Guía básica para utilizar @keyframes en CSS

`@keyframes` es una regla clave en CSS que nos permite definir animaciones detalladas para elementos HTML. Con esta herramienta, puedes especificar los estilos que un elemento deberá tener en diferentes puntos de su ciclo de animación, creando efectos atractivos y dinámicos.

¿Qué es `@keyframes`?

`@keyframes` se utiliza para definir animaciones que cambian las propiedades CSS de un elemento a lo largo del tiempo. Funciona mediante la definición de "etapas" (o keyframes) que describen cómo debería cambiar el elemento en diferentes porcentajes de la duración total de la animación.

Un ciclo de animación siempre tiene:

- **Un estado inicial:** Se define con `from` o `0%`.
- **Un estado final:** Se define con `to` o `100%`.

Estructura básica

Ejemplo básico

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

En este ejemplo, una animación llamada `ejemplo` cambia la opacidad de un elemento de 0 (invisible) a 1 (completamente visible).

Uso en un elemento

Aplica la animación al elemento usando la propiedad `animation`:

```
.elemento {
    animation: ejemplo 2s ease-in-out;
}
```

Esto indica que:

- La animación `ejemplo` durará **2 segundos**.
- Utilizará una transición **suave al inicio y al final** (`ease-in-out`).

Definiendo múltiples etapas

Puedes usar porcentajes para describir cómo evoluciona la animación en diferentes momentos.

```
@keyframes movimiento {
    0% {
        transform: translateX(0);
    }
    50% {
        transform: translateX(100px);
    }
    100% {
        transform: translateX(0);
    }
}
```

En este caso:

- Al **0%**, el elemento está en su posición original.
- Al **50%**, se mueve 100px a la derecha.
- Al **100%**, vuelve a su posición inicial.

Aplicación:

```
.elemento {
    animation: movimiento 1s ease-in-out infinite;
}
```

Esto crea un movimiento continuo (con `infinite`) de ida y vuelta.

Ejemplo aplicado: Indicador de carga

Un caso común para un desarrollador frontend es crear un "indicador de carga" para mostrar que algo está procesándose.

HTML

```
<div class="loader"></div>
```

CSS

```
.loader {
    width: 50px;
    height: 50px;
    border: 5px solid #ccc;
    border-top: 5px solid #3498db;
    border-radius: 50%;
    animation: girar 1s linear infinite;
}

@keyframes girar {
    from {
        transform: rotate(0deg);
    }
    to {
        transform: rotate(360deg);
    }
}
```

Resultado: Un círculo que gira continuamente, ideal como indicador de carga.

Buenas prácticas

1. **Usa nombres descriptivos:** Elige nombres que expliquen claramente lo que hace la animación.
   - ✔ `@keyframes fadeIn`
   - X `@keyframes animacion1`
2. **Evita abusar de las animaciones:** Usar demasiadas animaciones puede distraer al usuario y afectar el rendimiento.

3. **Combina con transiciones:** A veces, una simple transición es suficiente para lograr un buen efecto.
4. **Optimiza para rendimiento:**
   - Usa propiedades que puedan acelerarse por GPU como `transform` o `opacity`.
   - Evita animar propiedades como `width`, `height` o `box-shadow`, ya que requieren recalcular el diseño.
5. **Hazlo accesible:**
   - Usa animaciones suaves para no incomodar a usuarios sensibles.
   - Respeta las preferencias del sistema con la media query `@media (prefers-reduced-motion)`.

```
@media (prefers-reduced-motion: reduce) {
    .loader {
        animation: none;
    }
}
```

Malas prácticas

1. **Evitar usar `!important`:**
   - X `animation: girar 1s linear infinite !important;`
   - Esto dificulta la depuración y mantenimiento del código.
2. **No usar animaciones sin propósito:**
   - Las animaciones deben aportar valor, no ser una decoración innecesaria.
3. **No definir duraciones extremas:**
   - X `animation: fadeIn 0.01s;` (demasiado rápida).
   - X `animation: fadeIn 10s;` (innecesariamente lenta).

Resumen de propiedades relacionadas

Propiedad	Descripción	Ejemplo
`animation-name`	Nombre de la animación definida con `@keyframes`.	`animation-name: fadeIn;`
`animation-duration`	Duración de la animación.	`animation-duration: 2s;`
`animation-timing-function`	Velocidad de la animación en cada etapa.	`animation-timing-function: ease;`
`animation-delay`	Retraso antes de que comience la animación.	`animation-delay: 1s;`
`animation-iteration-count`	Veces que se repite la animación.	`animation-iteration-count: infinite;`
`animation-direction`	Dirección de la animación (normal, inversa, alterna).	`animation-direction: alternate;`
`animation-fill-mode`	Define cómo se comporta el elemento antes/después de la animación.	`animation-fill-mode: forwards;`

Ejercicio de práctica

Crea una animación llamada `bounce` que haga que un elemento suba y baje constantemente.

1. Define un `@keyframes` que:
   - Al **0%** y **100%**: el elemento esté en su posición original.
   - Al **50%**: el elemento suba 20px.
2. Aplica la animación a un cuadro con clase `caja`.