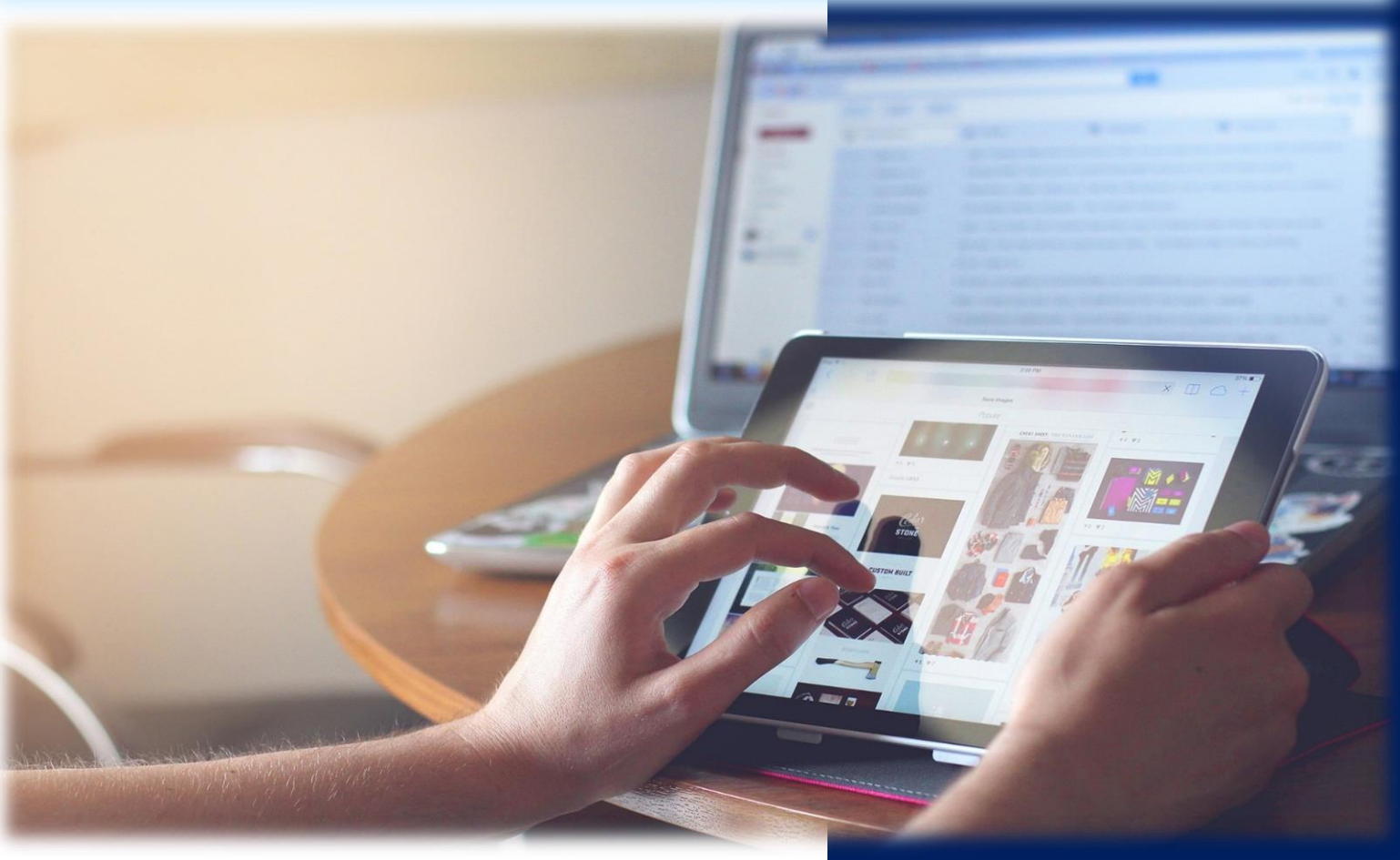
2024/25

U.D.4. ELEMENTOS MULTIMEDIA

4.2 Sonido, vídeo y animaciones



ÍNDICE

1. El sonido 3
1.1 Conceptos básicos 3
1.2 Formatos de sonido 4
1.2.1 Conversión de formatos 7
1.3 Optimizaciones en archivos de audio para web 9
1.4 Sitios web para la descarga de sonidos 9
2. El vídeo en la web 10
2.1 Conceptos básicos 10
2.2 Formatos de vídeo 10
2.2.1 Conversión de formatos 12
2.3 Optimización de archivos de vídeo 13
2.4 Sitios web para la descarga de vídeos 14
3. Animaciones 15
3.1 La regla @keyframes 15
3.2 La propiedad ‘animation’ 16
3.3 La propiedad ‘transition’ 18
3.4 Ejemplos de animaciones 2D 23
3.4.1 La animación FLASH 23
3.4.2 La animación CLICK 23
3.4.3 La animación CHICLE 24
3.4.4 La animación VIBRADOR 24
3.4.5 La animación CARTEL 25
3.4.6 La animación BALANCEO 25
3.4.7 La animación VAIVÉN 26
3.4.8 La animación CAÍDA o DESCOLGADO 26
3.4.9 Animación ENTRADA y SALIDA ROTATIVOS 27
3.4.10 Animación REBOTE 27
3.4.11 Animación REBOTEIN 28
3.4.12 Animación REBOTEOUT 28

3.4.13 Animación FADEIN 29
3.4.14 Animación VELOZIN 29
3.4.15 Animación VELOZOUT 30
3.4.16 Animación GIRO 30
3.4.17 Animación ARROW 31
3.4.18 Animación SOMBRA MÓVIL 31
3.5 Ejemplos de animaciones 3D 32
3.5.1 GIROX 32
3.5.2 GIROY 33
3.5.3 GIRO OUT 3D 34
3.5.4 TRANSLATE Z 34
3.5.5 CUBO 35
3.5.6 BOLA DEL MUNDO 36
3.5.7 CUBO RUBYK 36
3.5.8 PIRÁMIDE 3D 37
3.6 Integración de vídeo en una animación 38
3.7 Integración de sonido en una animación 39
3.7.1 Animación de un TAMBOR 39
3.7.2 Animación PASO A NIVEL 42
3.7.3 Animación LATIDO DE CORAZÓN 43

1. El sonido

Se define por **sonido** a la sensación producida en el órgano del oído por el movimiento vibratorio de los cuerpos, transmitido por un medio elástico como el aire. **El audio es la técnica relacionada con la reproducción, grabación y transmisión del sonido**.

