@Keyframes

@keyframes es una regla de CSS que permite crear animaciones personalizadas definiendo cómo deben cambiar los estilos de un elemento en distintos momentos de la animación.

En lugar de simplemente hacer una transición entre dos estados (como en transition), @keyframes permite especificar múltiples etapas intermedias, con total control sobre el ritmo, la dirección y el comportamiento de la animación.

Cuándo usar @keyframes

Usamos @keyframes cuando necesitamos:

- Animaciones complejas con varios pasos.
- Animaciones independientes (no necesitan :hover).
- Repetición o control total del ciclo de animación.
- Coordinación de movimientos, transformaciones o estilos en diferentes momentos.

1. Animación Básica

Efecto: Una animación simple que cambia la opacidad de un elemento.

```
.fadeIn {
  opacity: 0;
  animation-name: fadeIn;
  animation-duration: 2s;
  animation-delay: 0.5s;
}

@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}
```

Propiedades utilizadas:

- @keyframes: Define los estados de la animación (e.g., from { opacity: 0; } to { opacity: 1; }).
- animation-name: Establece el nombre de la animación (fadeIn).
- animation-duration: Define la duración de la animación (2s).
- animation-delay: Establece un retraso antes de que comience la animación (0.5s).

2. Fill Mode

Efecto: Controla el comportamiento del elemento después de que finaliza la animación, utilizando diferentes valores para animation-fill-mode.

- Forwards: Mantiene el estado final de la animación después de que esta termine.
- Backwards: Mantiene el estado inicial de la animación antes de que comience.
- Both: Combina los efectos de forwards y backwards, manteniendo el estado inicial durante el retraso y el estado final después de la animación.
- None: No mantiene ningún estilo fuera de la animación, ni antes ni después.

```
.fillForwards {
  animation-name: fadeIn;
  animation-duration: 2s;
  animation-fill-mode: forwards;
}

@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}
```

Propiedades utilizadas:
- animation-fill-mode: Define el comportamiento después de la animación (forwards, backwards, both, none).

3. Dirección de la Animación
Efecto: Controla la dirección en la que se ejecuta la animación, como si es normal, en reversa o alternando entre ambas.

- Normal: La animación se ejecuta de principio a fin.
- Reverse: La animación se ejecuta en orden inverso.
- Alternate: La animación alterna entre las direcciones de principio a fin y de fin a principio.

```
.animateReverse {
  animation-name: moveRight;
  animation-duration: 2s;
  animation-direction: reverse;
}

@keyframes moveRight {
  from { transform: translateX(0); }
  to { transform: translateX(100px); }
}
```

Propiedades utilizadas:
- animation-direction: Define la dirección de la animación (normal, reverse, alternate).

4. Repetición de la Animación
Efecto: Controla cuántas veces se repite la animación.

- Repetir un número específico de veces (ej. 3 veces).
- Repetir infinitamente.

```
.repeatInfinite {
  animation-name: bounce;
  animation-duration: 1s;
  animation-iteration-count: infinite;
```

```
}

@keyframes bounce {
  from { transform: translateY(0); }
  to { transform: translateY(-50px); }
}
```

Propiedades utilizadas:

- animation-iteration-count: Establece el número de repeticiones (3, infinite).

5. Transformaciones

Efecto: Aplica transformaciones como mover, rotar, escalar o sesgar un elemento.

- translateX(): Mueve el elemento horizontalmente.
- rotate(): Rota el elemento en grados.
- scale(): Cambia el tamaño del elemento.
- skew(): Deforma el elemento en ángulos específicos.

```
.transformScale {
  animation-name: scaleUp;
  animation-duration: 2s;
}

@keyframes scaleUp {
  from { transform: scale(1); }
  to { transform: scale(1.5); }
}
```

Propiedades utilizadas:

- transform: Aplica las transformaciones a los elementos (translateX(), rotate(), scale(), skew()).
- transition: Suaviza la transición de transformaciones y cambios de estilo (transform 0.4s ease-in-out).

