Tema 5

ANIMACIONES

Animaciones

La mayoría de los sitios web modernos incluyen alguna forma de interactividad. La interactividad suele integrarse en un sitio web mediante CSS y JavaScript.

En este capítulo, explicaremos cómo integrar la animación en una página web utilizando CSS y JavaScript. Aprenderemos a utilizar la propiedad de transformación de CSS para hacer que los elementos se muevan en una página web. También utilizarás CSS para crear una animación utilizando keyframe.

Animaciones

Las transformaciones CSS, las transiciones CSS y las animaciones CSS son tres especificaciones CSS distintas. Aunque los tres términos parecen hacer lo mismo -hacer que algo se mueva- las transiciones y animaciones CSS hacen que las cosas se muevan en el tiempo.

Las transiciones y animaciones permiten definir la transición entre dos o más estados de un elemento. Las transformaciones cambian la apariencia de un elemento a través de la traslación, la rotación, la escala y la inclinación, pero no tienen un componente temporal.

Roadmap

En esta presentación aprenderemos a realizar las siguientes actividades:

- Crear transformaciones con css.
- Crear transiciones con css.
- Crear una animación con css.
- Crear un menú hamburguesa

1. Transformaciones

1. Transformaciones

Desde los inicios de las Hojas de Estilo en Cascada (CSS), los elementos han sido rectangulares y orientados firmemente en los ejes horizontal y vertical.

A finales de la década de 2000, creció el interés por poder romper los grilletes de esa cuadrícula y transformar los objetos de forma interesante, y no sólo en dos dimensiones.

1. Transformaciones

Si alguna vez has colocado un objeto, ya sea de forma relativa o absoluta, ya has transformado ese objeto. Además, cada vez que has utilizado flotadores o trucos de margen negativo (o ambos), has transformado un objeto.

Todos esos son ejemplos de traslación, o el movimiento de un elemento desde donde normalmente aparecería a algún otro lugar. Con las transformaciones CSS, tienes una nueva forma de traducir elementos, y mucho más.

1. Transformaciones

Ya sea algo tan sencillo como girar un poco algunas fotografías para que parezcan más naturales, o crear interfaces en las que la información se pueda revelar volteando los elementos, o simplemente hacer interesantes trucos de perspectiva con las barras laterales, las transformaciones CSS pueden transformar tu forma de diseñar.

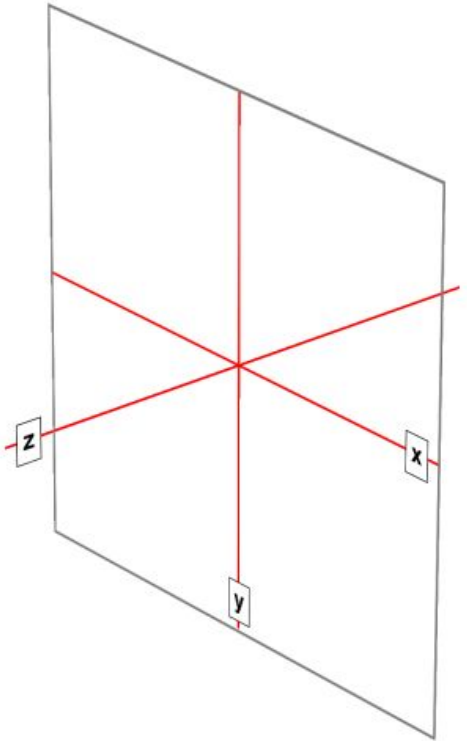
1.1. Sistemas de coordenadas

Antes de emprender este viaje, tomemos un momento para orientarnos. Hay **dos tipos de sistemas de coordenadas** que se utilizan en las transformaciones, y es una buena idea estar familiarizado con ambos.