1.1 Conceptos básicos

Existe un conjunto de características de los ficheros de sonido que han de tenerse en cuenta a la hora de escoger uno u otro. Es necesario un proceso de **codificación**, puesto que el sonido es analógico, y, dado que la información se transmite a través de la red, esta presentará un formato digital, por lo que estará formada por un grupo de bits (ceros y unos).

Las principales **características** son:

1. **Frecuencia**. Es el número de vibraciones por segundo que da origen al sonido analógico. El espectro de un sonido se caracteriza por su rango de frecuencias, medido en hercios (**Hz**). El oído humano capta aquellos sonidos comprendidos en un rango de frecuencias entre 20Hz y 20.000 Hz.

2. **Tasa de muestreo** (*sample rate*). Un audio digital es una secuencia de ceros y unos que se obtiene del muestreo de la señal analógica. La tasa de muestreo define cada cuánto tiempo se tomará el valor de la señal analógica para generar el audio digital. Esta tasa se mide en hertzios (Hz).

3. **Resolución**. Es el número de bits utilizados para almacenar cada muestra de la señal analógica. Una resolución de 8 bits proporciona 256 (25) niveles de amplitud. Un audio digital tendrá más calidad cuanto mayor sea su resolución.

4. **Velocidad de transmisión** (*bit rate*). Define la cantidad de espacio físico (en bits) que ocupa un segundo de duración de ese audio. El audio tendrá más calidad cuanto mayor sea su bit rate, y el archivo que lo contiene tendrá mayor peso. Esta magnitud se utiliza, sobre todo, en el formato MP3 de audio más destinado a la descarga por Internet.

5. **CBR/VBR** (Constant/Variable Bit Rate). CBR indica que el audio ha sido codificado, manteniendo el bit rate constante a lo largo del clip de audio, mientras que VBR varía entre un rango máximo y mínimo en función de la tasa de transferencia.