6. Perspectiva 3D

Efecto: Crea un efecto tridimensional rotando un elemento en el espacio 3D.

```
.perspective3D {
  perspective: 500px;
}

.box {
  transform-style: preserve-3d;
  animation-name: rotate3D;
  animation-duration: 3s;
}

@keyframes rotate3D {
```

```
  from { transform: rotateY(0); }
  to { transform: rotateY(360deg); }
}
```

Propiedades utilizadas:

- perspective: Establece la distancia de la cámara para la vista 3D.
- transform-style: Permite que los hijos del contenedor con perspectiva se comporten en 3D.
- transform: Rota el elemento en el eje Y (rotateY()).

8. Otros Ejemplos de Keyframes

Efectos adicionales usando @keyframes:

- Heartbeat (Latido): Crea un efecto de "latido" usando la propiedad scale() para ampliar y reducir el tamaño de un elemento.
- Rainbow (Arco iris): Cambia el color de fondo del elemento a través de una serie de colores del arco iris.
- Spinner (Giro tipo loading): Rota el elemento continuamente para simular una animación de carga.
- Slide Up (Aparecer desde abajo): El elemento se desliza desde abajo hacia su posición final.
- Bubble (Burbuja): El elemento "sube" mientras se desvanece, creando una animación tipo burbuja.
- Rocket (Despegue): El elemento sube verticalmente como si fuera un cohete despegando.
- Pulse (Pulso): El elemento crece y se reduce como un pulso, con un retraso antes de que ocurra.

Propiedades utilizadas:

- @keyframes: Define la animación con los estados clave (e.g., from { opacity: 0; } to { opacity: 1; }).
- animation-name: Establece el nombre de la animación (e.g., heartbeat).
- animation-duration: Define el tiempo que toma la animación (2s).
- animation-timing-function: Controla la velocidad de la animación (e.g., ease, linear).

Transition

Las transiciones permiten cambiar suavemente los valores de una o más propiedades CSS durante un período de tiempo determinado.

Efecto: Aplica una transición suave de un estado a otro, como un cambio de color o un efecto de escala.

```
.transitionColor {
  transition: background-color 0.5s ease;
}
```

```
.transitionColor:hover {
  background-color: 3498db;
}
```

Propiedades utilizadas:

- transition: Aplica una transición suave a los cambios de estilo (por ejemplo, color, tamaño, etc.) (all 0.5s ease-in-out).

Propiedades de transition

- transition-property: Define qué propiedad CSS será animada.
- transition-duration: Define cuánto tiempo toma completar la transición.
- transition-delay: Establece cuánto esperar antes de que inicie la transición.
- transition-timing-function: Controla la velocidad del cambio en el tiempo, usando funciones como ease, linear, ease-in, ease-out, ease-in-out o cubic-bezier.
- transition: Shorthand que permite escribir las anteriores en una sola línea.

Ejemplo con shorthand:

```
transition: background-color 0.5s ease-in 0.2s;
```

Traducción: cambia background-color en 0.5s, con aceleración al inicio (ease-in), después de 0.2s de espera.

Funciones de transition-timing-function

- ease: aceleración suave al inicio y al final.
- linear: velocidad constante.
- ease-in: comienza lento y acelera.
- ease-out: comienza rápido y desacelera.
- ease-in-out: combinación de las dos anteriores.
- cubic-bezier(x1, y1, x2, y2): curva personalizada.

Cubic Bezier

Permite diseñar curvas de velocidad más complejas. Ejemplo:

```
transition-timing-function: cubic-bezier(0.68, -0.55, 0.27, 1.55);
```

Esto genera un efecto de "rebote" visual al final.

Transition vs keyframes

Diferencias clave entre transition y @keyframes

Característica	`transition`	`@keyframes`
Simplicidad	Muy simple, para cambios directos	Más complejos y controlables
Paso a paso	No permite múltiples etapas	Permite definir varios pasos (`0%` , `50%` , `100%` , etc.)
Activación	Generalmente con `:hover` , `:focus` , etc.	Se puede ejecutar al cargar o bajo clases específicas
Control avanzado	Limitado	Total control de cada punto de la animación

transition:
Objetivo: Animar un solo cambio de propiedad cuando se produce un evento (ejemplo: :hover).

