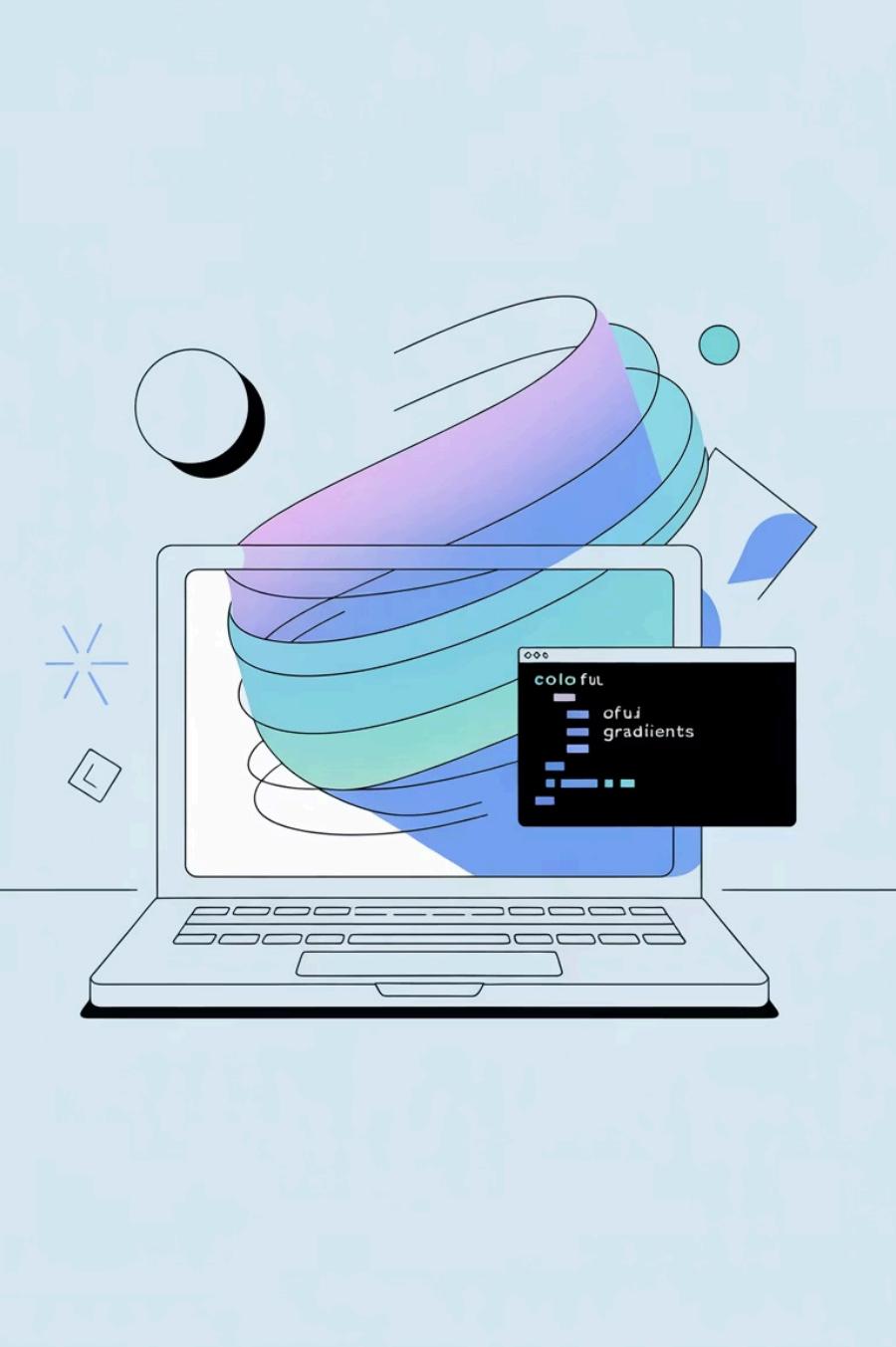


Introducción al Desarrollo web

Curso: Desarrollo Frontend con React



Animaciones y Transiciones en CSS

¿Qué aprenderemos hoy?



Fundamentos

1

Introducción a animaciones y pseudoclases para interacciones



Propiedades clave

2

Transition, transform y @keyframes



Casos prácticos

3

Ejemplos reales y buenas prácticas de implementación



Optimización

4

Cómo crear animaciones sin afectar el rendimiento

¿Por qué animar elementos web?

Mejora la experiencia de usuario

Las animaciones proporcionan feedback visual y guían la atención del usuario

Comunica información

Indican estados, jerarquías y relaciones entre elementos

Añade personalidad

Diferencia tu sitio y refuerza la identidad de marca

Create a website
ud ur devenes

Sign Up

Pricing

Docs

Home

Features

Sign Up

Honicative website
ud ur devenes

Sigjo ut

Petfor Up

Colets of Sulca

Toitotarc's Terzeof Service

Eetons

Privacy Policy

Paulo Antropoe idt Riamatocé Eleteruc Coo D3B Polon Aet Dajc Prilst

Dos formas de añadir movimiento

Transiciones

Cambios suaves entre dos estados

- Simples de implementar
- Ocurren en respuesta a un evento
- Control limitado (inicio y fin)

```
.boton {  
    transition: transform 0.3s ease;  
}  
.boton:hover {  
    transform: scale(1.1);  
}
```

Animaciones

Secuencias complejas de cambios

- Mayor flexibilidad creativa
- Control total del timing
- Pueden ejecutarse automáticamente

```
@keyframes pulsar {  
    0% { transform: scale(1); }  
    50% { transform: scale(1.2); }  
    100% { transform: scale(1); }  
}
```

Pseudoclases para interacciones

Las pseudoclases nos permiten aplicar estilos cuando el elemento está en un estado específico

:hover

Cuando el cursor está sobre el elemento

```
a:hover {  
    color: #2589C9;  
}
```

:active

Cuando se está pulsando el elemento

```
button:active {  
    background: #063E5F;  
}
```