- Sistema cartesiano
- Sistema esférico

1.1.1. Sistemas de coordenadas: cartesianas

El primero es el sistema de **coordenadas cartesianas**, o lo que suele llamarse el sistema de coordenadas **x/y/z**. Este sistema es una forma de describir la posición de un punto en el espacio utilizando dos números (para la colocación bidimensional) o tres números (para la colocación tridimensional). En CSS, el sistema utiliza tres ejes: el eje x, u horizontal; el eje y, o vertical; y el eje z, o de profundidad.



1.1.1. Sistemas de coordenadas: cartesianas

Para cualquier transformación 2D (bidimensional), sólo hay que preocuparse por los ejes x y y. Por convención, los valores **x positivos van hacia la derecha y los negativos hacia la izquierda**. Del mismo modo, los valores positivos de **y van hacia abajo a lo largo del eje de y, mientras que los valores negativos van hacia arriba a lo largo del eje de y**.

1.1.1. Sistemas de coordenadas: cartesianas

Por lo tanto, para mover un elemento hacia la izquierda y hacia abajo, se le daría un valor x negativo y un valor y positivo, así:

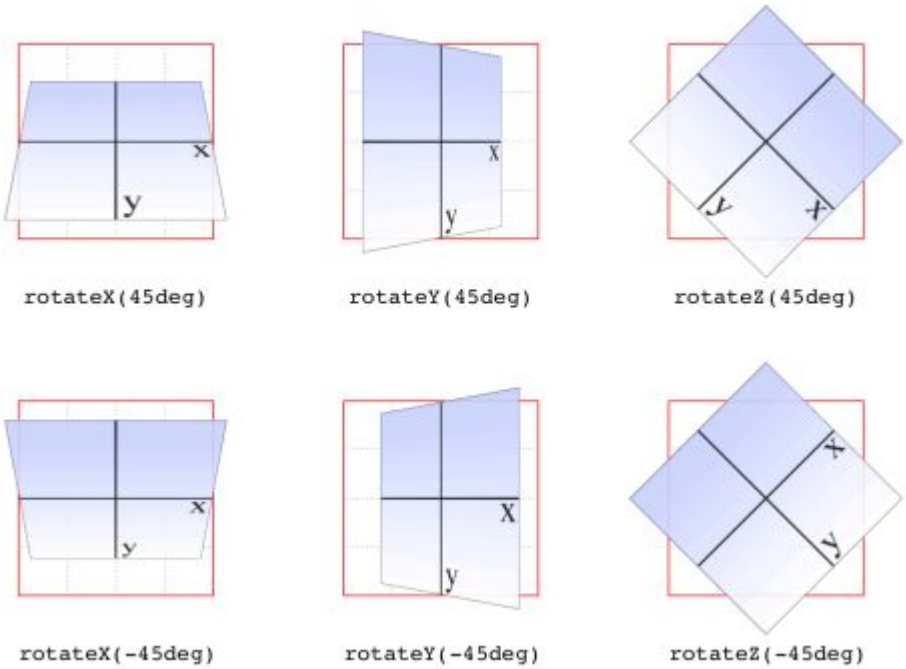
translateX(-5em) translateY(3em)

Este es un valor de transformación válido, como veremos en un momento. Su efecto es trasladar (mover) el elemento cinco ems a la izquierda y 3em hacia abajo.

1.1.2. Sistemas de coordenadas: esférico

El otro sistema de coordenadas utilizado en las transformaciones CSS es un **sistema esférico**, que describe **ángulos** en el espacio 3D.

Cuando se trata de rotaciones, una rotación 2D describe en realidad una **rotación alrededor del eje z**. Del mismo modo, las rotaciones en torno al eje **x inclinan el elemento hacia o lejos de ti**, y las rotaciones en torno al eje **y giran el elemento de lado a lado**.



1.2. Transform

En realidad sólo hay una propiedad que aplica las transformaciones, junto con algunas propiedades auxiliares que afectan exactamente a cómo se aplican las transformaciones.

