PROMETEO

Unidad 5:

Animaciones y microinteracciones

Lenguaje de marcas y sistemas de gestión de información

Técnico Superior de DAM / DAW

Sesión 20 – Keyframes, propiedades de animación y ejemplos básicos

Fundamentos: El lenguaje del movimiento en la web

Las animaciones en CSS representan uno de los grandes saltos en la evolución del diseño web moderno. Permiten dotar a las interfaces de vida, ritmo y respuesta visual, generando una experiencia más fluida y atractiva para el usuario. Pero, más allá de su componente estético, las animaciones cumplen una función esencial: guiar la atención, indicar jerarquías visuales, y ofrecer retroalimentación inmediata sobre las acciones del usuario.

En el entorno digital actual, donde la experiencia de usuario (UX) es el centro de toda estrategia, las microinteracciones y animaciones sutiles marcan la diferencia entre una web funcional y una que realmente conecta. Una animación bien implementada puede hacer que una transición entre pantallas parezca natural, que un botón "responda" al clic o que el usuario perciba fluidez aunque el sistema esté procesando información.

Cómo funciona una animación en CSS

Las animaciones en CSS se basan en la regla @keyframes, una especie de guion que define los estados que atraviesa un elemento durante su movimiento. Cada "fotograma clave" o keyframe representa un punto en el tiempo (expresado en porcentaje o en las palabras clave from y to) con un conjunto de estilos específicos.

Por ejemplo:

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

En este ejemplo, el elemento se desplazará horizontalmente 100 píxeles a lo largo de la duración que le asignemos.

Para aplicar esta secuencia a un elemento, utilizamos la propiedad animation, que agrupa un conjunto de subpropiedades clave:

- **animation-name:** el nombre de la animación definida con @keyframes.
- **animation-duration:** la duración total (por ejemplo, 2s o 500ms).
- **animation-iteration-count:** cuántas veces se repite (un número o infinite).
- **animation-timing-function:** la curva de aceleración (por ejemplo, ease, ease-in-out, linear).
- **animation-delay:** tiempo de espera antes de comenzar la animación.
- **animation-fill-mode:** cómo se comporta el elemento antes y después de la animación.

En conjunto, estas propiedades permiten un control muy detallado del comportamiento temporal y visual. Un ejemplo completo:

```
.cuadro {
  animation-name: mover;
  animation-duration: 2s;
  animation-iteration-count: infinite;
  animation-timing-function: ease-in-out;
}
```

El resultado será un movimiento constante de ida y vuelta, con una aceleración suave al inicio y al final.

