EXERCISES QUE TRABAJAREMOS EN EL CUE

- EXERCISE 1: UTILIZANDO TRANSICIONES DE CSS.
- EXERCISE 2: CREANDO ANIMACIONES CON CSS.

EXERCISE 1: UTILIZANDO TRANSICIONES DE CSS

En el siguiente ejercicio aprenderemos a realizar transiciones en CSS.

Como base, utilizaremos el hexágono creado en el CUE 7: SVG, el cual tenía el siguiente código:

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 150 150"
width="200" height="200">
   <defs>
      <style>
         .hexagono {
         fill: pink;
         stroke-width: 6px;
         stroke: yellow;
         }
      </style>
   </defs>
   <polygon class="hexagono" points="50 3,100 28,100 75, 50 100,3
75,3 25" />
</svg>
```


Imagen 1. Hexágono

En primer lugar, agregaremos una transición que cambie el color de los bordes (`stroke`) de la figura, cada vez que se posicione el cursor sobre ésta.

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 150 150"
width="200" height="200">
   <defs>
      <style>
         .hexagono {
         fill: pink;
         stroke-width: 6px;
         stroke: yellow;
         transition: stroke 3s;
         }
         .hexagono:hover {
         stroke: black;
         }
      </style>
   </defs>
   <polygon class="hexagono" points="50 3,100 28,100 75, 50 100,3
75,3 25" />
</svg>
```

Ahora, si cargamos esta imagen SVG con la transición agregada a una página web, comprobaremos que efectivamente, al posicionar el cursor sobre ella, el color de los bordes cambiará a negro por 3 segundos.

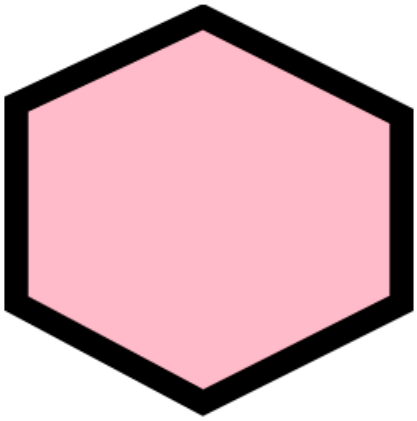

Imagen 2. Hexágono con transición.

Como puede observarse en el código, hemos utilizado el selector de clase `.hexagono` para agregar la propiedad `transition`, sobre la que se agregará el efecto y el tiempo que durará la transición. Por otro lado, más abajo en el código, se insertó la pesudo-clase `:hover`, la cual nos permitirá definir el momento en que comenzará la transición. En este caso, comenzará cada vez que se posicione el cursor sobre la imagen, y durará 3 segundos.

También podemos definir más de una transición a la vez. Haremos que cuando se posicione el cursor sobre la imagen, su color de relleno y de bordes cambie, y que además, se mueva algunos pixeles hacia la derecha.

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 150 150"
width="200" height="200">
   <defs>
      <style>
         .hexagono {
         fill: pink;
         stroke-width: 6px;
         stroke: yellow;
         transition: stroke 3s, fill 3s, transform 3s;
         }
         .hexagono:hover {
         stroke: black;
         fill: greenyellow;
         transform: translate(30px,0px);
         }
      </style>
   </defs>
   <polygon class="hexagono" points="50 3,100 28,100 75, 50 100,3
75,3 25" />
</svg>
```

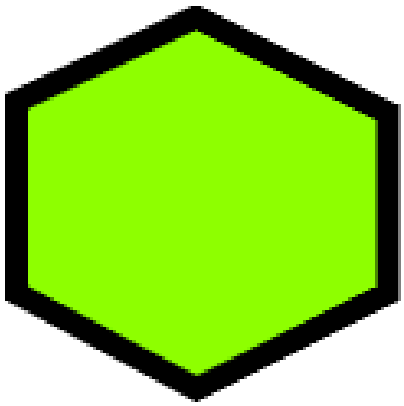

Imagen 3. Hexágono con múltiples transiciones.

La propiedad **transition-delay**, permite agregar un retraso a la ejecución de los efectos de transición. Probémosla agregando un retraso de 1 segundo a las transiciones ya agregadas.

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 150 150"
width="200" height="200">
   <defs>
      <style>
         .hexagono {
         fill: pink;
         stroke-width: 6px;
         stroke: yellow;
         transition: stroke 3s, fill 3s, transform 3s;
         transition-delay: 1s;
         }
         .hexagono:hover {
         stroke: black;
         fill: greenyellow;
         transform: translate(30px,0px);
         }
      </style>
   </defs>
   <polygon class="hexagono" points="50 3,100 28,100 75, 50 100,3 75,3
25" />
</svg>
```