:focus

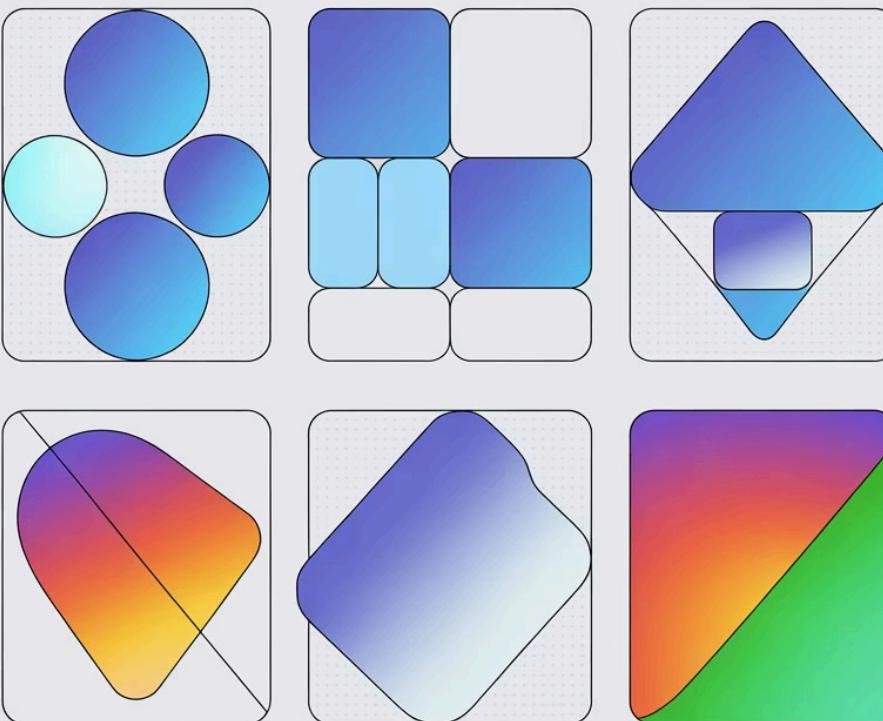
Cuando el elemento tiene el foco

```
input:focus {  
    border-color: #75bae6;  
}
```

Estas pseudoclases son perfectas para combinar con transiciones y crear interacciones naturales

Interacting UI Element Estates

20/4 3:19pm



Propiedad transition

La forma más sencilla de añadir movimiento a tu interfaz

```
.elemento {  
  transition: [propiedad] [duración] [timing-function] [delay];  
}
```



Duración

Tiempo que tarda en completarse (0.3s, 200ms)



Timing function

Curva de aceleración (ease, linear, ease-in-out)



Delay

Tiempo antes de iniciar (opcional)

Propiedades animables

No todas las propiedades CSS pueden animarse de forma eficiente

Recomendadas

- transform
- opacity
- color, background-color
- filter

Rendimiento óptimo - Utilizan la GPU

Utiliza con precaución

- width, height
- padding, margin
- top, left, bottom, right
- font-size

Mayor coste - Provocan reflow (recálculo del layout)

La regla de oro: Prioriza transformaciones y opacidad para animaciones fluidas

Ejemplo: Transición simple

```
.boton {  
background-color: #75bae6;  
padding: 10px 20px;  
border-radius: 4px;  
transition:  
background-color 0.3s ease,  
transform 0.2s ease;  
}  
  
.boton:hover {  
background-color: #2589C9;  
transform: translateY(-3px);  
}  
  
.boton:active {  
transform: translateY(0);  
}
```

Este código crea:

- Un botón con fondo azul claro
- Al hacer hover, se oscurece y se eleva ligeramente
- Al hacer clic, vuelve a su posición original
- Todos los cambios ocurren de manera suave y fluida

Explore now



Discover the future
of productivity

La magia de transform

Modifica visualmente un elemento sin afectar al flujo del documento



rotate

Gira el elemento un número específico de grados

```
transform: rotate(45deg);
```



scale

Cambia el tamaño del elemento

```
transform: scale(1.5);
```



translate

Mueve el elemento horizontal o verticalmente

```
transform: translateY(-10px);
```



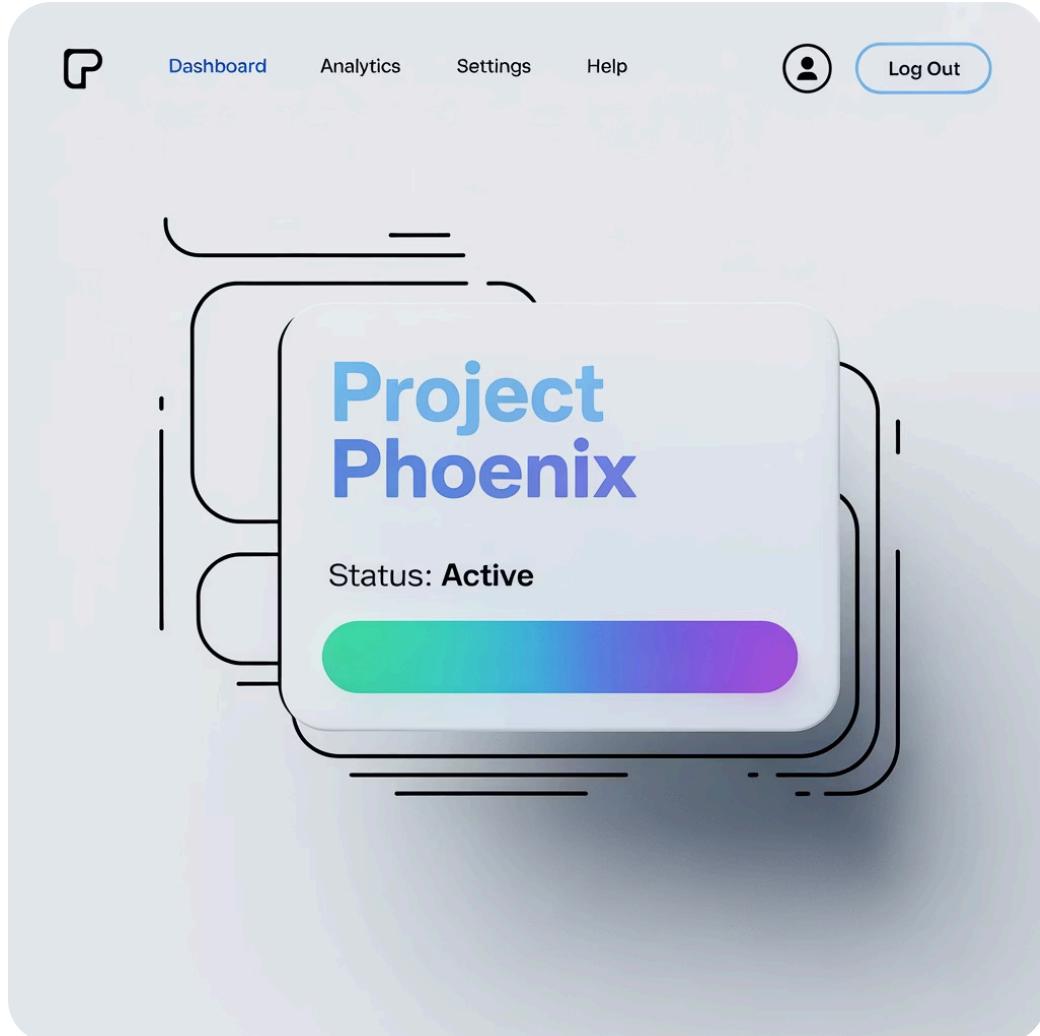
skew

Inclina el elemento en uno o ambos ejes

```
transform: skewX(10deg);
```

Las transformaciones son combinables: `transform: rotate(45deg) scale(1.2);`

Combinando transformaciones y transiciones



```
.tarjeta {  
    transition:  
        transform 0.3s ease-out,  
        box-shadow 0.3s ease;  
}
```

```
.tarjeta:hover {  
    transform:  
        translateY(-5px)  
        rotate(2deg);  
    box-shadow:  
        0 10px 20px rgba(0,0,0,0.2);  
}
```

Al combinar varias transformaciones con transiciones suaves, conseguimos efectos sofisticados con pocas líneas de código.

