Guía de estudio - Animaciones CSS

 ¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase, además de profundizar temas adicionales que complementan aquellos vistos en la clase.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Introducción

Con CSS podemos dar vida y dinamismo a nuestros elementos HTML utilizando diferentes técnicas, entre ellas, se encuentran las transiciones, transformaciones y animaciones.

Estas propiedades permiten añadir efectos visuales y de movimiento a nuestros diseños web, lo que resulta especialmente útil para mejorar la experiencia de usuario y hacer nuestros sitios más atractivos y funcionales.

A continuación, veremos en detalle cada una de las siguientes técnicas.

- Transiciones (Transition).
- Transformaciones (Transform).
- Animaciones (Animation)

Transiciones (Transitions)

La propiedad transition permite crear animaciones de transición suave entre dos estados de un elemento; por ejemplo, al cambiar el color de fondo de un botón al pasar el cursor sobre él, se puede crear una transición suave que hace que el cambio de color sea más atractivo para el usuario.

Veamos este mismo ejemplo en código:

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de animación CSS</title>
  <meta charset="utf-8">
  <style>
    button {
      background-color: #0077FF;
      color: white;
      padding: 10px 20px;
      font-size: 16px;
      border: none;
      border-radius: 5px;
      transition: background-color 0.3s ease-out;
    }

    button:hover {
      background-color: rgb(255, 140, 0);
    }
  </style>
```

```
                                    </head>
<body>
  <button> Pasa el cursor del mouse sobre el botón </button>
</body>
</html>
```

En el archivo HTML, creamos un botón simple que tiene un texto dentro, mientras que en el archivo CSS, aplicamos estilos al botón.

Esto incluye el color de fondo, el color del texto, el relleno, el tamaño de fuente, el borde y el radio de borde. También se agrega una propiedad transition, que indica al navegador cómo se debe animar el cambio en el color de fondo del botón.

El secreto de la animación está tras la propiedad transition, esta toma cuatro valores:

1. La propiedad que se va a animar (background-color en este caso).
2. La duración de la transición (0.3 segundos en este caso).
3. La función de tiempo que se utiliza para la transición (ease-out en este caso).
4. Retraso. En este caso no la especificamos, pero se puede agregar un tiempo más para demorar el inicio de la transición.

A continuación profundizaremos en estas propiedades.

Propiedades de una transición

En el ejemplo anterior, cuando utilizamos los siguientes valores para la transición: `background-color 0.3s ease-out` realmente utilizamos múltiples propiedades por separado. Estas son:

- transition-property: background-color;
- transition-duration: 0.3s
- transition-timing-function: ease-out
- transition-delay: 0s

Sin embargo, es bastante más corto escribir:`transition: background-color 0.3s ease-out 1s;`

Transition-Property

La transición se puede aplicar a todas las propiedades simultáneamente o solo a algunas. Veamos un ejemplo con una caja, que al pasar el mouse por encima cambie todas las propiedades simultáneamente.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Ejemplo de transición con múltiples propiedades</title>
    <style>
      .box {
        width: 100px;
        height: 100px;
        background-color: blue;
        transition: all 3s;
      }

      .box:hover {
        width: 200px;
        height: 200px;
        background-color: red;
      }
    </style>
  </head>
  <body>
    <div class="box"></div>
  </body>
</html>
```

Ahora modifiquemos la propiedad transition para que únicamente se aplique al ancho de la caja utilizando: `transition: width 3s;` con este veremos que las propiedades color y alto cambian inmediatamente, pero la propiedad ancho se demora 3 segundos en cambiar.