1.2. Transform

Table 10–1 Transform Property Values

Method	Description	Example
matrix( )	A 2D transformation; accepts six values	transform: matrix(1, 0.5, -0.5, 1, 10, 0)
rotate( )	A 2D rotation; rotates an element a specified number of degrees clockwise or counter-clockwise	transform: rotate(10deg)
rotateX( )	A 3D rotation; rotates an element a specified number of degrees on the elements X-axis	transform: rotateX(40deg)
rotateY( )	A 3D rotation; rotates an element a specified number of degrees on the elements Y-axis	transform: rotateY(30deg)
rotateZ( )	A 3D rotation; rotates an element a specified number of degrees on the elements Z-axis	transform: rotateX(20deg)
scale( )	A 2D scale transformation; resizes an element	transform: scale(1.5)
scaleX( )	A 2D scale transformation; resizes an element on its X-axis	transform: scaleX(1.5)
scaleY( )	A 2D scale transformation; resizes an element on its Y-axis	transform: scaleY(1.5)
skew( )	A 2D skew transformation; moves the top and bottom or left and right sides a specified number of degrees	transform: skew(10deg, 10deg)
skewX( )	A 2D skew transformation for the X-axis of an element	transform: skewX(20deg)
skewY( )	A 2D skew transformation for the Y-axis of an element	transform: skewY(30deg)
translate( )	A 2D translation; moves the block element from its original position on the webpage	transform: translate(30px, 40px)
translateX( )	A 2D translation; moves the block element from its original position on the webpage	transform: translateX(30px)
translateY( )	A 2D translation; moves the block element from its original position on the webpage	transform: translateY(40px)

1.2.1. Transform-origin

La propiedad CSS transform-origin establece el origen de las transformaciones de un elemento.

Un ejemplo:

https://developer.mozilla.org/en-US/docs/Web/CSS/transform-origin

1.2.2. Transform-style

La propiedad CSS transform-style establece si los hijos de un elemento se posicionan en el espacio 3D o se aplanan en el plano del elemento.

Ejemplo:

https://developer.mozilla.org/en-US/docs/Web/CSS/transform-style

2. Transiciones

2. Transiciones

Las transiciones CSS nos permiten animar las propiedades CSS desde un valor original a un nuevo valor en el tiempo cuando el valor de una propiedad cambia.

Normalmente, cuando el valor de una propiedad CSS cambia, cuando se produce un "evento de cambio de estilo" el cambio, es instantáneo. El nuevo valor de la propiedad sustituye a la antigua en los milisegundos que se tarda en repintar (o reflotar y repintar, si es necesario) el contenido afectado.

2. Transiciones

La mayoría de los cambios de valor parecen instantáneos, ya que tardan menos de 16ms en renderizarse. Incluso si los cambios tardan más, se trata de un solo paso de un valor al siguiente.

Por ejemplo, cuando se cambia el color de fondo al pasar por encima, el fondo cambia de un color a otro, sin transición gradual. **Las transiciones CSS nos permiten animar suavemente** las propiedades CSS desde un valor original a un nuevo valor a lo largo del tiempo, a medida que avanza el recálculo del estilo:

2. Transiciones

```
button {
        color: magenta;
        transition: color 200ms ease-in 50ms;
        }

button:hover {
        color: rebeccapurple;
        transition: color 200ms ease-out 50ms;
}
```

2. Transiciones

La propiedad CSS transition es una abreviatura de:

- Transition-property: Nombre de la propiedad CSS a animar
- Transition-duration: La duración de la animación (s o ms)
- Transition-timing-function: Cómo se interpola la animación a lo largo del tiempo (ease, linear, ease-in, ease-out, ease-in-out, etc.).
- Transition-delay: El valor inicial de la propiedad.

Las transiciones permiten definir la transición entre dos estados de un elemento. Los diferentes estados pueden definirse mediante pseudoclases como **:hover** o **:active** o establecerse dinámicamente mediante JavaScript.