6. **Códec**. Un códec es un **algoritmo** especial que reduce el número de bytes que ocupa un archivo de audio. Los archivos codificados con un códec específico requieren el mismo códec para ser decodificados y reproducidos. El códec más utilizado en audio es el **MP3**.

7. **Decibelio**. Unidad de medida del volumen o intensidad de un sonido. El silencio o ausencia de sonido se cuantifica como 0 dB, y el umbral del dolor para el oído humano se sitúa en torno a los 130-140 dB.

¿Cuánto ocupa un audio MP3 con una duración de 5 minutos y una tasa de bit de 128 kbit/s bit rate habitual de los ficheros en MP3)?

- Duración: 5 minutos = 300 segundos.
- Tasa de bit = 128 kbit/s.
- **Tamaño = Duración x Tasa de bit** = 300 segundos x 128 kbit/s = 38.400 kbit.

(**Importante**: para pasar de kbit a bit, el factor de conversión es 1024 si usamos el sistema binario; si utilizamos el sistema internacional, el factor de conversión es 1000).

SISTEMA INTERNACIONAL:
- Tamaño (bit) = 38.400 kbit x 1000 bit/kbit = 38.400.000 bit.
- Tamaño (B) = 38.400.000 bit / (8 bit/B) = 4.800.000 B.
- Tamaño (MB) = 4.800.000 B x (1 kB/1000 B) x (1 MB/ 1000 kB) = **4,8 MB.**

SISTEMA BINARIO:
- Tamaño (bit) = 38.400 kbit x 1024 bit/kbit = 393.216 x 10^2bit.
- Tamaño (B) = 393.216 x 10^2bit / (8 bit/B) = 4.915.200 B.
- Tamaño (MiB) = 4.915.200 B x (1 kiB/1024 B) x (1 MB/ 1024 kiB) = **4,68 MiB.**

1.2 Formatos de sonido

El principal formato de sonido reproducido por cualquier navegador web es **WAV**. Aunque otros formatos también serán reproducidos previa carga del correspondiente plugin, como es el caso de Real Audio o Quicktime.

A la hora de elegir un formato de sonido, debe tenerse en cuenta el uso que se le va a dar a ese sonido. Si el sitio web objeto de diseño lo que necesita es alojar una emisora de radio que desea realizar una transmisión en tiempo real, habrá que optar por formatos del tipo de

Windows Media Audio, mientras que, si su objetivo es la venta de sonido, suele escogerse MP3.

- **WAV** (Waveform Audio File Format). Es el estándar en los ordenadores con sistema operativo Windows. Admite sonido mono y estéreo con diferentes frecuencias de muestreo que van desde los 8 Khz hasta los 192 Khz. Los archivos tienen extensión **.wav*.
    - *Es ideal para guardar **audios originales** a partir de los cuales se puede comprimir y guardar en distintos tamaños de muestreo para publicar en la web.*
    - *Es un formato de **excelente calidad** de audio. Sin embargo, produce **archivos de un peso enorme**.*
    - *Los archivos WAV se pueden guardar con **distintos tipos de compresión**. Las más utilizadas son la compresión **PCM** y la compresión **ADPCM**.*

- **MP3** (Motion Picture Experts Group). Formato de archivo de sonido que tiene una **alta calidad con un tamaño muy reducido** (tiene un alto grado de compresión). Los archivos tienen extensión **.mp3*.
    - *Es ideal para publicar audios en la web. Se puede escuchar desde la mayoría de reproductores.*
    - *La **transformación** de WAV a MP3 o la publicación directa de una grabación en formato MP3 es un proceso **fácil** y al alcance de los principales editores de audio.*
    - *Tiene un **enorme nivel de compresión** respecto al WAV. En igualdad del resto de condiciones reduciría el tamaño del archivo de un fragmento musical con un factor **entre 1/10 y 1/12**.*
    - *Presentan una **mínima pérdida de calidad**.*

- **OGG**. Formato **contenedor de audio y vídeo** con licencia pública general de **GNU**. Por tanto, es **libre y de código abierto**, a diferencia del formato MP3. Los archivos tienen extensión *.*ogg*.
    - *Muestra un **grado de compresión similar al MP3** pero según los expertos en música, la **calidad** de reproducción es ligeramente **superior**.*

- ***No todos los reproductores multimedia son capaces de leer*** *por defecto este formato. En algunos casos es necesario instalar los códecs o filtros oportunos.*

- **RA** (Real Audio). Formato empleado en la transmisión de sonido por Internet en tiempo real sin necesidad de descarga previa de un archivo. Es un modo de streaming que permite variar la velocidad de transmisión para adaptarse a las características de la conexión del usuario.