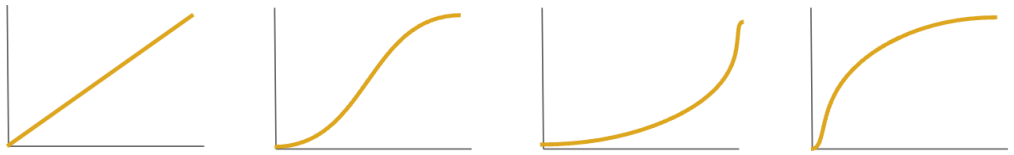
Si queremos una transición sobre todas las propiedades simultáneamente ocuparemos el valor `all`

Timing-duration

La duración es bastante sencilla de explicar, consiste en la cantidad de segundos o milisegundos que durará la transición. Podemos utilizar `transition-duration: 1s` o `transition-duration: 1000ms` o incluir la duración directamente en la transición como lo hemos hecho hasta ahora: `transition: all 1s;`

Timing-function: La función de tiempo

La función de tiempo es una propiedad que permite controlar la curva de velocidad de la transición. Existen diversos tipos de curvas para la transición de animaciones, donde dos ejemplos comunes son la curva lineal y la curva ease.



Linear
uniforme

Ease (por defecto)
Lento al principio y al final, rápido en la mitad

Ease in
Lento al principio, rápido al final.

Ease out
Rápido al principio, lento al final.

Imagen 1. Curvas de la función tiempo
Fuente: Desafío Latam

La curva lineal avanza en cada paso la misma cantidad, mientras que la curva ease produce una transición más natural y suave, empezando lentamente, acelerando y desacelerando hacia el final. La elección de la función tiempo adecuada depende del efecto que se desee lograr en la animación.

Tipos de función de tiempo

Función de tiempo	Descripción
ease	Inicia lentamente, luego acelera y luego desacelera hacia el final.
ease-in	Inicia lentamente y luego acelera hacia el final.
ease-out	Inicia rápidamente y luego desacelera hacia el final.
ease-in-out	Inicia lentamente, luego acelera y luego desacelera hacia el final.
linear	Avanza a una velocidad constante durante toda la transición.
step-start	La transición comienza inmediatamente.
step-end	La transición termina inmediatamente.

steps(n)	Divide la transición en n pasos, donde n es un número entero. Cada paso tiene la misma duración y se completa al mismo tiempo.
steps(n, start)	Divide la transición en n pasos, donde n es un número entero. El valor start indica si cada paso comienza o termina en el momento de la transición.

Tabla 1. Funciones de tiempo
Fuente: Desafío Latam

Probemos un ejemplo similar al anterior cambiando la función de tiempo por steps, a steps le pasaremos el valor 5.

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de animación CSS</title>
  <meta charset="utf-8">
  <style>
    button {
      background-color: #0077FF;
      color: white;
      padding: 10px 20px;
      font-size: 16px;
      border: none;
      border-radius: 5px;
      transition: background-color 1s steps(5);
    }