2. Transiciones

Para crear la navegación desplegable utilizaremos las cuatro propiedades de transición de CSS:

```
nav ul li {
        transition-property: transform;
        transition-duration: 200ms;
        transition-timing-function: ease-in;
        transition-delay: 50ms;
        transform: scale(1, 0);
        transform-origin: top center;
}

nav ul:hover li {
        transform: scale(1, 1);
}
```

2. Transiciones

Las transiciones **se declaran junto con los estilos regulares de un elemento**. Siempre que una propiedad cambie, si se establece una transición en esa propiedad, el navegador aplicará una transición para que el cambio sea **gradual**.

Aunque la iniciación más común de una transición es el cambio de los valores de la propiedad de un estado por defecto a un estado hovered, las transiciones también funcionan si la propiedad se cambia añadiendo una clase, manipulando el DOM, o cambiando el estado de otra manera

2. Transiciones

Puedes declarar las propiedades de transición en el **estado inicial**, en el **estado cambiado**, o en **ambos** estados, el inicial y el cambiado.

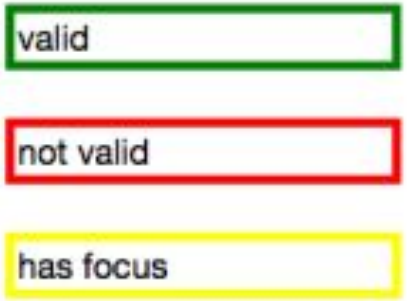
Si sólo declaras la transición en el estado inicial, cuando el estado cambie, pasará al estado cambiado tal y como indiques con las propiedades de transición de CSS. Cuando cambia de nuevo al estado inicial, el tiempo de transición se invierte. **Puedes anular esta transición inversa** por defecto declarando transiciones diferentes en los estados inicial y modificado.

2. Transiciones

Por estado inicial, me refiero a un estado que coincida con el elemento en la carga de la página. Esto podría ser un estado que el elemento siempre tiene, como las propiedades establecidas en un selector de elemento frente a un estado :hover para ese elemento, o un elemento editable de contenido que puede obtener :focus.

Un estado inicial también puede ser un estado temporal que puede cambiar, como una casilla de verificación :checked o un control de formulario :valid, o incluso una clase que se activa y desactiva

2. Transiciones




```
/* selector that matches element some of the time */
        input:valid {
                border-color: green;
        }
/* selector that matches element some of the time, when the prior selector does NOT match. */
        input:invalid {
                border-color: red;
        }
/* selector that matches element some of the time, whether the input is valid or invalid */
        input:focus {
                border-color: yellow;
        }
```

2. Transiciones

Generalmente, se desea declarar las propiedades de transición como mínimo en el selector que se aplica al elemento todo el tiempo.

En este caso no estátan claro: si la transición sólo se establece en el estado :inválido, el color pasará de rojo a verde cuando el estado de la entrada cambie de inválido a válido, y pasará de rojo a amarillo cuando una entrada inválida reciba el foco. Sin embargo, no pasará lentamente de verde a amarillo, o de amarillo a verde, cuando una entrada válida reciba o pierda el foco.

2. Transiciones

En otras palabras, si siempre quieres que la propiedad transite, es probable que quieras poner la transición en todos los estados. Las propiedades de transición que se utilizan son las del estado de destino; los nuevos valores de las propiedades de transición se utilizan para la transición al nuevo valor de la propiedad.

2. Transiciones

```
nav li ul {
        transition-property: transform;
        transition-duration: 200ms;
        transition-timing-function: ease-in;
        transition-delay: 50ms;
        transform: scale(1, 0);
        transform-origin: top center;
}
nav li:hover ul {
        transition-property: transform;
        transition-duration: 2s;
        transition-timing-function: linear;
        transition-delay: 1s;
        transform: scale(1, 1);
}
```

2.1. Transition-property

La propiedad **transition-property especifica los nombres de las propiedades CSS que desea transicionar**.

El valor de transition-property es una **lista de propiedades separada por comas**; la palabra clave none si no quiere que se transicione ninguna propiedad; o el valor por defecto all, que significa "transición de todas las propiedades transicionables". También puede incluir la palabra clave all dentro de una lista de propiedades separada por comas.