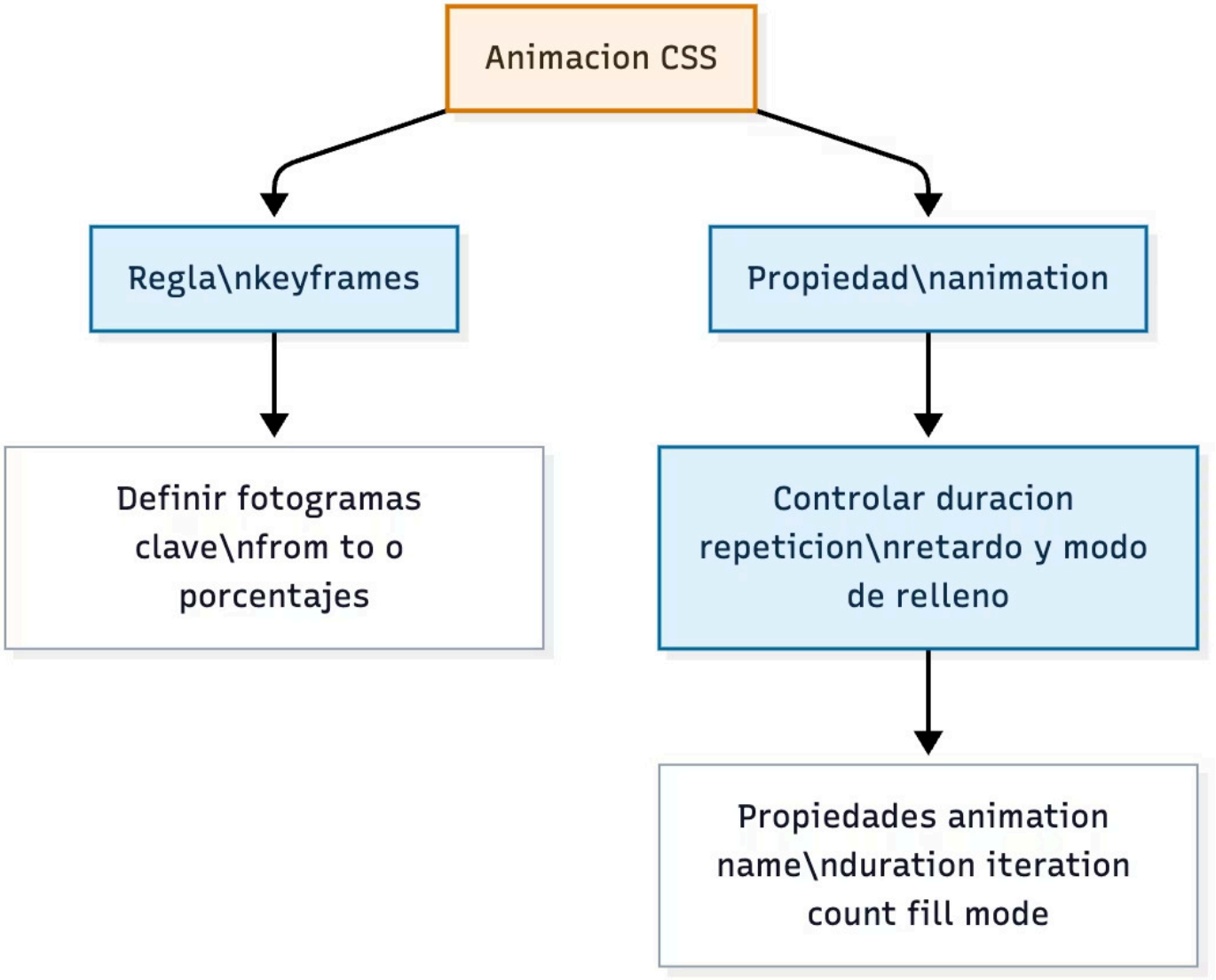
De la transición a la animación

Una confusión común es pensar que las propiedades transition y animation son equivalentes. Sin embargo, hay una diferencia esencial:

- **transition** solo define un cambio entre dos estados (por ejemplo, de azul a rojo al pasar el ratón).
- **animation** permite definir múltiples estados y repeticiones, incluso sin que haya interacción del usuario.

Mientras las transiciones responden a eventos, las animaciones pueden ejecutarse de forma autónoma, en bucle o como parte de una narrativa visual.

Esquema Visual: Cómo se estructura una animación en CSS



Descripción del esquema:

- **Animación CSS (A):** Es el proceso completo de creación de movimiento mediante código.
- **@keyframes (B):** Define los distintos estados del elemento a lo largo del tiempo. Cada keyframe actúa como una "foto fija" dentro de la secuencia.
- **Propiedad animation (C):** Se aplica al elemento HTML para ejecutar la animación definida.
- **Parámetros de control (E y F):** Permiten ajustar ritmo, repeticiones y comportamiento final.

En conjunto, el esquema ilustra cómo @keyframes proporciona el contenido (la narrativa del movimiento), mientras animation actúa como el director que decide cuándo y cómo se ejecuta.

Caso de Estudio: Los "loaders" animados – movimiento útil y perceptible

Uno de los ejemplos más extendidos y funcionales del uso de animaciones CSS son los indicadores de carga o loaders. Estos elementos se utilizan para mantener la atención del usuario durante los tiempos de espera, evitando la sensación de bloqueo o error.