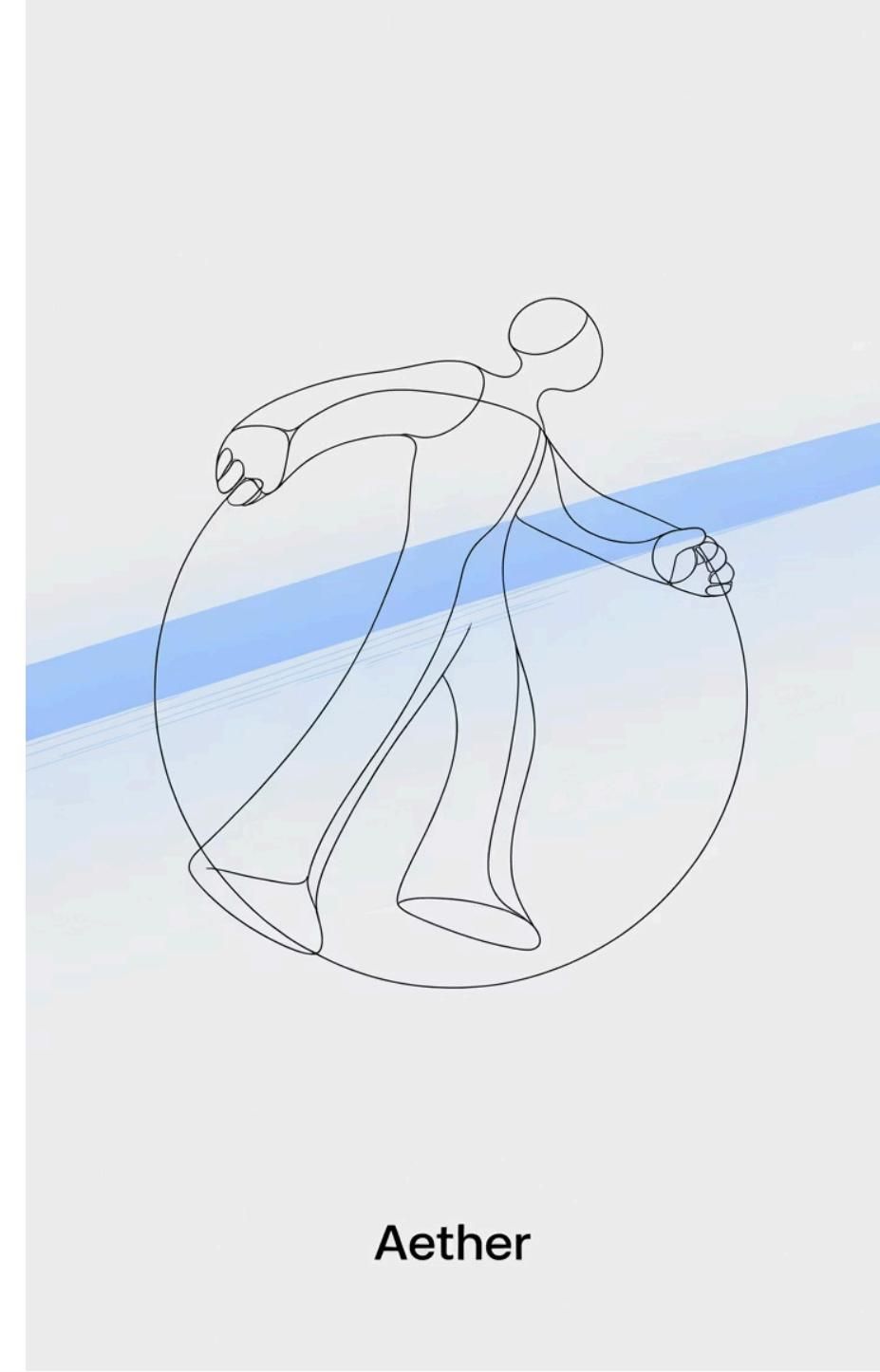
Animaciones con @keyframes

Para secuencias más complejas donde las transiciones no son suficientes

```
@keyframes nombreAnimacion {  
    0% {  
        /* Estado inicial */  
    }  
    50% {  
        /* Estado intermedio */  
    }  
    100% {  
        /* Estado final */  
    }  
}
```

Después, aplicas la animación a un elemento:

```
.elemento {  
    animation: nombreAnimacion 2s ease infinite;  
}
```



Aether

Propiedades de animation

animation-name

Nombre del @keyframes a utilizar

animation-duration

Duración del ciclo completo

animation-timing-function

Curva de aceleración (ease, linear, cubic-bezier)

animation-delay

Retraso antes de iniciar

animation-iteration-count

Número de repeticiones (número o infinite)

animation-direction

Dirección (normal, reverse, alternate)

animation-fill-mode

Estado antes/después (forwards, backwards, both)

animation-play-state

Estado de reproducción (running, paused)

Forma abreviada: animation: nombre 2s ease infinite alternate;

Ejemplo: Botón con efecto de pulso

```
@keyframes pulso {  
    0% {  
        transform: scale(1);  
        box-shadow: 0 0 0 0 rgba(117, 186, 230, 0.7);  
    }  
  
    70% {  
        transform: scale(1.05);  
        box-shadow: 0 0 0 10px rgba(117, 186, 230, 0);  
    }  
  
    100% {  
        transform: scale(1);  
        box-shadow: 0 0 0 0 rgba(117, 186, 230, 0);  
    }  
  
}  
  
.boton-accion {  
    background-color: #2589C9;  
    color: white;  
    border: none;  
    border-radius: 4px;  
    padding: 12px 24px;  
    animation: pulso 2s infinite;  
}
```