2.1. Transition-property

```
div {
        color: #ff0000;
        border: 1px solid #00ff00;
        border-radius: 0;
        transform: scale(1) rotate(0deg);
        opacity: 1;
        box-shadow: 3px 3px rgba(0, 0, 0, 0.1);
        width: 50px;
        padding: 100px;
        transition-property:  color,  border,  border-radius,
transform, opacity,
        box-shadow, width, padding;
        transition-duration: 1s;
}
```

```
div {
        color: #ff0000;
        border: 1px solid #00ff00;
        border-radius: 0;
        transform: scale(1) rotate(0deg);
        opacity: 1;
        box-shadow: 3px 3px rgba(0, 0, 0, 0.1);
        width: 50px;
        padding: 100px;
        transition-property: all;
        transition-duration: 1s;
}
```

2.2. Transition event: transitionend

Un evento **transitionend ocurre al final de cada transición**, en cada dirección, para cada propiedad que es transicionada durante cualquier cantidad de tiempo o después de cualquier retraso, ya sea que la propiedad sea declarada individualmente o sea parte de la declaración de todas. Para algunas declaraciones de propiedades aparentemente individuales, habrá varios eventos transitionend, ya que cada propiedad animable dentro de una propiedad abreviada obtiene su propio evento transitionend.

2.2. Transition event: transitionend

HTML:

```
<div class="caja" id="miCaja"></div>
<button id="boton">Iniciar transición</button>
```

CSS:

```
.caja {
    width: 200px;
    height: 200px;
    background-color: red;
    transition: all 2s ease-in-out;
}
.caja.moviendo {
    width: 300px;
    height: 300px;
    background-color: blue;
}
```

JAVA SCRIPT

```
const caja = document.getElementById('miCaja');
const boton = document.getElementById('boton');

// Función que escucha el evento transitionend
caja.addEventListener('transitionend', function(evento) {
        console.log('La transición ha terminado para la propiedad:', evento.propertyName);
});

// Cambiar la clase para iniciar la transición
boton.addEventListener('click', function() {
        caja.classList.toggle('moviendo');
});
```

2.3. Transition-duration

La propiedad transition-duration toma como valor una lista separada por comas de longitudes de tiempo, en segundos (s) o milisegundos (ms), que debe tomar la transición de los valores originales de la propiedad a los valores finales de la misma.

Si queremos que cada propiedad tenga una duración de transición diferente, tenemos que incluir un valor diferente separado por comas para cada nombre de propiedad declarado:

```
div {
        color: #ff0000;
        ...
        transition-property: color, border, border-radius, transform, opacity,
        box-shadow, width, padding;
        transition-duration: 200ms, 180ms, 160ms, 140ms, 120ms, 100ms, 1s, 2s;
}
```

2.4. Transition-timing

¿Quiere que su transición comience lenta y se acelere, que comience rápida y termine más lenta, que avance de manera uniforme, que salte por varios pasos o incluso que rebote? La función de temporización de la transición proporciona una forma de controlar el ritmo de la transición. La propiedad transition-timing-function describe cómo avanza la transición mientras se ejecuta.

Timing function	Definition
`ease`	Starts slow, then speeds up, then ends very slowly
`linear`	Proceeds at the same speed throughout transition
`ease-in`	Starts slow, then speeds up
`ease-out`	Starts fast, then slows down
`ease-in-out`	Similar to ease; faster in the middle, with a slow start but not as slow at the end
`cubic-bezier()`	Specifies a cubic-bezier curve

```
/* Keyword values */
    transition-timing-function: ease;
    transition-timing-function: ease-in;
    transition-timing-function: ease-out;
    transition-timing-function: ease-in-out;
    transition-timing-function: linear;
    transition-timing-function: step-start;
    transition-timing-function: step-end;
```

2.5. Transition delay

La inclusión de un **retraso de transición** con un número positivo de milisegundos (ms) o segundos (s) para retrasar la transición retrasará el inicio del efecto de transición. La unidad de tiempo, como s o ms, es necesaria.