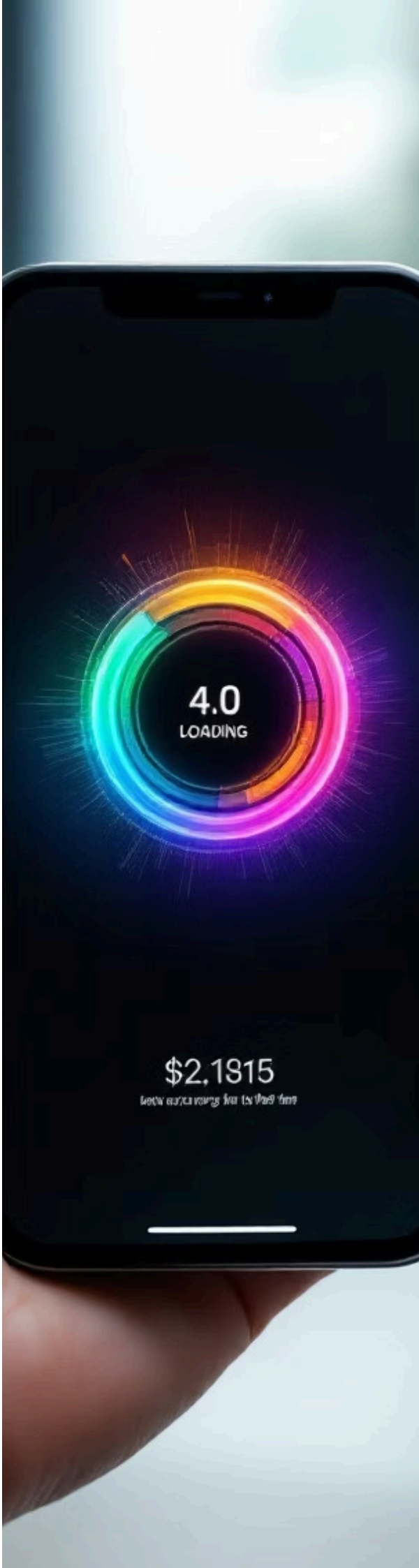
Contexto

Cuando una aplicación o sitio web necesita tiempo para procesar una acción (por ejemplo, cargar una página, enviar un formulario o recuperar datos del servidor), un indicador de carga comunica que el sistema sigue trabajando. En lugar de mostrar una pantalla estática, una animación transmite dinamismo y reduce la frustración del usuario.

Estrategia

Un "spinner" clásico puede crearse únicamente con HTML y CSS, sin imágenes ni JavaScript. Por ejemplo, un simple circular con borde parcialmente coloreado puede girar continuamente gracias a una animación con @keyframes.