    button:hover {
      background-color: rgb(255, 140, 0);
    }
  </style>
</head>
<body>
  <button> Pasa el cursor del mouse sobre el botón </button>
</body>
</html>
```

Si ahora probamos pasando el cursor del mouse veremos que el cambio de color es pausado por intervalos, donde en un total de 5 intervalos cambia de color.

Actividad 1

- Prueba cambiando el valor de step por un valor distinto, ¿qué efecto tiene?
- Prueba cambiando la función temporizador por las siguientes funciones.
    - ease
    - ease-in
    - ease-out
    - linear

Puede ser difícil detectar los cambios, prueba aumentando la duración de la transición para que sea más sencillo.

Delay

Con la propiedad transition-delay podemos controlar cuánto se demora la transición en empezar. Al igual que la duración, puede ser en segundos o milisegundos.

Resumen de las propiedades

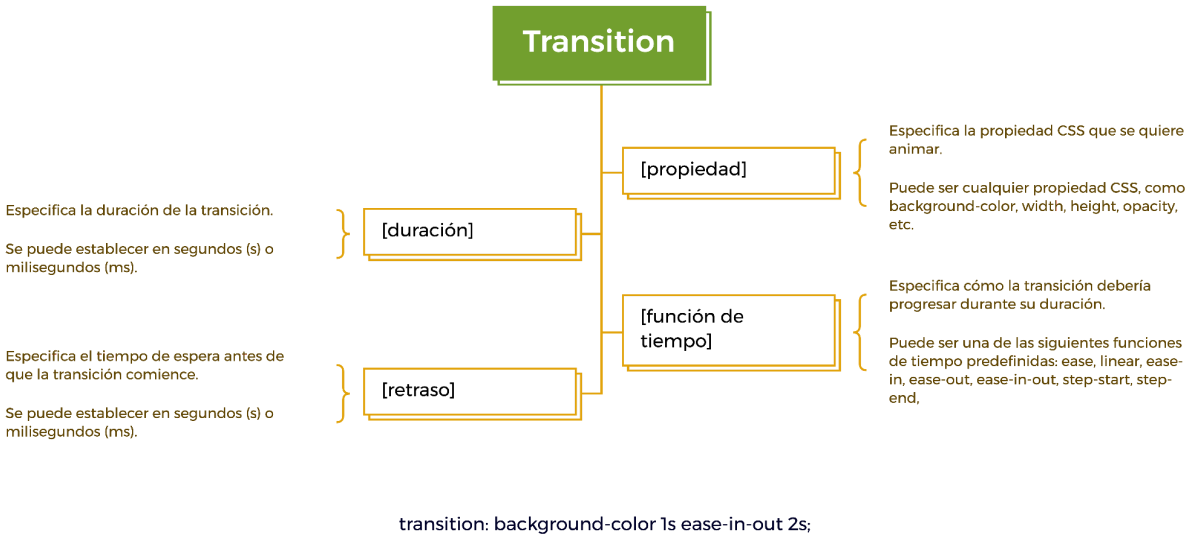


Imagen 2. Resumen de transiciones
Fuente: Desafío Latam

Transformaciones (Transform)

Las transformaciones en CSS se utilizan para modificar las propiedades geométricas de un elemento, como su tamaño, posición y rotación, utilizando la propiedad **transform**.

Esta tabla resumen muestra las transformaciones que podemos aplicar.

Transformación	Descripción
translate()	Mueve un elemento en la dirección y distancia especificada.
translateX()	Mueve un elemento horizontalmente en la distancia especificada.
translateY()	Mueve un elemento verticalmente en la distancia especificada.
scale()	Escala un elemento en la cantidad especificada.

scaleX()	Escala horizontalmente un elemento en la cantidad especificada.
scaleY()	Escala verticalmente un elemento en la cantidad especificada.
rotate()	Rota un elemento en el ángulo especificado.
skew()	Sesga un elemento en los ángulos especificados.
skewX()	Sesga horizontalmente un elemento en el ángulo especificado.
skewY()	Sesga verticalmente un elemento en el ángulo especificado.
matrix()	Combina varias transformaciones en una sola matriz.

Tabla 2. Transformaciones
Fuente: Desafío Latam

Veamos un ejemplo con rotación **rotate** y **scale**:

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de transformación en CSS</title>
    <style>
        .box {
            width: 100px;
            height: 100px;
            background-color: blue;
            transform: rotate(45deg) scale(1.5);
            /* La transformación rota y escala la caja */
        }
    </style>
</head>
<body>
    <div class="box"></div>
    <!-- La caja se transforma según las propiedades del estilo -->
</body>
</html>
```

Al abrir la página veremos que aparece una caja girada 45 grados, también veremos que es un 50% más grande debido a scale(1.5).

Ver esto a simple vista puede ser complejo, pero revisemos un ejercicio donde agreguemos 2 cajas, una normal y otra con transformación.