Explore Now

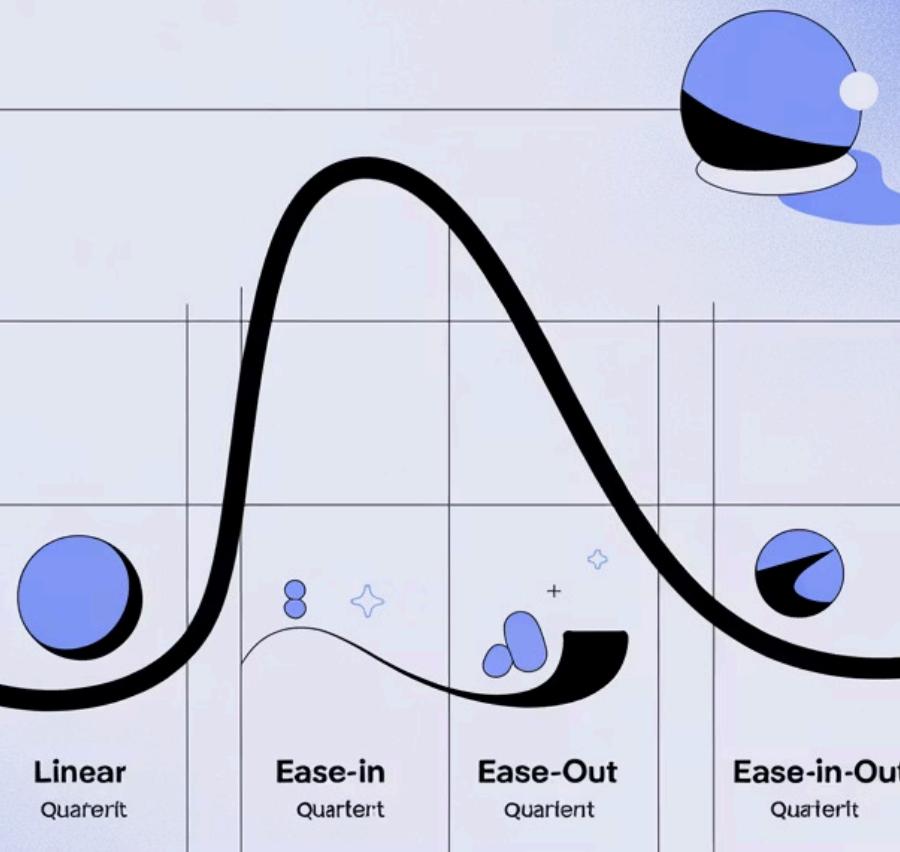


Discover the future of simplicity

TERMS PRIVACY CONTACT

Esta animación llama la atención sobre el botón principal de acción, indicando al usuario dónde debe hacer clic.

Easing Curves



Timing Functions

La forma en que se acelera o desacelera una animación puede cambiar completamente su sensación

linear

Velocidad constante

Útil para rotaciones continuas

ease-in

Comienza lento, termina rápido

Para elementos que salen de pantalla

ease-out

Comienza rápido, termina lento

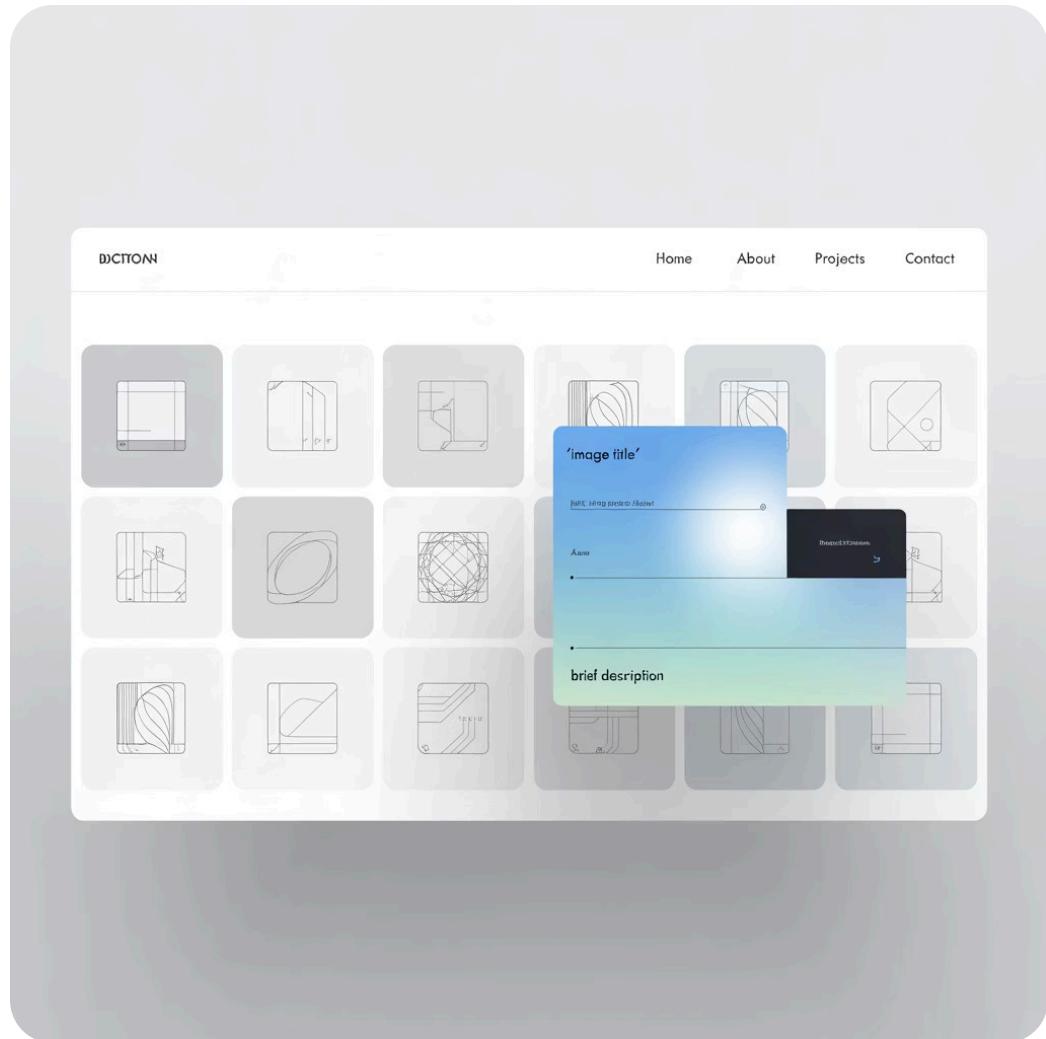
Para elementos que entran en pantalla

ease-in-out

Lento al inicio y al final

Para movimientos naturales

Caso práctico: Galería de imágenes animada



```
.galeria-item {  
    position: relative;  
    overflow: hidden;  
}
```

```
.galeria-img {  
    transition: transform 0.5s ease;  
}
```

```
.galeria-info {  
    position: absolute;  
    bottom: -100%;  
    background: rgba(0,0,0,0.7);  
    color: white;  
    width: 100%;  
    padding: 20px;  
    transition: bottom 0.4s ease-out;  
}
```

```
.galeria-item:hover .galeria-img {  
    transform: scale(1.1);  
}
```

```
.galeria-item:hover .galeria-info {  
    bottom: 0;  
}
```