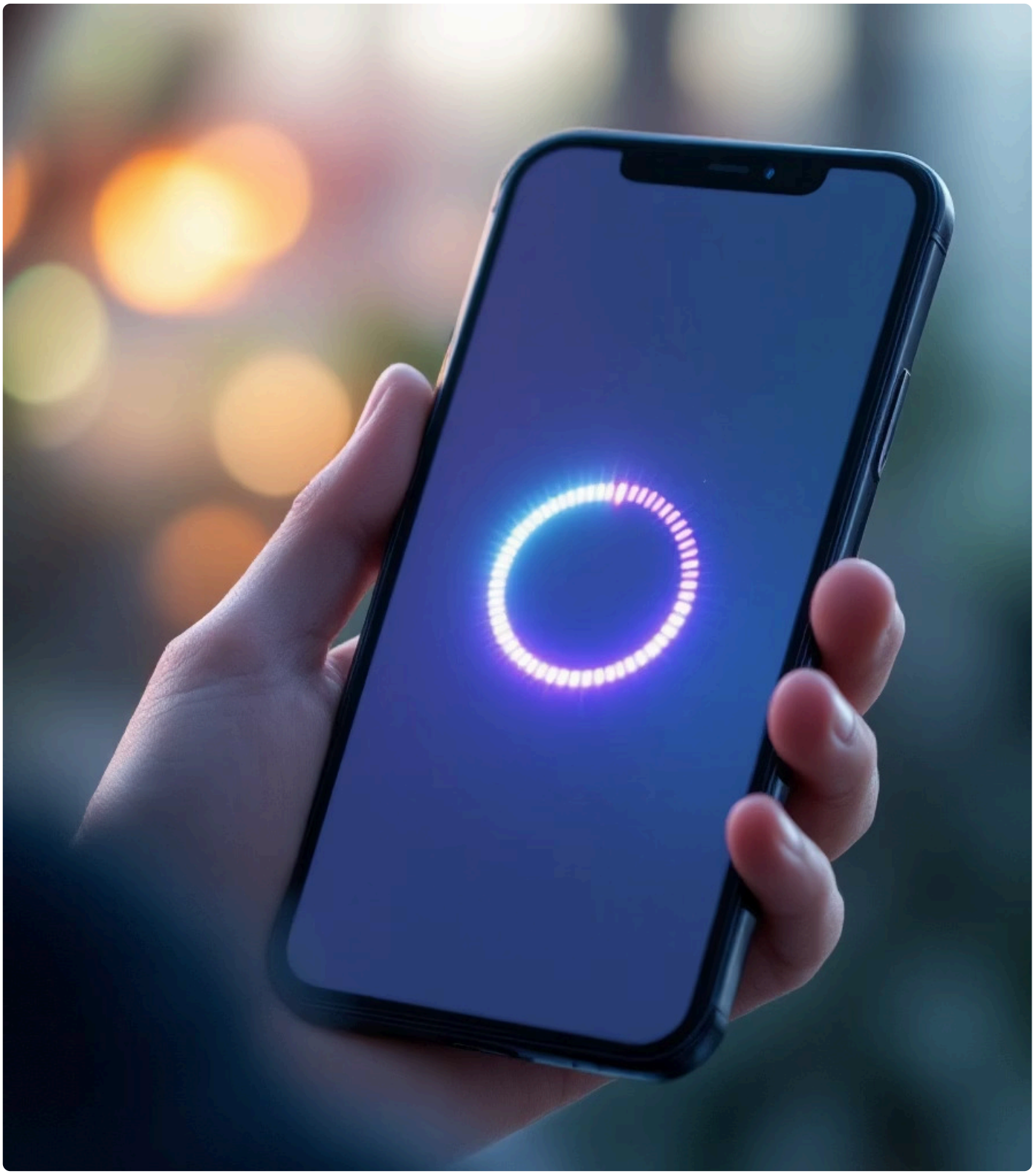
```
.loader {
  border: 6px solid #f3f3f3;
  border-top: 6px solid #3498db;
  border-radius: 50%;
  width: 40px;
  height: 40px;
  animation: girar 1s linear infinite;}

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



Resultado

- **Eficiencia:** No requiere imágenes ni scripts externos.
- **Ligereza:** Peso mínimo y carga instantánea.
- **Utilidad:** Comunica al usuario que el proceso está activo.

Empresas como Google, Spotify o Airbnb utilizan microanimaciones de este tipo en sus interfaces. Aunque son simples, generan una sensación de fluidez que contribuye al estándar de calidad visual que los usuarios esperan hoy.



Herramientas y Consejos para tu Futuro Profesional

Anima solo lo necesario (transform y opacity)

Limitar las animaciones a propiedades como transform y opacity mejora enormemente el rendimiento, ya que los navegadores las procesan mediante aceleración por hardware. Animar propiedades como width, height o margin obliga a recalcular todo el diseño, ralentizando la página.

Usa animation-fill-mode para controlar el estado final

Si no especificas esta propiedad, el elemento vuelve a su estado inicial al terminar la animación. Usar forwards hace que se mantenga en el último fotograma, útil para transiciones visuales que simulen permanencia.

Domina las curvas de velocidad con animation-timing-function

La sensación de naturalidad depende mucho de esta curva. Por ejemplo:

- **ease-in-out:** acelera y desacelera suavemente.
- **linear:** movimiento constante.
- **cubic-bezier:** define curvas personalizadas (puedes crearlas visualmente en cubic-bezier.com).

Aprovecha librerías predefinidas para ganar tiempo

Animate.css es una colección de animaciones listas para usar, perfectas para prototipos o para añadir dinamismo sin escribir cada @keyframes desde cero.

Verifica el rendimiento con las DevTools del navegador

En Chrome o Firefox, abre la pestaña "Performance" y graba la ejecución. Si la animación cae por debajo de 60 FPS, identifica qué propiedades o elementos están generando repintados innecesarios.

Piensa en la accesibilidad

Algunos usuarios pueden tener configuraciones del sistema para reducir el movimiento. Usa la media query @media (prefers-reduced-motion: reduce) para desactivar o simplificar animaciones cuando sea necesario.

Mitos y Realidades

❌ Mito: "Las animaciones CSS son solo decorativas y empeoran el rendimiento."

→ FALSO. Las animaciones, cuando se implementan correctamente, mejoran la percepción de velocidad, guían la atención del usuario y aportan coherencia visual. Si se limitan a transform y opacity, son altamente eficientes gracias al renderizado acelerado por GPU.