- **WMA** (Windows Media Audio). Formato de compresión con pérdida creado como alternativa al MP3 y Real Audio.

- **AAC** (Advanced Audio Coding). Formato estándar creado por MPEG (Moving Pictures Expert Group), que utiliza una tasa de bits variable y tiene un rendimiento superior al MP3.

- **AIFF** (Audio Interchange Format File). Es el estándar en los sistemas Apple. Es multicanal y soporta varias profundidades de bit y diferentes frecuencias de muestreo.

- **MIDI** (*Musical Instrument Digital Interface*). El formato MIDI en realidad no resulta de un proceso de digitalización de un sonido analógico y, por ello, no es un formato de audio propiamente dicho. Sus **características** son las siguientes:
  - Un archivo de extensión **.mid* **almacena secuencias** de dispositivos MIDI (sintetizadores) donde se recoge qué instrumento interviene, en qué forma lo hace y cuándo.
  - Los archivos MIDI se pueden **editar y manipular mediante programas especiales** y distintos de los empleados para editar formatos WAV, MP3, etc. El manejo de estos programas suele conllevar ciertos conocimientos musicales.
  - Los archivos MIDI permiten **audios de cierta duración con un reducido peso**. Esto es debido a que no guardan el sonido sino la información o partitura necesaria para que el ordenador la componga y reproduzca a través de la tarjeta de sonido.

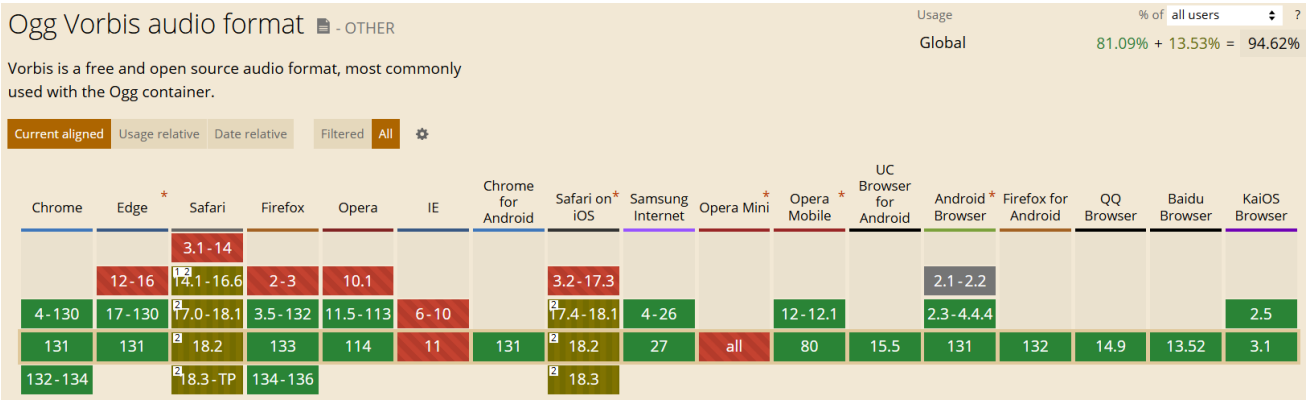
- Se suelen utilizar en **sonidos de fondo de páginas HTML** o para escuchar composiciones musicales de carácter instrumental.