Un valor de tiempo negativo para el retardo de transición hará que la transición comience inmediatamente, parcialmente a través de la transición. Un valor negativo de tiempo de transición que sea menor que la duración de la transición hará que la transición comience inmediatamente, parcialmente a través de la transición.

Recuerda...

Para que la transición se active, debes cambiar la propiedad que se está animando.

Esto generalmente se hace a través de pseudoclases, eventos de JavaScript o animaciones generadas dinámicamente.

Propiedades animables

Casi todas las propiedades CSS son animables con transiciones, pero algunas de las más comunes incluyen:

- Color (color)
- Ancho (width)
- Alto (height)
- Opacidad (opacity)
- Margen (margin)
- Relleno (padding)
- Fondo (background)
- Transformaciones (transform)
- Y todo lo que se os ocurra!!!

Transiciones múltiples

Puedes aplicar múltiples propiedades de transición separándolas con comas en la propiedad transition.

```
elemento {

transition:
propiedad-1 duración-1 tiempo-1,
propiedad-2 duración-2 tiempo-2;

}
```

3. Animación

3. Animación

Las transiciones CSS, tratadas en el capítulo anterior, permiten realizar animaciones sencillas.

Con las transiciones, las propiedades de un elemento cambian de los valores establecidos en un bloque de estilo a los valores establecidos en un bloque de estilo diferente a medida que el elemento cambia de estado con el tiempo en lugar de hacerlo instantáneamente.

3. Animación

Con las transiciones CSS, los estados de inicio y fin de los valores de las propiedades están controlados por los valores de las propiedades existentes y proporcionan poco control sobre cómo progresa la interpolación de los valores de las propiedades en el tiempo.

Las transiciones sólo nos permiten animar desde un valor inicial a un valor de destino y viceversa. **Las animaciones nos permiten decidir si una animación se repite y cómo**, y nos dan un control granular sobre lo que ocurre a lo largo de la animación a través de los KeyFrames.

3. Animación

La animación CSS nos permite animar los valores de las propiedades CSS en el tiempo utilizando keyframes. Al igual que las transiciones, la animación nos proporciona control sobre el retraso y la duración.

Con las animaciones CSS, podemos controlar el número de iteraciones, el comportamiento de la iteración, lo que ocurre antes de que comience la primera animación y el estado de las propiedades animadas después de que concluya la última iteración de la animación.

3. Animación

Mientras que las **transiciones provocan cambios implícitos en los valores de las propiedades**, las **animaciones se ejecutan explícitamente cuando se aplican las propiedades de los keyframe de la animación**.

Los valores de las propiedades establecidas en el elemento animado no tienen que ser necesariamente parte de la progresión de la animación. Por ejemplo, con las transiciones, pasar de negro a blanco sólo mostrará diferentes tonos de gris. Puedes hacer una transición a través de los tonos de gris, podrías convertir el elemento en amarillo, y luego animar de amarillo a naranja.

3.1. Keyframes

Los keyframes en CSS son una forma de definir animaciones complejas para elementos HTML, especificando etapas o "fotogramas clave" que representan cambios en el estado de un elemento en diferentes momentos de la animación. Esto permite crear transiciones de propiedades de manera fluida entre un punto de inicio y un punto final, o entre múltiples etapas intermedias.

3.1. Keyframes

Para animar un elemento, necesitamos establecer el nombre de una animación de keyframes; para ello, necesitamos una animación de keyframe con nombre.

Nuestro primer paso es definir esta animación de keyframe CSS reutilizable utilizando la regla **@keyframes**.

3.1. Keyframes

Para definir una animación CSS, declaramos una animación de keyframe reutilizables utilizando la regla @keyframes, dando un nombre a nuestra animación. El nombre que creamos se utilizará entonces dentro de un bloque de código de un selector CSS para adjuntar esta animación particular al (los) elemento(s) y/o pseudo-elemento(s) definido(s) por el (los) selector(es).