❌ Mito: "transition y animation son lo mismo."

→ FALSO. Aunque ambas crean movimiento, cumplen funciones distintas. transition actúa entre dos estados (por ejemplo, un color que cambia en :hover), mientras que animation define una secuencia completa con múltiples etapas o repeticiones.

✅ Realidad: "Las animaciones aportan feedback y mejoran la UX."

Los usuarios necesitan señales visuales que les confirmen que una acción ha sido procesada. Un botón que se hunde ligeramente al hacer clic o un icono que gira mientras se carga un contenido son ejemplos de microinteracciones que aportan confianza y claridad.

✅ Realidad: "Una buena animación es invisible."

Las mejores animaciones no distraen ni saturan; simplemente hacen que la interacción se sienta más fluida. Si el usuario nota la animación más que la acción, algo está mal diseñado.

Resumen Final

- Las animaciones CSS se definen con @keyframes y se aplican con la propiedad animation.
- Las subpropiedades clave controlan nombre, duración, repetición y aceleración.
- Para un rendimiento óptimo, anima transform y opacity.
- transition = cambios simples; animation = secuencias complejas.
- Las animaciones bien usadas mejoran la usabilidad y la experiencia del usuario.

Sesión 21 – Animaciones avanzadas y librerías (Animate.css, GSAP)

La evolución del movimiento digital

Cuando ya dominas los fundamentos de las animaciones en CSS —keyframes, propiedades y control de secuencias— el siguiente paso natural es aprender a llevarlas al siguiente nivel: crear animaciones más ricas, interactivas y precisas. En el diseño de interfaces modernas, las animaciones no se limitan a hacer que algo "se mueva", sino que comunican, emocionan y guían. El usuario no solo mira la pantalla: la siente.

Aquí entran en juego las librerías de animación, herramientas que te permiten ahorrar tiempo, lograr resultados más sofisticados y garantizar un rendimiento profesional. En este contexto destacan dos grandes protagonistas:

- **Animate.css**, una librería de CSS con animaciones predefinidas y listas para usar (rebotes, fades, zooms, flips, etc.).
- **GSAP (GreenSock Animation Platform)**, una potente librería de JavaScript que ofrece control total sobre cualquier elemento, propiedad y línea de tiempo de animación.

Animate.css: rapidez y simplicidad

Animate.css es ideal para proyectos donde quieres añadir movimiento sin complicaciones. Solo necesitas importar su hoja de estilos y aplicar las clases correspondientes al elemento HTML. Por ejemplo:

```
<h1 class="animate__animated animate__bounce">¡Hola mundo!</h1>
```

En este ejemplo, el título "salta" suavemente hacia la pantalla. No necesitas escribir ni una línea de JavaScript. Es perfecta para prototipos, presentaciones, landings y proyectos educativos, donde la prioridad es la agilidad.

GSAP: potencia y control absoluto

GSAP, en cambio, es otra historia. Es el estándar de la industria para animación avanzada en la web. Permite animar no solo elementos HTML, sino también SVG, Canvas, WebGL o incluso propiedades personalizadas de JavaScript.

Su mayor fortaleza está en la precisión temporal y la sincronización. Puedes crear líneas de tiempo, encadenar secuencias, controlar la velocidad, pausar, invertir y sincronizar efectos con eventos del usuario.

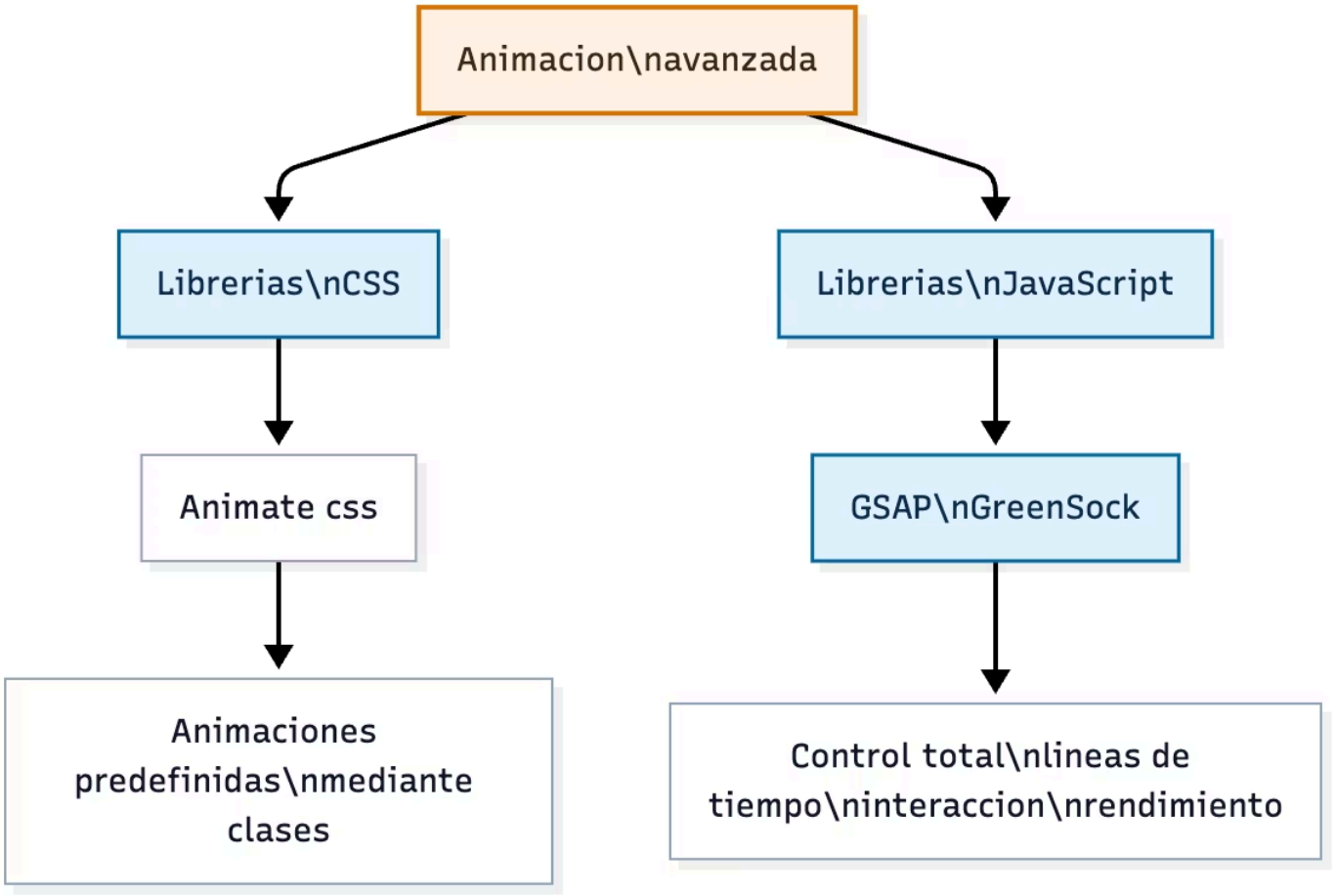
Un ejemplo básico:

```
gsap.to(".caja", {
  x: 200,
  duration: 1,
  ease: "power2.inOut"
});
```

Esta simple línea mueve un elemento con clase .caja 200 píxeles hacia la derecha en un segundo. Pero GSAP va mucho más allá: puedes orquestar animaciones simultáneas, coordinar escenas, e incluso integrar desplazamientos (scroll) y eventos interactivos.

Ambas librerías —Animate.css y GSAP— se complementan. Animate.css te da velocidad y facilidad; GSAP te da poder y libertad total. Saber cuándo usar una u otra es lo que te convierte en un profesional eficiente y estratégico.

Esquema Visual: Ecosistema de animaciones avanzadas



Descripción del esquema:

- **A – Animación avanzada:** representa el conjunto de técnicas y herramientas que superan el nivel básico de las animaciones CSS.
- **B – Librerías CSS:** son soluciones visuales predefinidas, rápidas de aplicar, centradas en la eficiencia y la estética inmediata.
- **D – Animate.css:** ejemplo de librería basada en CSS puro, perfecta para proyectos rápidos.
- **C – Librerías JavaScript:** ofrecen dinamismo avanzado e interactividad.
- **F – GSAP (GreenSock):** proporciona control absoluto sobre la animación, sincronización con eventos y optimización del rendimiento.

El esquema ilustra cómo ambas vías —CSS y JavaScript— cubren diferentes niveles de complejidad: mientras Animate.css simplifica, GSAP amplifica.

Caso de Estudio: La magia del movimiento en los sitios ganadores de Awwwards

Contexto

Cada año, los sitios web más innovadores del mundo son premiados en Awwwards, una plataforma que reconoce la excelencia en diseño, creatividad y experiencia de usuario. Si visitas cualquiera de estos proyectos, notarás un elemento común: las animaciones no son adorno, son parte del relato. Desde sutiles transiciones de texto hasta experiencias inmersivas en 3D, todo está diseñado para emocionar y mantener la atención.

Estrategia

Gran parte de estas experiencias visuales se logra gracias a GSAP, que se ha convertido en un estándar en agencias de diseño como Media.Monks, Resn o Dogstudio. La mayoría de los sitios premiados utilizan GSAP para:

- Sincronizar animaciones con el scroll (usando el plugin ScrollTrigger).
- Crear transiciones fluidas entre páginas sin recargar.
- Generar efectos de profundidad y movimiento 3D con SVG y WebGL.
- Coordinar múltiples elementos en una misma línea de tiempo.

Por ejemplo, en el sitio "Species in Pieces" —un proyecto galardonado por su innovación visual — cada especie animal se construye mediante animaciones vectoriales que se transforman suavemente unas en otras. Cada forma responde a interacciones del usuario, generando una experiencia educativa, emocional y artística a la vez.

Resultado

El uso de GSAP en estos proyectos no solo embellece el resultado, sino que garantiza un rendimiento fluido, incluso con animaciones complejas. A diferencia de soluciones improvisadas en CSS o con jQuery, GSAP optimiza automáticamente los cuadros por segundo y la gestión de la memoria. En el competitivo mundo del diseño digital, la diferencia entre una interfaz funcional y una inolvidable suele estar en estos detalles.

Herramientas y Consejos para tu Futuro Profesional

Usa Animate.css para prototipos rápidos o interfaces ligeras

Si estás desarrollando una web estática, un portfolio o una landing de producto, Animate.css es tu mejor aliada. Solo necesitas enlazar el archivo CSS y añadir las clases animate__animated junto al efecto deseado (por ejemplo, animate__fadeIn, animate__bounce, animate__zoomIn). Puedes ver todas las opciones en su galería oficial.

Aprende los fundamentos de GSAP paso a paso

GSAP puede parecer abrumadora al principio, pero su sintaxis es muy legible. Comienza con lo esencial:

```
gsap.to(".logo", { opacity: 1, duration: 1 });
```

Luego experimenta con secuencias:

```
const tl = gsap.timeline();
tl.to(".logo", { y: -50, duration: 0.5 })
  .to(".menu", { opacity: 1, duration: 0.3 }, "-=0.2")
  .to(".boton", { scale: 1.2, duration: 0.4 });
```

Aquí creas una línea de tiempo (timeline) donde los elementos aparecen de forma sincronizada.

Combina GSAP con ScrollTrigger

Este plugin oficial permite vincular animaciones al desplazamiento del usuario. Por ejemplo, puedes hacer que una imagen se mueva o un texto aparezca cuando entra en el viewport. Es una herramienta esencial para sitios con storytelling visual o diseño "scroll-based".

Optimiza el rendimiento: 60 FPS es la meta

Usa las DevTools del navegador para medir el rendimiento. Evita animar propiedades que afecten al layout (como width o top). GSAP ya se encarga de gran parte de esta optimización, pero una estructura HTML limpia y estilos bien organizados siempre ayudarán.

Integra GSAP con frameworks modernos

GSAP se adapta perfectamente a React, Vue y Angular. En React, por ejemplo, puedes crear animaciones que se activen al montar o desmontar un componente, lo que mejora la fluidez en las transiciones entre vistas.

Inspírate en las grandes agencias

Explora sitios como Awwwards o Codrops para analizar cómo las animaciones cuentan historias. La mejor forma de aprender es observar cómo los profesionales integran movimiento, interacción y emoción.

Recursos imprescindibles:

- **Animate.css:** librería completa y documentación.
- **GreenSock.com:** demos, plugins y ejemplos oficiales.
- **Snorkl.tv – GSAP Express:** curso gratuito y actualizado.
- **CodePen.io:** comunidad donde puedes experimentar y ver código de otros desarrolladores.

Mitos y Realidades

❌ Mito: "Las librerías de animación son pesadas y ralentizan la web."

→ FALSO. GSAP, por ejemplo, está obsesivamente optimizada para el rendimiento. Pesa menos de 40 KB comprimida y se ejecuta de forma más eficiente que la mayoría de scripts personalizados. De hecho, muchas animaciones complejas hechas con jQuery o CSS puro resultan menos fluidas que una animación bien implementada con GSAP, que maneja internamente la sincronización de frames y el rendering.

❌ Mito: "Todo se puede hacer con animaciones CSS; no hace falta JavaScript."

→ FALSO. CSS tiene limitaciones: no permite crear líneas de tiempo, sincronizar varios elementos o responder dinámicamente a la interacción del usuario (por ejemplo, seguir el cursor o responder al scroll). Para ese nivel de control, GSAP o similares son indispensables.

✅ Realidad: "Animate.css acelera el trabajo en entornos productivos."

Aplicar animaciones de entrada o salida a modales, menús o textos puede llevar segundos. En equipos de desarrollo ágil o diseño UI, esta eficiencia es clave.

✅ Realidad: "GSAP es el estándar de la industria para animaciones web profesionales."

Desde banners publicitarios hasta experiencias 3D interactivas, GSAP se utiliza en la mayoría de los proyectos que buscan animaciones precisas, fluidas y de alto impacto visual. Grandes marcas como Google, Adobe y Sony confían en ella para sus interfaces interactivas.

Resumen Final

- Animate.css: librería CSS con animaciones predefinidas; rápida y fácil de usar.
- GSAP: librería JavaScript para animaciones complejas, precisas e interactivas.
- Animate.css = velocidad; GSAP = control.
- Los sitios premiados en Awwwards demuestran el poder del movimiento planificado.
- Las librerías modernas son ligeras y están optimizadas para el rendimiento.
- Para animaciones interactivas o dependientes del usuario, el camino es JavaScript.