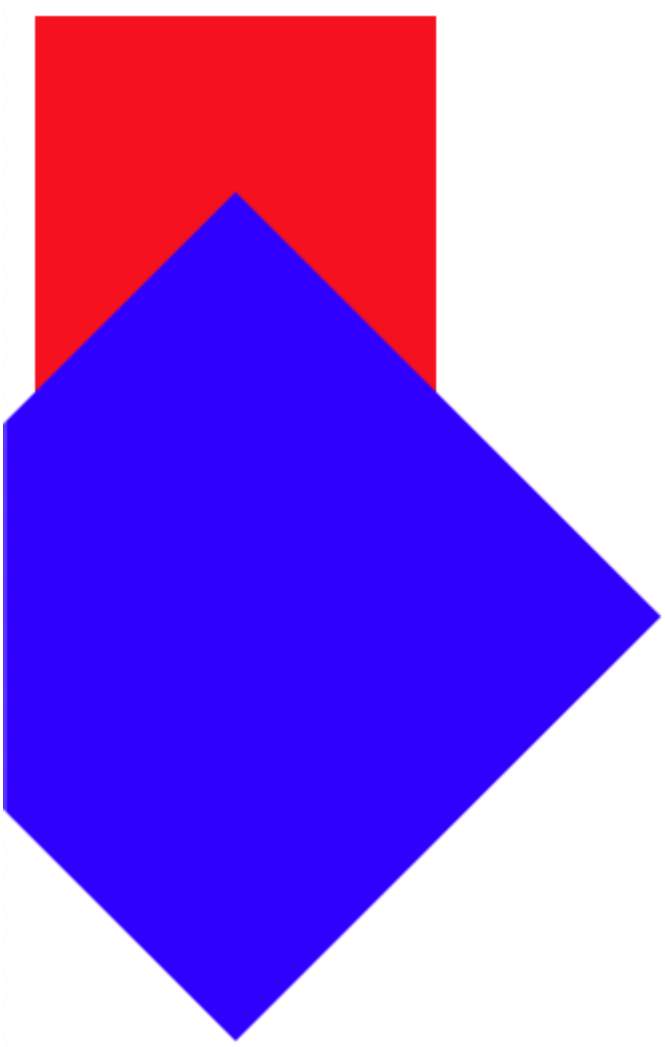


Imagen 3. Ejercicio
Fuente: Desafío Latam

Actividad 2

- Mover una caja hacia la derecha y hacia abajo: Crea una caja rectangular y muévela 50 píxeles hacia la derecha y 50 píxeles hacia abajo utilizando la propiedad transform con la función translate().

- Escalar una imagen al doble de su tamaño original: Crea un elemento div con una imagen de fondo y escala la imagen al doble de su tamaño original utilizando la propiedad transform con la función scale().

- Rotar un texto 180 grados: Crea un elemento p con un texto y rota el texto 180 grados utilizando la propiedad transform con la función rotate().

Transformaciones con transiciones

Podemos combinar transformación y transiciones para lograr efectos interesantes, veamos un ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de transición con transformación en CSS</title>
    <style>
        .box {
            width: 100px;
            height: 100px;
            background-color: blue;
            transform: rotate(0deg);
            transition: transform 1s;
            /* La transición se aplicará a la transformación */
        }
        .box:hover {
            transform: rotate(45deg);
            /* Al pasar el ratón por encima, se rota la caja */
        }
    </style>
</head>
<body>
    <div class="box"></div>
    <!-- La caja girará suavemente al pasar el ratón por encima -->
</body>
</html>
```

Al pasar el mouse por encima de la caja veremos que se rota.

Actividad 3

1. Escalar una imagen al doble de su tamaño original al pasar el ratón por encima:

- Crea un elemento div con una imagen de fondo y escala la imagen al doble de su tamaño original cuando el usuario pase el ratón por encima del elemento. Para lograr esto, se debe utilizar la propiedad transform con la función scale() y la propiedad transition para hacer que la transición sea suave y gradual.

2. Rotar una caja 360 grados cuando se mueve el mouse sobre ella.

- Crea una caja rectangular y haz que rote 360 grados cuando el usuario haga clic en ella. Para lograr esto, se debe utilizar la propiedad transform con la función rotate() y la propiedad transition para hacer que la rotación sea suave y gradual.

3. Hacer que un texto se desplace hacia la derecha al pasar el ratón por encima.

- Crea un elemento p con un texto y haz que se desplace hacia la derecha cuando el usuario pase el ratón por encima del elemento. Para lograr esto, se debe utilizar la propiedad transform con la función translate() y la propiedad transition para hacer que el desplazamiento sea suave y gradual.

Animaciones (animations)

La propiedad animation de CSS nos permite crear animaciones avanzadas y personalizadas en nuestros sitios web.

Una de las características más útiles de esta propiedad es la posibilidad de definir **keyframes**, que son puntos clave en la animación donde se establecen los valores específicos de las propiedades CSS. Gracias a los keyframes, podemos tener un control preciso sobre cómo se comporta la animación en cada momento, lo que nos permite crear efectos complejos.

En esta ocasión, nos centraremos en entender cómo funcionan los keyframes y cómo podemos aprovecharlos para crear animaciones dinámicas y sorprendentes. Veamos un ejemplo:

```
<!DOCTYPE html>
<html>
<head>
    <title>Ejemplo de animaciones en CSS</title>
    <style>
        .box {
            width: 100px;
            height: 100px;
            background-color: blue;
            position: relative;
            animation-name: example;
            animation-duration: 2s;
            animation-iteration-count: infinite;
        }
        @keyframes example {
            from { left: 0px; }
```

```
                                                to { left: 200px; }
            /* La animación moverá la caja de 0px a 200px a la izquierda */
        }
    </style>
</head>
<body>
    <div class="box"></div>
    <!-- La caja se anima con la propiedad animation definida -->
</body>
</html>
```

En las animaciones hay muchos conceptos claves, en esta tabla los resumimos.

Concepto	Descripción
Keyframes	Los keyframes definen los diferentes estados de la animación.
Duración animation-duration	La duración de la animación se refiere al tiempo que tarda en completarse la animación.
Delay animation-delay	El delay se refiere al tiempo que tarda en comenzar la animación después de que se ha activado.
Número de iteraciones animation-iteration-count	Es el número de veces que se repetirá la animación. Puede ser una cantidad definida o infinito.
Dirección	La dirección de la animación se refiere a la forma en que se reproduce la animación.
Timing function	La función de tiempo define cómo se aplican los cambios de estilo a lo largo de la duración de la animación.
Fill mode	El fill mode se refiere a cómo se mantienen los estilos después de que se completa la animación.
Eventos de animación	Los eventos de animación permiten ejecutar funciones específicas en JavaScript en diferentes momentos de la animación.

Tabla 3. Conceptos clave.

Fuente: Desafío Latam.

Keyframes

Los keyframes son puntos clave en una animación que nos permiten establecer los valores de las propiedades CSS en momentos específicos, donde al utilizar varias de estas keyframes en una animación, se crea la ilusión de movimiento suave entre cada punto.

Por ejemplo, si queremos animar un objeto en CSS para que se mueva de un punto A a un punto B, podemos definir un keyframe para la posición inicial en A y otro para la posición final en B. Con esto se calculará automáticamente los fotogramas intermedios necesarios para crear una transición suave entre ambos puntos.

```
@keyframes example {
    from { left: 0px; }
    to { left: 200px; }
}
```

En el primer ejemplo que vimos los keyframes eran `from` (desde) y `to` (hasta) y la animación consistía en el movimiento desde el primer punto hasta el segundo.

Duración y retraso

Para animaciones los conceptos de duración y retraso son los mismos que para transiciones. Con `animation-duration` controlamos cuánto tiempo se demora en completarse la animación y con `animation-delay` cuánto se demora en empezar.

Por ejemplo, podríamos agregar:

```
animation-duration: 2s;
animation-delay: 1s;
```

El código anterior retrasa el comienzo de la animación 1 segundo y haría que cada iteración de la animación durará 2 segundos.

Dirección

La propiedad `animation-direction` nos permite hacer una animación reversible y de esta forma lograr una animación más fluida de forma sencilla.

Veamos un ejemplo de una animación sin esta propiedad y luego agreguemos para ver la diferencia.