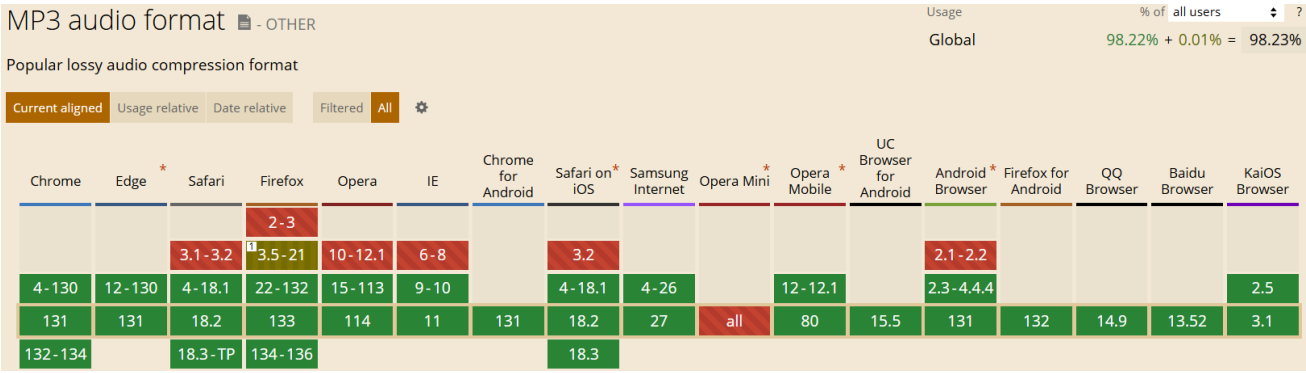
1.2.1 Conversión de formatos

A la hora de utilizar un formato de audio en nuestra web, debemos tener en cuenta el tipo de **formato** que tendrá para comprobar **qué navegadores van a ser capaces de soportarlo**.

En la siguiente tabla se muestra una relación de los navegadores y los tipos de formatos que soportan:

Formato	IE 9	Firefox 7	Opera 11.5	Chrome 14.0	Safari 5.0
OGG	No	Sí	Sí	Sí	No
MP3	Sí	No	No	Sí	Sí
WAV	No	Sí	Sí	Sí	Sí





Wav audio format - OTHER

Usage % of all users ?
Global 97.76%

Waveform Audio File Format, aka WAV or WAVE, a typically uncompressed audio format.

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
4-7		3.1-3.2	2-3	10.1								2.1-2.2				
8-130	12-130	4-18.1	3.5-132	11.5-113	6-10		3.2-18.1	4-26		12-12.1		2.3-4.4.4				2.5
131	131	18.2	133	114	11	131	18.2	27	all	80	15.5	131	132	14.9	13.52	3.1
132-134		18.3-TP	134-136				18.3									

En HTML5 podremos añadir formatos de audio según el formato que soporte cada navegador. Así, referenciamos a todos ellos y cada navegador usará el que pueda reproducir.

Para poder tener audios en distintos formatos, tendremos que usar **conversores**. Algunas **herramientas gratuitas** para convertirlos son:

- **Free Studio**: esta herramienta es capaz de convertir vídeo, audio e imagen. Tiene un fácil manejo y es adecuada para los diseñadores que trabajan con audio en la web. Tiene solo versión para Windows.

- **Audacity**: esta es una herramienta más avanzada para grabar y editar. Tiene como ventaja que es libre y de código abierto. Presenta muchas funcionalidades, además de poder convertir los archivos. Podremos crear nuestros propios sonidos. Posee una versión para Linux y Windows.

- **VLC Media Player**. Es un reproductor y framework multimedia, libre y de código abierto desarrollado por el proyecto VideoLAN. Algunas de sus características son:
    - Creación de efectos de audio y vídeo.
    - Grabación del escritorio.
    - Conversión de archivos de audio y vídeo.
    - Transmisión de información y datos a través de la red local, así como en Internet.
    - Reproducción y descarga de forma segura los vídeos de YouTube.
    - Añadir marca de agua en el video.

Existen también muchas **páginas web** que son capaces de convertir archivos de audio en distintos formatos. Un ejemplo puede ser **Switchr**, pero existen muchas más.

1.3 Optimizaciones en archivos de audio para web

Para **optimizar el peso del archivo** de audio que vamos a subir a nuestra web será necesario utilizar un editor para reproducir alguno o algunos de los siguientes parámetros:

1) **Tasa de muestreo**. Definir valores inferiores: 44100Hz, 22050 Hz, 11025Hz, etc.
2) **Resolución**. Establecer resoluciones más pequeñas: 32 bits, 16 bits, 8 bits, 4 bits, etc.
3) **Duración**. En ocasiones se puede utilizar un **fragmento más corto** que **reproducido en bucle** cubre el tiempo suficiente de acompañamiento musical. A éstos se les llama **loops**.
4) **Calidad estéreo/mono**. La **reducción a calidad "mono"** reduce considerablemente el peso del archivo. La calidad de reproducción "mono" para la mayoría de audios y de usuarios es apenas perceptible.
5) **Formato**. Es **preferible usar el formato MP3 u OGG** en lugar del WAV por su potente factor de compresión y su aceptable calidad de audio.