Las transiciones aplicadas anteriormente, escribiendo solo una línea de código, se pueden dividir en dos: `transition-property`, la cual permite seleccionar qué propiedades serán afectadas por una transición; y `transition-duration`, la cual permite definir la duración de dichas transiciones.

Como en el ejemplo anterior, las tres transiciones tienen la misma duración, y corresponden a todas las propiedades dentro de la pseudoclase `:hover`. Podemos reescribir el código de la siguiente manera:

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 150 150"
width="200" height="200">
   <defs>
      <style>
         .hexagono {
         fill: pink;
         stroke-width: 6px;
         stroke: yellow;
         transition: stroke 3s, fill 3s, transform 3s;
         transition-delay: 1s;
         }

         .hexagono:hover {
         stroke: black;
         fill: greenyellow;
         transform: translate(30px,0px);
         }

      </style>
   </defs>

   <polygon class="hexagono" points="50 3,100 28,100 75, 50 100,3 75,3
25" />

</svg>
```

Otra propiedad muy interesante es `transition-timing-function`, la cual permite la aceleración y movimiento de una transición. Ingresaremos a MDN Web Docs, y generaremos nuestra propia curva de Bézier, que nos permitirá definir su velocidad momento a momento.

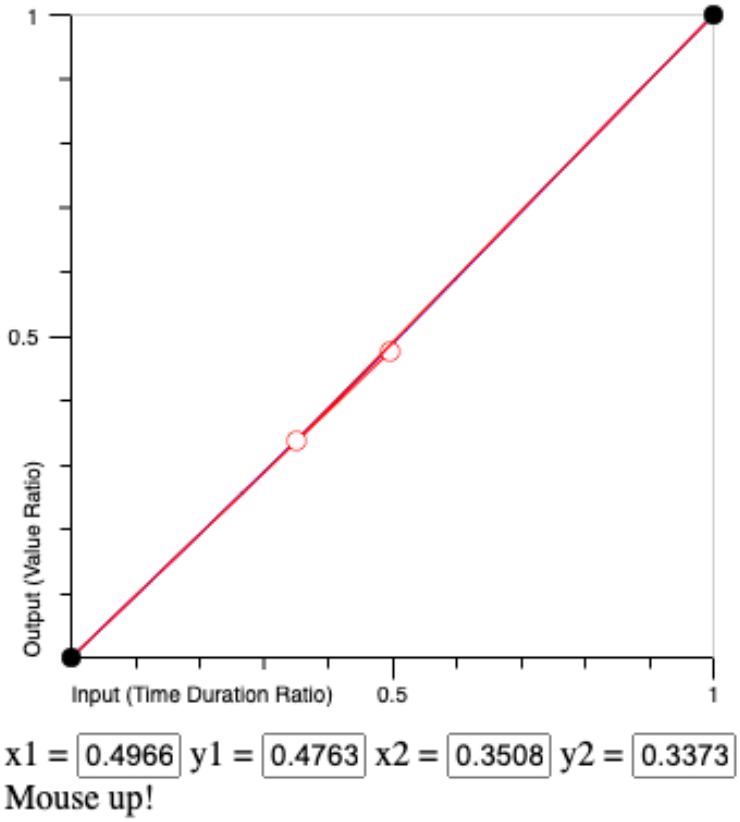

Imagen 4. Generador de curvas de bézier.

Ahora, aplicaremos esta propiedad a nuestro código.

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 150 150"
width="300" height="200">
   <defs>
      <style>
         .hexagono {
         fill: pink;
         stroke-width: 6px;
         stroke: yellow;
         transition-property: all;
         transition-duration: 3s;
         transition-delay: 1s;
         transition-timing-function: cubic-bezier(0.4966, 0.3508,
0.4763, 0.3373);
         }
```

```
15
16          .hexagono:hover {
17          stroke: black;
18          fill: greenyellow;
19          transform: translate(30px,0px);
20          }
21
22      </style>
23    </defs>
24    <polygon class="hexagono" points="50 3,100 28,100 75, 50 100,3 75,3
25 25" />
26 </svg>
```