```
/* CSS */
.container {
```

```
                                        width: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
}

.text {
  font-size: 3em;
  animation: move 2s infinite;
}

@keyframes move {
  from {
    transform: translateX(0);
  }
  to {
    transform: translateX(200px);
  }
}
```

```
<!-- HTML -->
<div class="container">
  <h1 class="text">¡Hola, mundo!</h1>
</div>
```

En cada iteración de la animación vemos que la caja se traslada automáticamente, resultando en una animación que se ve extraña, esto podemos mejorarlo fácilmente agregando la propiedad `animation-direction: alternate`.

Nuestro CSS resultante quedaría así:

```
/* CSS */
.container {
  width: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
}

.text {
  font-size: 3em;
  animation: move 2s infinite;
  animation-direction: alternate;
```

```
                                    }

@keyframes move {
  from {
    transform: translateX(0);
  }
  to {
    transform: translateX(200px);
  }
}
```

Al recargar la página veremos que el movimiento de la animación continua en reversa una vez que termina, resultando en un efecto más natural.

Animaciones con múltiples frames

Hasta el momento hemos utilizado únicamente keyframes con 2 puntos claves, pero es posible utilizar múltiples para lograr animaciones más avanzadas:

```
/* CSS */
.container {
  width: 100%;
  height: 100vh;
  display: flex;
  justify-content: center;
  align-items: center;
}

.text {
  position: absolute;
  font-size: 48px;
  animation: move 4s ease infinite;
}

@keyframes move {
  0% {
    transform: translateX(0);
  }
  25% {
    transform: translateX(100px);
  }
  50% {
    transform: translateX(0);
    transform: rotate(180deg)
```

```
                                        ;
  }  75% {
    transform: translateX(-100px);
  }
  100% {
    transform: translateX(0);
  }
}
```

```
<!-- HTML -->
<div class="container">
  <h1 class="text">¡Hola, mundo!</h1>
</div>
```

Actividad 4

- En un archivo html nuevo debes crear una caja de 100px por 100px.
- Utilizando keyframes en el instante cero, cambia su color a verde.
- En el 50 cambia el color a rojo.
- En el 100 cambia el color azul.
- Haz la animación reversible.

Preguntas de cierre

1. **¿Qué propiedad de CSS se utiliza para crear una transformación que aumenta o disminuye el tamaño de un elemento?**
   a) transform-scale
   b) transform-rotate
   c) transform-translate
   d) transform-skew

2. **¿Qué propiedad se utiliza para especificar cuántas veces debe ejecutarse una animación?**
   a) animation-play-state
   b) animation-iteration-count
   c) animation-timing-function
   d) animation-fill-mode

3. **¿Cómo se especifica la duración de una animación?**
   a) usando la propiedad animation-duration

b) usando la propiedad animation-play-state
c) usando la propiedad animation-iteration-count
d) usando la propiedad animation-timing-function

4. **¿Cuál propiedad se utiliza para especificar la curva de velocidad de una animación?**
   a) animation-play-state
   b) animation-iteration-count
   c) animation-timing-function
   d) animation-fill-mode

5. **¿Cuál es la función predefinida de CSS que proporciona una curva de velocidad constante en una animación?**
   a) linear
   b) ease
   c) ease-in
   d) ease-out

6. **¿Qué propiedad de CSS se utiliza para definir la dirección de una animación?**
   a) animation-play-state
   b) animation-direction
   c) animation-fill-mode
   d) animation-iteration-count

Respuestas

Pregunta	Respuesta correcta
1	A
2	B
3	A
4	C
5	A
6	B