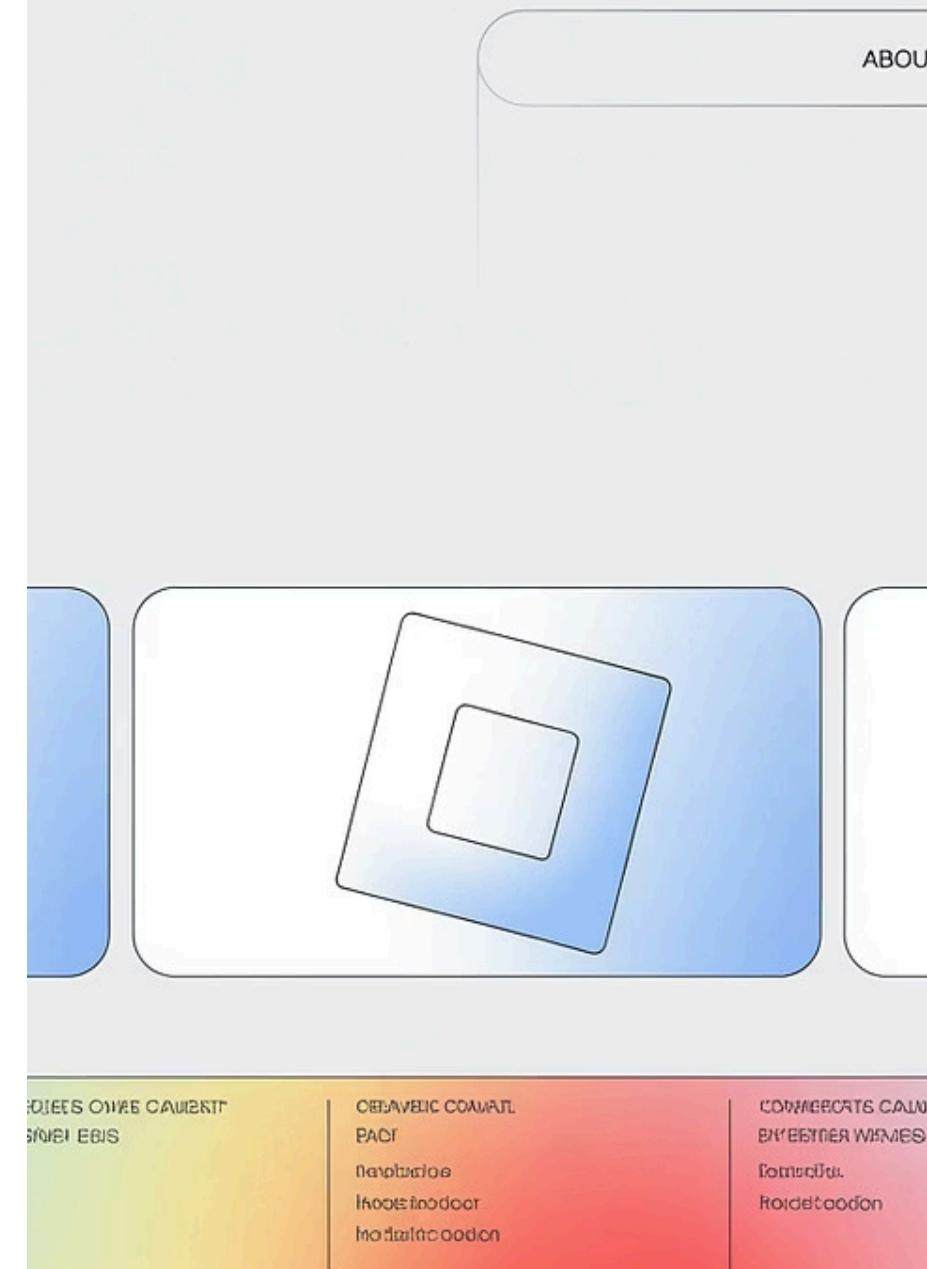
Animaciones escalonadas

Animar varios elementos con pequeños retrasos entre ellos crea un efecto más natural y sofisticado

```
.menu-item:nth-child(1) { animation-delay: 0.1s; }  
.menu-item:nth-child(2) { animation-delay: 0.2s; }  
.menu-item:nth-child(3) { animation-delay: 0.3s; }  
.menu-item:nth-child(4) { animation-delay: 0.4s; }
```

Esta técnica es especialmente efectiva para:

- Menús desplegables
- Listas de elementos que se cargan
- Entrada secuencial de contenido



Buenas prácticas: Rendimiento

Las animaciones mal implementadas pueden afectar negativamente al rendimiento de la página

Propiedades eficientes

Prioriza transform y opacity sobre propiedades que causan reflow (width, top, etc.)

will-change

Advierte al navegador sobre elementos que se animarán para optimizar el rendimiento

```
will-change: transform, opacity;
```

Animaciones breves

Mantén las animaciones por debajo de 300-500ms para interfaces reactivas

Considera dispositivos móviles

Reduce complejidad en móviles o utiliza media queries para adaptar animaciones

Media queries para animaciones

Adapta tus animaciones según las preferencias del usuario y el dispositivo

Preferencia de movimiento reducido

```
@media (prefers-reduced-motion: reduce) {  
  * {  
    animation-duration: 0.01ms !important;  
    transition-duration: 0.01ms !important;  
  }  
}
```

Respecta a usuarios con trastornos vestibulares o sensibilidad al movimiento

Dispositivos de bajo rendimiento

```
@media (max-width: 768px) {  
  .elemento {  
    /* Animación simplificada */  
    animation: nombreSimple 1s;  
  }  
}
```

```
@media (min-width: 769px) {  
  .elemento {  
    /* Animación compleja */  
    animation: nombreComplejo 2s;  
  }  
}
```

¿Cuándo usar animaciones?



Feedback de interacción

Confirmación visual de acciones del usuario



Transiciones entre estados

Cambios de página o contenido



Dirigir la atención

Destacar elementos importantes



Contar una historia

Revelar información de forma secuencial

- ⓘ Las animaciones deben **mejorar** la experiencia, no distraer de ella. Cada animación debe tener un propósito claro.

¿Cuándo evitar animaciones?

