 'default-J-L-01',

                leave(data) {
                    let current = data.current.container;
                    const done = this.async();
                    gsap.fromTo(
                        current,
                        { opacity: 1, y: 0},
                        // For onComplete, it Can Be:
                            // 1--> Simple Reference Function.
                            // 2--> Arrow Function
                        { opacity: 0, y: 50, duration: 1.5, onComplete: () =>
done()},
                    );

                },
                enter(data) {
                    const next = data.next.container;
                    const done = this.async();
                    gsap.fromTo(
                        next,
                        { opacity: 0, y: 50},
                        { opacity: 1, y: 0, duration: 1, onComplete: done }
                    );

                }
            }
        ],
    });
```
****

The Problem With The Above Animation, Is That When We Click Fast, The Animations Will Not Worked Ok.

****

To Solve The Above Problem, We Use preventRunning: true:

```
barba.init({
        preventRunning: true, // this changed.
        transitions:[
            // Showcase Transitions
            {
                name: 'default-J-L-01',
                leave(data) {
                   // same code here
                },

                enter(data) {
                   // same code here
                }
            }
        ],
    });
```

****

To Solve The Problem Of Page Overflow The Current Screen, We Can Do That:

```
body {
    overflow: hidden;
}
```

****

If We Animate The Background That Use Gradient, May We Have A Problem Of Flush Colors, Because GSAP Not Know The First State Of Color:

```
tlEnter.to('body', { background: gradient }, '<')


const getGradient = (name) => {
        switch(name) {
            case 'handbag':
                return 'linear-gradient(260deg, #b75d62, #754d4f)';

            case 'boot':
                return 'linear-gradient(260deg, #5d8cb7, #4c4f70)';

            case 'hat':
                return 'linear-gradient(260deg, #b27a5c, #7f5450)'
        }
    }
```

****

The Solution Of Above Problem, Is To Use once-Function Of transitions-Array:

```
once(data) {
      const next = data.next.container;

      const done = this.async();

      const gradient = getGradient(data.next.namespace);

      gsap.set('body', { background: gradient});

      enterAnimation(next, done, gradient);
},
```

****

Note (To Remember): To Set The Number Of Column Of Grid-Element As The Screen Size With Range For Size Of Each Element Inside The Grid:

We Use Here, auto-fit for Number of Columns.

We Use Here, minmax(...) For Size Of Each Column.

```
.product-gallery {
    display: grid;
    padding: 0% 15%;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    align-items: stretch;
    gap: 2.5rem;
}
```

****

When We Use This Way For Page Transition, The Leave Will Be For The Handbag-Page, Not For The Product Page.

Note: We Use Sync To Run The Animation For Both Pages At The Same Time.

```
{
                name: 'product-transition',

                sync: true,

                from: { namespace: ['handbag']},

                to: { namespace: ['product'] },
}
```

****

Note (From Me): If We Don't Set The done-Function, Strange Behavior Will Occur:

```
barba.init({
        preventRunning: true,
        transitions:[
            // Showcase Transitions
            {
                name: 'default-J-L-01',

                once(data) {
                    const next = data.next.container;
                    const done = this.async();
                    const gradient = getGradient(data.next.namespace);
                    gsap.set('body', { background: gradient});
                    enterAnimation(next, done, gradient);
                },

                leave(data) {
                    let current = data.current.container;
                    const done = this.async();
                    leaveAnimation(current, done);
                },

                enter(data) {
                    const next = data.next.container;
                    const done = this.async();
                    const gradient = getGradient(data.next.namespace);
                    enterAnimation(next, done, gradient);
                }
            },

            // HandBag To Product Page Transition
            {
                name: 'product-transition',

                sync: true,

                from: { namespace: ['handbag']},

                to: { namespace: ['product'] },

                leave(data) {

                    let done = this.async();
```

```
                let current = data.current.container;

                productLeaveAnimation(current, done);
            },

            enter(data) {
                const done = this.async();

                const next = data.next.container;

                console.log("In The Enter Function Of Product Page
Transition: ", next);

                productEnterAnimation(next, done);
            }
        },

        // Product To HandPage Animation Test
        {
            name: 'product-to-handbag-animation',
            sync: true,
            from: { namespace: ['product'] },
            to: { namespace: ['handbag']},

            leave(data) {
                const done = this.async();

                const current = data.current.container;

                productLeaveAnimation(current, done);
            },

            enter(data) {
                const next = data.next.container;

                const done = this.async();

                const gradient = getGradient(data.next.namespace);

                enterAnimation(next, done, gradient);
            }
        }
    ],
});
```
****

```
const leaveAnimation = (current, done) =>{

        const product = current.querySelector('.img-container');

        const text = current.querySelector('.showcase-text');

        const circles = current.querySelectorAll('.circle');

        const arrow = current.querySelector('.showcase-arrow');

        return (
            tlLeave.fromTo(arrow,
                { opacity: 1, y: 0 },
                { opacity: 0, y: -200, onComplete: done }),

            tlLeave.fromTo(product,
                { opacity: 1, y: 0 },
                { opacity: 0, y:-100, onComplete: done }, '<'),

            tlLeave.fromTo(circles,
                { opacity: 1, y: 0 },
                { opacity: 0, y: -50, stagger: 0.15, ease: "back.out(4)",
duration: 1 }, '<'),

            tlLeave.fromTo(text, { opacity: 1, y: 0 }, { opacity: 0, y: -50 },
'<')
        );

    }

    const enterAnimation = (current, done, gradient) =>{

        const product = current.querySelector('.img-container');

        const text = current.querySelector('.showcase-text');

        const circles = current.querySelectorAll('.circle');

        const arrow = current.querySelector('.showcase-arrow');

        return (
            tlEnter.fromTo(arrow,
                { opacity: 0, y: 200 },
                { opacity: 1, y: 0, onComplete: done }),
```

```
            tlEnter.to('body', { background: gradient }, '<'),

            tlEnter.fromTo(product,
                { opacity: 0, y: -100 },
                { opacity: 1, y: 0, onComplete: done }, '<'),

            tlEnter.fromTo(circles,
                { opacity: 0, y: 50 },
                { opacity: 1, y: 0, stagger: 0.15, ease: "back.out(4)", duration:
1 }, '<'),

            tlEnter.fromTo(text, { opacity: 0, y: -50 }, { opacity: 1, y: 0 },
'<')
        );

    }

    const getGradient = (name) => {
        switch(name) {
            case 'handbag':
                return 'linear-gradient(260deg, #b75d62, #754d4f)';

            case 'boot':
                return 'linear-gradient(260deg, #5d8cb7, #4c4f70)';

            case 'hat':
                return 'linear-gradient(260deg, #b27a5c, #7f5450)'
        }
    }

    const productEnterAnimation = (next, done) => {

        tlProductEnter.fromTo(next,
            { opacity: 0, y: '100%' },
            { opacity: 1, y: 0, onComplete: done }
        );

        tlProductEnter.fromTo('.card',
            { opacity: 0, y: 50 },
            { opacity: 1, y: 0, stagger: 0.1, onComplete: done },
            '<50%'
        );

    }
```

```
    // ! Note: Here The Leave For HandBag Page.
    const productLeaveAnimation = (current, done) => {
        tlProductLeave.fromTo(current,
            { opacity: 1, },
            { opacity: 0, onComplete: done }
        );
    }
```
****