En este ejercicio hemos aprendido a crear transiciones en CSS. Te invitamos a seguir realizando las actividades correspondientes a este CUE, y a continuar avanzando en el curso.

EXERCISE 2: CREANDO ANIMACIONES CON CSS

En este ejercicio aprenderemos a animar el contenido de nuestras páginas web, utilizando CSS.

A continuación, crearemos un cuadrado utilizando un div, y agregándole estilo con CSS.

```
 1 <!DOCTYPE html>
 2 <html lang="es">
 3    <head>
 4       <meta charset="UTF-8">
 5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
 6       <meta name="viewport" content="width=device-width, initial-
 7 scale=1.0">
 8       <title>Document</title>
 9    </head>
10
11    <style>
12       .cuadrado {
13       width: 200px;
14       height: 200px;
15       background-color: blueviolet;
16       }
17    </style>
18
19    <body>
```

```
20        <div class="cuadrado"></div>
22    </body>
23 </html>
```

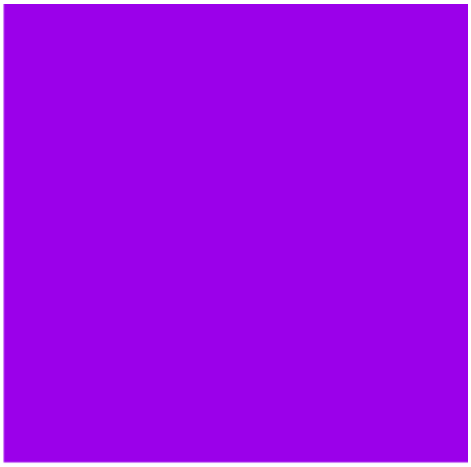

Imagen 1. Cuadrado.

Luego, agregaremos un `@keyframes` con nombre desplazar, el cual dividiremos en 5 etapas: `0%`, `25%`, `50%`, `75%` y `100%`. También, luego de cada porcentaje, incluiremos dentro de llaves, el estilo a aplicarse.

```
1  @keyframes desplazar {
2          0% {
3                  margin: 0
4          }
5          25% {
6                  margin-left: 100px
7          }
8          50% {
9                  margin-top: 100px
10         }
11         75% {
12                 margin-left: 0
13         }
14         100% {
15                 margin-top: 0
16         }
17 }
```

Además, debemos incluir la propiedad *animation* a la clase `.cuadrado`, con los valores: `desplazar 10s ease-in-out infinite (animation-name animation-duration animation-timing-function animation-iteration-count)`.

```
.cuadrado {
        width: 200px;
        height: 200px;
        background-color: blueviolet;
        animation: desplazar 10s ease-in-out infinite;
}
```

Una vez guardados los cambios, volveremos al navegador y observaremos que el cuadrado morado que hemos creado se desplazará infinitamente en sentido horario.

Es importante recordar, que existen otras propiedades asociadas a las animaciones que podemos explorar por nuestra cuenta, como: `animation-delay`, `animation-direction` y `animation-fill-mode`.