Contenido principal

Nunca animes el texto principal que el usuario necesita leer

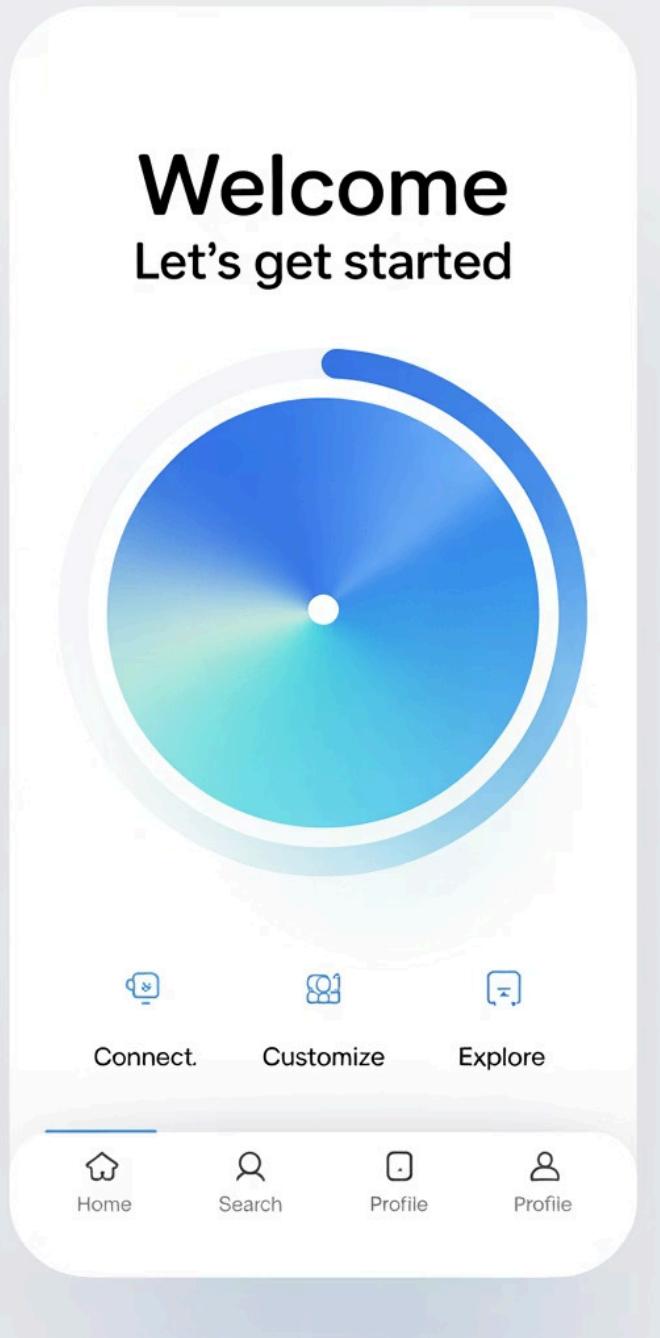
Animaciones perpetuas

Movimientos constantes en pantalla que distraen la atención

Interfaces críticas

Formularios importantes donde la eficiencia es prioritaria

"Las buenas animaciones son invisibles. No las notas por sí mismas, sino por cómo mejoran la experiencia del usuario."



Animando al hacer scroll

Una técnica popular para crear páginas más dinámicas

Opciones para implementarlo:

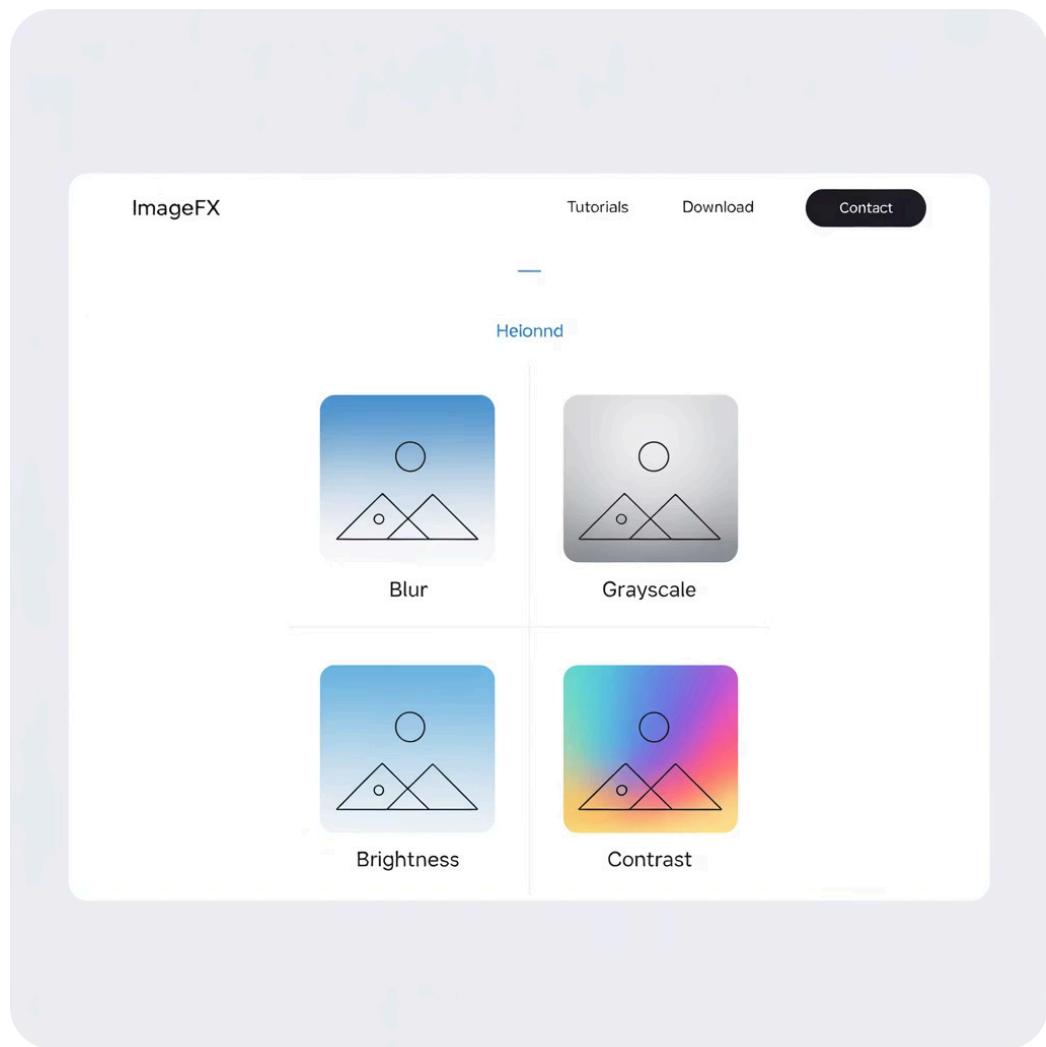
- Intersection Observer API (recomendado)
- Bibliotecas como AOS (Animate On Scroll)
- CSS nativo con scroll-behavior

```
.elemento {  
    opacity: 0;  
    transform: translateY(30px);  
    transition: opacity 0.6s ease,  
    transform 0.6s ease;  
}
```

```
.elemento.visible {  
    opacity: 1;  
    transform: translateY(0);  
}
```

La propiedad filter

Efectos visuales que combinan bien con animaciones



```
.imagen {  
    transition: filter 0.5s ease;  
}  
  
.imagen:hover {  
    filter: brightness(1.2) contrast(1.1);  
}  
  
/* Otros valores de filter */  
filter: blur(5px);  
filter: grayscale(100%);  
filter: sepia(70%);  
filter: hue-rotate(180deg);  
filter: invert(100%);  
filter: drop-shadow(5px 5px 10px black);
```