- Simplicidad: Ideal para animaciones simples.
- Acción basada en eventos: Depende de la interacción del usuario.
- Propiedades clave: transition-property, transition-duration, transition-timing-function, transition-delay.

@keyframes:
Objetivo: Crear animaciones complejas con múltiples cambios de propiedades a lo largo del tiempo.

- Complejidad: Permite más control y flexibilidad, con pasos intermedios y animaciones continuas.
- Sin interacción necesaria: Se ejecuta automáticamente o según otros disparadores.
- Propiedades clave: @keyframes, animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, animation-direction.

¿Cuándo usar cada uno?
Usar transition: Cuando quieras animar un solo cambio entre dos estados, como al hacer hover o cuando se recibe un enfoque.

Usar @keyframes: Cuando necesites animaciones más complejas que involucran múltiples pasos, o cuando no dependa de una interacción del usuario.

Recursos generadores de efecto:
Webcode
Animista

Actividad Creando Animaciones Básicas con CSS

Les propongo una actividad introductoria que cubre los efectos básicos. Esta actividad les permitirá experimentar con las propiedades clave y entender cómo se aplican las animaciones.

Objetivo: Aprender a aplicar animaciones básicas a elementos HTML utilizando CSS. Trabajarán con animaciones simples como la opacidad, el movimiento, y las transformaciones, y verán cómo se configuran las propiedades relacionadas con @keyframes, animation, transition, entre otras.

Instrucciones:

1. Crear una estructura básica de HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Animaciones Básicas con CSS</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1></h1>
  <div></div>
</body>
</html>
```

2. Estilo básico de la caja:

Luego, deben crear una clase .box en su archivo styles.css que defina las características visuales de los elementos (tamaño, color, borde, etc.):

```
.box {
  width: …;
  height: …;
  margin: …;
  background-color: …;
  color: …;
    }
```

3. Aplicar animaciones básicas:

A continuación, deben agregar las animaciones a los elementos. Cada uno de los elementos debe tener una animación distinta.

- *Elemento 1: Animación de Fade (opacidad)*
  - **Instrucción**: El primer elemento debe aparecer con una animación de desvanecimiento (fadeIn).

```
.box{
    animation: nombreAnimacion tiempo timing-function;
}

  @keyframes nombreAnimacion {
    from { …; }
    to { …; }
  }
```

- *Elemento 2: Animación de Movimiento*
  - **Instrucción**: El segundo elemento debe moverse de izquierda a derecha.

```
  .box {
    animation: nombreAnimacion tiempo timing-function;
  }

  @keyframes nombreAnimacion {
    from { transform: …; }
    to { transform: …; }
  }
```

- *Elemento 3: Animación de Escala*
  - **Instrucción**: El tercer elemento debe agrandarse y reducirse en tamaño (efecto de pulso).

```
  .box {
    animation: nombreAnimacion tiempo timing-function;
  }

  @keyframes nombreAnimacion {
    0% { transform: …; }
    50% { transform: …; }
    100% { transform: …; }
  }
```

4. Exploración y Modificación:

Finalmente, invita a los estudiantes a que experimenten modificando los parámetros de las animaciones:

- ¿Qué sucede si cambian la duración (**animation-duration**)?
- ¿Qué ocurre si utilizan una **función de tiempo** diferente como linear o ease-in?

- ¿Cómo cambia la animación si se ajusta el valor de **animation-iteration-count** para repetirla un número específico de veces?

5. Bonus:

Los estudiantes deben hacer el codigo por su cuenta, añadir más elementos a la página y aplicarles animaciones adicionales, como transformaciones 3D, cambios de color o rotaciones.

Sugerencias:

- Observar cómo las animaciones pueden hacer que las interfaces sean más interactivas y visualmente atractivas.

- Es importante usar animaciones con moderación para no distraer demasiado al usuario.

- Recuerden que pueden usar el inspector de su navegador para ver cómo funcionan las animaciones en tiempo real.