3.1. Keyframes

La regla @keyframes keyterm va seguida del identificador de la animación (el nombre que le das a tu animación para futuras referencias), seguido de llaves que abarcan la serie de bloques de keyframe.

Cada bloque de keyframe incluye uno o más selectores de keyframe. Los selectores de keyframe son posiciones de porcentaje de tiempo a lo largo de la duración de la animación; se declaran como porcentajes o con los términos clave desde o hasta

```
@keyframes animation_identifier {
        keyframe_selectorA {
                property1: value1a;
                property2: value2b;
        }
        keyframe_selectorB {
                property1: value1b;
                property2: value2b;
        }
}
```

3.2. Elementos de animación

Una vez que has creado una animación de keyframe, necesitas **aplicar** esa animación a los elementos y/o pseudo-elementos para que algo se anime realmente.

La animación CSS nos proporciona numerosas propiedades de animación para adjuntar una animación de keyframe a un elemento y controlar su progresión. Como **mínimo, necesitamos incluir el nombre de la animación para que el elemento se anime, y una duración** si queremos que la animación sea realmente visible.

3.2. Elementos de animación

La propiedad animation combina múltiples aspectos de la animación en una sola línea.

Puedes especificar name, duration, timing-function, delay, iteration-count, direction, fill-mode, y play-state.

No necesitas usar todas las propiedades, pero deben ir en el orden correcto si usas la abreviatura:

animation: [name] [duration] [timing-function] [delay] [iteration-count] [direction] [fill-mode] [play-state];

3.2. Elementos de animación

En total tiene las siguientes propiedades:

- Animation-name: el nombre del keyframe
- Animation-duration: por defecto 0s
- Animation-timing-function: aceleración (como cambiar de velocidad)
- Animation-delay: cuánto esperar antes de comenzar la animación
- Animation-iteration-count: cuántas veces se repite la animación
- Animation-direction: en qué dirección se ejecuta la animación
- Animation-fill-mode: cómo se comporta antes y después de ejecutarse
- Animation-play-state: si la animación está pausada o no

3.2.1. Elementos de animación: Animation-name

La propiedad animation-name toma como valor el nombre o la lista separada por comas de los nombres de la animación de fotogramas clave que desea aplicar a un elemento o grupo de elementos. Los nombres son los identificadores sin comillas que creó en su regla @keyframes.

```
div {
        animation-name: change_bgcolor, round, W;
}

@keyframes change_bgcolor
@keyframes round
@keyframes W
```

3.2.2. Elementos de animación: Animation-duration

La propiedad animation-duration define el tiempo que debe durar una sola iteración de la animación en segundos (s) o milisegundos (ms).

La propiedad animation-duration toma como valor el tiempo, en segundos (s) o milisegundos (ms), que debería tomar **completar un ciclo a través de todos los fotogramas clave**. Si se omite, la animación se aplicará con una duración de 0s, con el inicio de la animación y el final de la animación todavía se disparan a pesar de que la animación, teniendo 0s, es imperceptible. **Los valores negativos no son válidos**

3.2.3. Elementos de animación: Animation-timing

Similar a la propiedad transition-timing, la propiedad animation-timing describe **cómo progresará la animación durante un ciclo** de su duración, o iteración.

Valores típicos: *ease* | *linear* | *ease-in* | *ease-out* | *ease-in-out* | *step-start* | *step-end*

3.2.4. Elementos de animación: Animation-delay

La propiedad animation-delay define el **tiempo que el navegador espera después de que la animación se adjunte al elemento antes de comenzar la primera iteración de la animación**. El valor por defecto es 0s, lo que significa que la animación comenzará inmediatamente cuando se aplique.