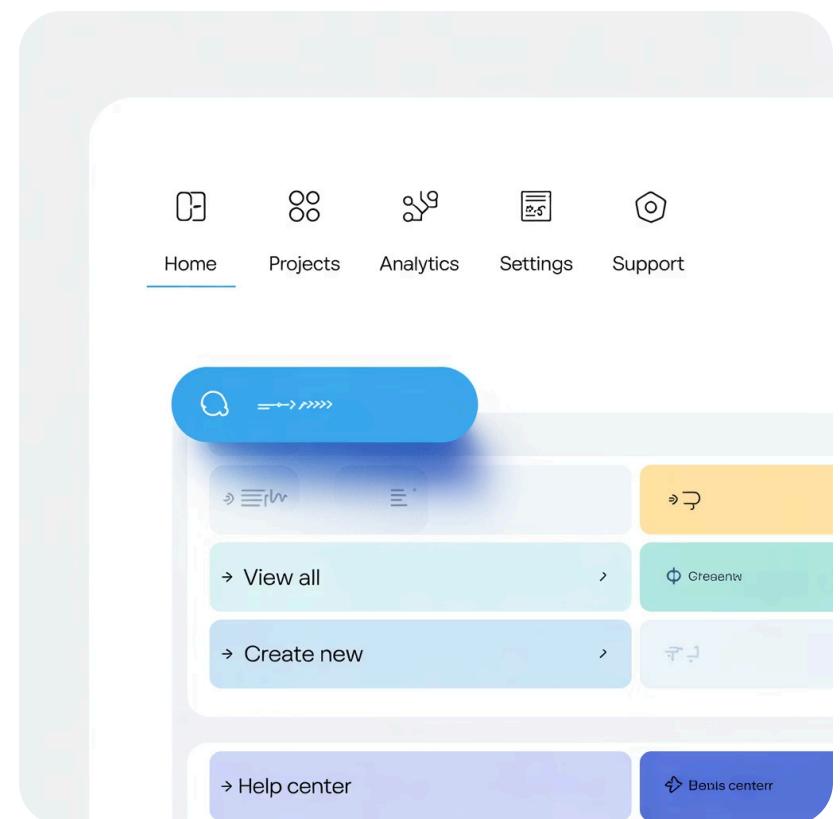
Ejemplo práctico: Menú de navegación

```
.nav-item {  
  position: relative;  
}  
  
.nav-item::after {  
  content: " ";  
  position: absolute;  
  width: 0;  
  height: 2px;  
  bottom: 0;  
  left: 0;  
  background-color: #75bae6;  
  transition: width 0.3s ease-out;  
}  
  
}
```

```
.nav-item:hover::after,  
.nav-item.active::after {  
  width: 100%;  
}
```

```
.submenu {  
  max-height: 0;  
  overflow: hidden;  
  opacity: 0;  
  transition:  
    max-height 0.5s ease,  
    opacity 0.3s ease;  
}  
  
}
```

```
.nav-item:hover .submenu {  
  max-height: 300px;  
  opacity: 1;  
}
```



Este código crea:

- Una línea que se extiende al hacer hover
- Un submenú que se despliega suavemente

Animando con variables CSS

Las propiedades personalizadas (variables CSS) permiten animaciones más dinámicas y reutilizables

```
:root {  
  --color-primario: #75bae6;  
  --escala: 1;  
  --rotacion: 0deg;  
}
```

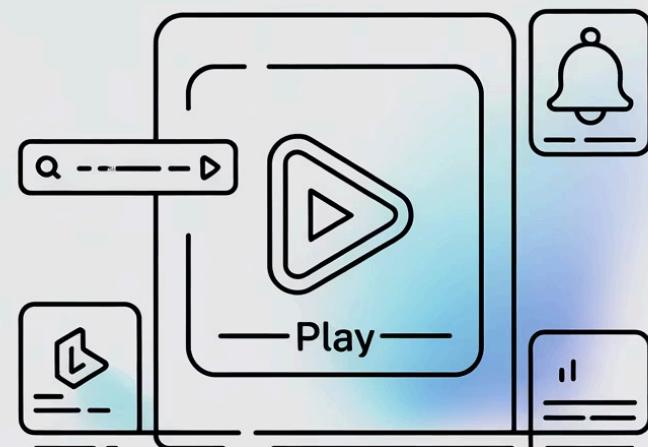
```
.boton {  
  background-color: var(--color-primario);  
  transform:  
    scale(var(--escala))  
    rotate(var(--rotacion));  
  transition: transform 0.3s ease;  
}
```

```
.boton:hover {  
  --escala: 1.1;  
  --rotacion: 5deg;  
}
```

Ventajas:

- Centraliza valores en un solo lugar
- Facilita temas y cambios globales
- Permite manipulación con JavaScript

```
// Cambiar variables con JS  
document.documentElement.style  
  .setProperty('--escala', '1.5');
```



Animaciones 3D con CSS

```
.tarjeta-3d {  
    perspective: 1000px;  
}  
  
.tarjeta-interior {  
    width: 100%;  
    height: 100%;  
    position: relative;  
    transform-style: preserve-3d;  
    transition: transform 0.8s;  
}  
  
.tarjeta-3d:hover .tarjeta-interior {  
    transform: rotateY(180deg);  
}  
  
.cara-frontal, .cara-trasera {  
    position: absolute;  
    width: 100%;  
    height: 100%;  
    backface-visibility: hidden;  
}  
  
.cara-trasera {  
    transform: rotateY(180deg);  
}
```



Para crear efectos 3D necesitamos:

- `perspective`: Profundidad del efecto 3D
- `transform-style: preserve-3d`: Mantiene el contexto 3D
- `backface-visibility`: Oculta el reverso de los elementos

