

# Transiciones y Animaciones en CSS

Julian F. Latorre

October 7, 2024

## Objetivos de la clase

- ▶ Comprender la diferencia entre transiciones y animaciones
- ▶ Implementar transiciones básicas y avanzadas
- ▶ Crear animaciones personalizadas con keyframes
- ▶ Aplicar técnicas para mejorar la UX en sitios web

# Transiciones en CSS

- ▶ Permiten cambios suaves en propiedades CSS
- ▶ Propiedades principales:
  - ▶ `transition-property`
  - ▶ `transition-duration`
  - ▶ `transition-timing-function`
  - ▶ `transition-delay`

```
.button {  
    background-color: blue;  
    transition: background-color 0.3s ease-in-out;  
}  
  
.button:hover {  
    background-color: red;  
}
```

## Funciones de temporización

- ▶ `ease`: Comienza lento, acelera y luego desacelera (por defecto)
- ▶ `linear`: Velocidad constante
- ▶ `ease-in`: Comienza lento y acelera
- ▶ `ease-out`: Comienza rápido y desacelera
- ▶ `ease-in-out`: Comienza y termina lento, con aceleración en el medio
- ▶ `cubic-bezier()`: Función personalizada

## Transiciones múltiples

```
.card {  
  width: 200px;  
  height: 300px;  
  background-color: #f0f0f0;  
  transition:  
    width 0.5s ease,  
    height 0.5s ease,  
    background-color 0.5s linear;  
}  
  
.card:hover {  
  width: 220px;  
  height: 320px;  
  background-color: #e0e0e0;  
}
```

# Animaciones en CSS

- ▶ Permiten secuencias más complejas de cambios
- ▶ Utilizan `@keyframes` para definir estados intermedios
- ▶ Propiedades principales:
  - ▶ `animation-name`
  - ▶ `animation-duration`
  - ▶ `animation-timing-function`
  - ▶ `animation-delay`
  - ▶ `animation-iteration-count`
  - ▶ `animation-direction`
  - ▶ `animation-fill-mode`
  - ▶ `animation-play-state`

## Ejemplo de animación

```
@keyframes bounce {  
  0%, 100% { transform: translateY(0); }  
  50% { transform: translateY(-20px); }  
}  
  
.bouncing-ball {  
  width: 50px;  
  height: 50px;  
  background-color: red;  
  border-radius: 50%;  
  animation: bounce 1s ease-in-out infinite;  
}
```

# Animaciones múltiples

```
@keyframes changeColor {  
  0% { background-color: red; }  
  50% { background-color: blue; }  
  100% { background-color: green; }  
}  
  
.fancy-ball {  
  animation:  
    bounce 1s ease-in-out infinite,  
    changeColor 3s linear infinite;  
}
```




## Mejores prácticas y rendimiento

- ▶ Evite animar propiedades que desencadenen reflows
- ▶ Prefiera animar `transform` y `opacity`
- ▶ Use `will-change` con moderación
- ▶ Considere reducir la complejidad en dispositivos móviles
- ▶ 🚀 Optimice para un mejor rendimiento

## Accesibilidad en animaciones

- ▶ Respete la preferencia de movimiento reducido:

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

- ▶ Evite animaciones que puedan causar mareos
- ▶ Asegure que el contenido sea accesible sin animaciones
- ▶  Priorice la accesibilidad

# Ejercicios prácticos

## Conclusión

- ▶ Transiciones para cambios simples
- ▶ Animaciones para secuencias complejas
- ▶ Considerar rendimiento y accesibilidad
- ▶ Practicar y experimentar



¡Sigán aprendiendo y creando!