Un **valor positivo retrasará el inicio de la animación** hasta que haya transcurrido el tiempo prescrito, indicado como valor de la propiedad animation-delay. Un valor negativo hará que la animación comience inmediatamente, pero lo hará a mitad de camino.

3.2.5. Elementos de animación: Animation-iteration-count

Simplemente incluyendo el nombre de la animación requerida hará que la animación se reproduzca una vez. Incluya la propiedad animation-iteration-count si desea **iterar** la animación más o menos de la vez por defecto.

Los valores pueden ser números o infinito. Valores:

- Números: 1, 3, etc. (cantidad específica de repeticiones).
- infinite: Se repite para siempre

3.2.6. Elementos de animación: Animation-direction

Con la propiedad animation-direction, puedes controlar si la animación progresa desde el fotograma clave del 0% hasta el fotograma clave del 100%, o desde el fotograma clave del 100% hasta el fotograma clave del 0%. Puedes controlar si todas las iteraciones progresan en la misma **dirección**, o establecer que cada ciclo de animación progrese en la dirección opuesta.

3.2.6. Elementos de animación: Animation-direction

- **Normal:** cada iteración de la animación progresa desde el fotograma clave del 0% hasta el fotograma clave del 100%.
- **Reverse:** cada iteración de la animación progresa en orden inverso al fotograma clave, progresando siempre desde el fotograma clave del 100% hasta el fotograma clave del 0%.
- **Alternate:** El valor alternate significa que la primera iteración debe proceder de 0% a 100%, y la segunda iteración (y cada ciclo par subsiguiente) debe invertir la dirección, procediendo de 100% a 0%.
- **Alternate-reverse:** similar al valor alternate, excepto que las iteraciones impares están en la dirección inversa, y las iteraciones pares de la animación están en la dirección normal.

3.2.7. Elementos de animación: Animation-fill-mode

La propiedad animation-fill-mode nos permite aplicar los valores de las propiedades de cualquier fotograma clave a un elemento desde el momento en que la animación se aplica a ese elemento hasta la expiración del retraso de la animación.

También nos permite mantener los valores de las propiedades de los fotogramas clave de 100% o hasta después de que el último ciclo de animación se haya completado. Valores:

- none (por defecto): No aplica estilos fuera del intervalo activo.
- forwards: Mantiene los estilos del último fotograma después de la animación.
- backwards: Aplica los estilos del primer fotograma antes de comenzar.
- both: Combina forwards y backwards.

3.2.8. Elementos de animación: Animation-play-state

Si se ajusta a la ejecución por defecto, la animación procede de forma normal. Si se establece en **pausa**, la animación se pausa.

Cuando se pone en pausa, la animación se sigue aplicando al elemento, deteniéndose en el progreso que había hecho antes de la pausa. Cuando se establece de nuevo en ejecución o se vuelve al valor por defecto de ejecución, se reinicia desde donde se dejó, como si el "reloj" que controla la animación se hubiera detenido y comenzado de nuevo. Valores:

- running (por defecto): La animación está activa.
- paused: Detiene la animación en el estado actual.

3.2.9. Valores por Defecto de animation

animation-name: none;

animation-duration: 0s;

animation-timing-function: ease;

animation-delay: 0s;

animation-iteration-count: 1;

animation-direction: normal;

animation-fill-mode: none;

animation-play-state: running;

Ejemplo animation

light-on → animation-name: Nombre de la animación definida en @keyframes.

2s → animation-duration: Duración de 2 segundos.

ease-in-out → animation-timing-function: Acelera y desacelera al inicio y al final.

1s → animation-delay: Espera 1 segundo antes de comenzar.

infinite → animation-iteration-count: Se repite infinitamente.

alternate → animation-direction: Alterna entre normal y reverso.

forwards → animation-fill-mode: Mantiene el último fotograma después de cada iteración.

```
.bombilla {
   animation: light-on 2s ease-in-out 1s infinite alternate forwards;
}

@keyframes light-on {
        0% { opacity: 0;        }
        100% {     opacity: 0.75;  }
}
```