Coreografía de animaciones

El arte de coordinar múltiples animaciones para crear una experiencia cohesiva

Secuencia

Determina qué elementos se mueven primero y cuáles después

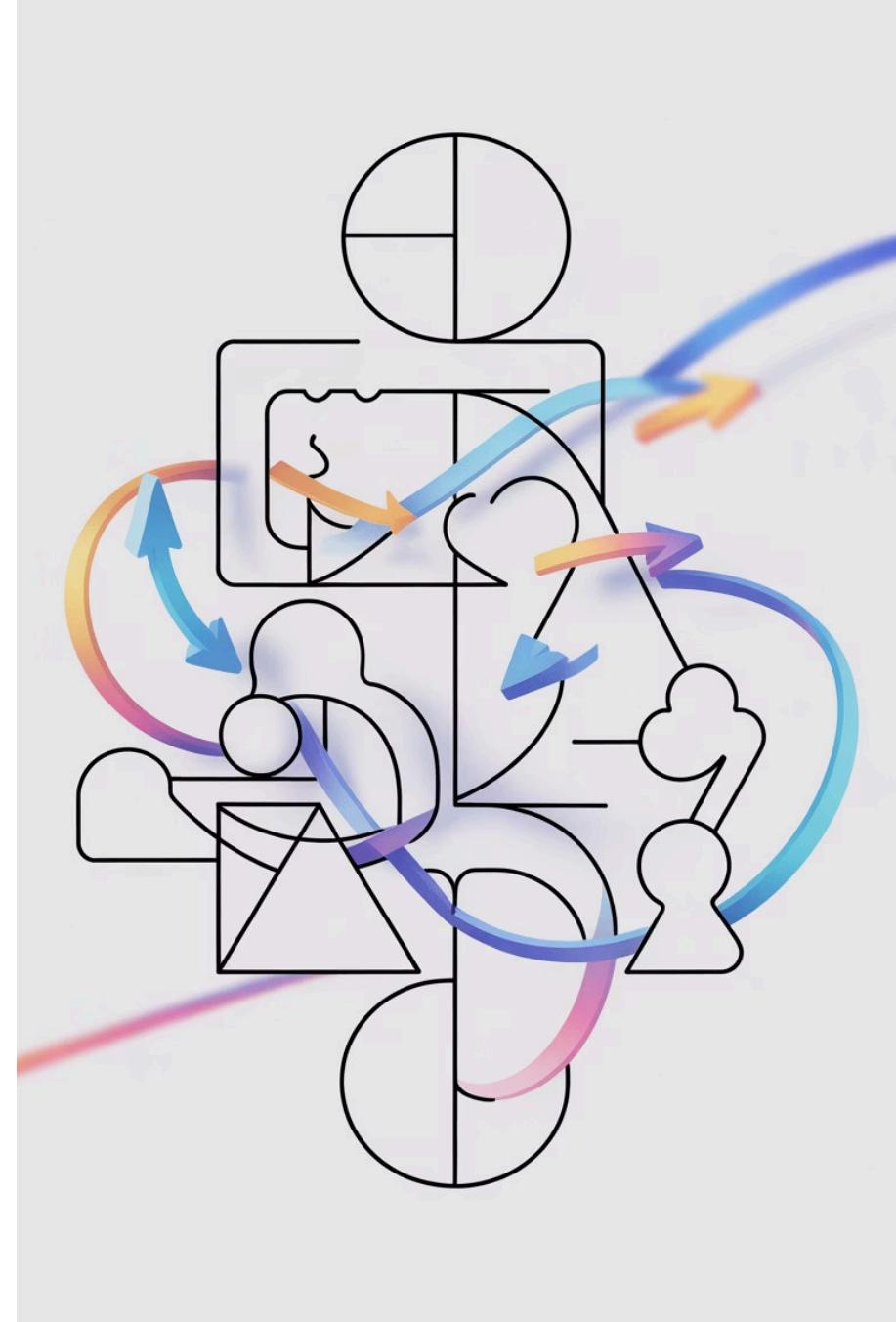
Ritmo

Varía la velocidad para crear dinamismo y jerarquía

Dirección

Crea flujos visuales coherentes (izquierda a derecha, arriba a abajo)

Las animaciones bien coreografiadas guían la mirada del usuario a través de la interfaz de forma natural e intuitiva.



Herramientas útiles



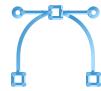
DevTools

Las herramientas de desarrollo de Chrome y Firefox incluyen un inspector de animaciones para analizar y ajustar el timing



Animate.css

Biblioteca de animaciones CSS predefinidas listas para usar



cubic-bezier.com

Generador visual de curvas de aceleración personalizadas



Animista

Generador de animaciones CSS con opciones personalizables

- Estas herramientas pueden acelerar tu flujo de trabajo, pero siempre es importante entender los conceptos subyacentes para poder adaptar las soluciones a tus necesidades específicas.

Preguntas de reflexión

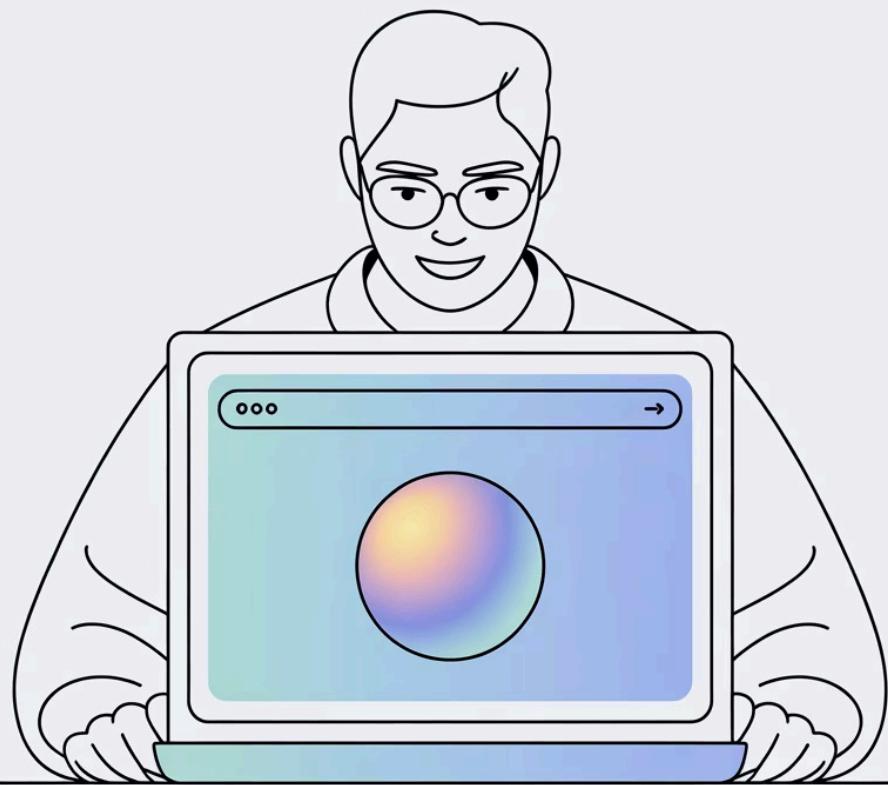
“¿Qué animaciones podrían mejorar la experiencia de usuario en el último proyecto en el que trabajaste?”

“¿Cómo equilibrarías el atractivo visual de las animaciones con las consideraciones de accesibilidad?”

“¿En qué casos crees que una animación podría ser contraproducente para la usabilidad de una interfaz?”

Recuerda:

Las mejores animaciones son aquellas que el usuario apenas percibe conscientemente, pero que mejoran significativamente su experiencia con la interfaz.



**"Bringing motion
to your vision"**

Recursos adicionales

- [MDN Web Docs: Animaciones CSS](#)
- [CSS-Tricks: Guía de transiciones](#)
- [Animista: Generador de animaciones](#)
- [Cubic-bezier.com: Editor de curvas](#)

Práctica

Intenta crear:

1. Un botón con efecto de pulso
2. Una tarjeta que se gire en 3D al hacer hover
3. Un menú de navegación con animaciones sutiles
4. Una galería de imágenes con efectos al hacer hover

¡Gracias por tu atención!

#EDCOUNIANDES

<https://educacioncontinua.uniandes.edu.co/>

Contacto: educacion.continua@uniandes.edu.co

© - Derechos Reservados: La presente obra, y en general todos sus contenidos, se encuentran protegidos por las normas internacionales y nacionales vigentes sobre propiedad Intelectual, por lo tanto su utilización parcial o total, reproducción, comunicación pública, transformación, distribución, alquiler, préstamo público e importación, total o parcial, en todo o en parte, en formato impreso o digital y en cualquier formato conocido o por conocer, se encuentran prohibidos, y solo serán lícitos en la medida en que se cuente con la autorización previa y expresa por escrito de la Universidad de los Andes.