1.4 Sitios web para la descarga de sonidos

A continuación, se propone un listado de bancos para la descarga de audios gratuitos para usarlos en proyectos propios.

- https://www.elongsound.com/
- https://sshhtt.com/
- https://www.soundeffectsplus.com/
- https://es.audiomicro.com/
- https://www.soundsnap.com/
- https://findsounds.com/

2. El vídeo en la web

Al igual que ocurre en el caso de los sonidos, la incorporación de vídeos escogidos adecuadamente y que aporten un valor añadido al sitio es una de las herramientas más poderosas que se presentan en la actualidad para captar la atención del usuario.

2.1 Conceptos básicos

Hay una serie de **características** de los ficheros de vídeo que han de tenerse en cuenta a la hora de ser usados en un proyecto web. Al igual que en el caso del sonido, dado que la información se transmite a través de la red, esta presentará un formato **digital** formada por un grupo de bits (ceros y unos).

1. **Duración**. Se trata del tiempo que dura la reproducción de un vídeo. Se expresa mediante la estructura: *HH:MM:SS*.

2. **Tamaño del clip**. Espacio, en MB, que ocupa un vídeo al ser almacenado en un archivo en cualquier dispositivo.

3. **Tamaño del fotograma**. Se denomina fotograma a cada una de las imágenes que componen un vídeo; su tamaño se mide en píxeles, diferenciándose entre dos magnitudes, la altura y la anchura.

4. **Relación de aspecto**. Se trata de la relación que existe entre el ancho y el alto de un fotograma y, por tanto, del vídeo, puesto que normalmente se mantiene el mismo tipo de formato a lo largo de toda la grabación. Las más conocidas son 4:3 y 16:9.

5. **Número de fotogramas por segundo**. Número de imágenes o fotogramas que, al ser visitadas, una detrás de otra, proporcionan la sensación de movimiento de la imagen.

2.2 Formatos de vídeo

Al igual que ocurría con el caso del audio, visto más arriba, para elegir un formato de vídeo que embeber en el sitio web debe tenerse en cuenta su uso. Por **ejemplo**, para la transmisión de vídeo por Internet en tiempo real, el formato más empleado es Real Video.

- **OGG**. Formato de archivo de vídeo o audio diseñado para hacer streaming online, tanto de vídeo como de audio. Está basado en un estándar libre y abierto, y está impulsado por Mozilla, Opera y Firefox.

- **WebM**. Formato contenedor de elementos multimedia. Se trata del formato estándar utilizado en HTML5. Es gratuito e impulsado por los mismos que apoyan el formato OGG (Mozilla, Opera y Firefox).

- **MP4**. Se trata de uno de los formatos más comunes en la actualidad, puesto que es válido tanto para descarga como para realizar streaming online. Además, ofrece una alta tasa de compresión frente a una baja pérdida de calidad. Es impulsado por Apple y Microsoft.

- **Real Video**. Formato empleado en la transmisión de vídeo por Internet en tiempo real sin necesidad de descarga previa de un archivo. Para su visualización es necesario tener instalado el reproductor asociado Real Player.

- **AV1**. AOMedia Video 1 es un formato de compresión de video abierto y libre de derechos de autor diseñado para las transmisiones de video a través de Internet. Fue desarrollado como sucesor de VP9 por la Alliance for Open Media (AOMedia). Este formato lo utilizan plataformas como Vimeo, Netflix, Youtube, Twitch, etc.

- **mov**. Este formato permite hacer streaming online. Se trata de la extensión habitual de los ficheros de Quick Time, reproductor que permite contener diferentes archivos de diferentes formatos.

- **wmv**. Formato de vídeo utilizado en las aplicaciones de streaming a través de la red.

- **Flash Video**. Formato contenedor utilizado comúnmente para entregar video a través de Internet.

2.2.1 Conversión de formatos

Al igual que sucede con el audio, podemos convertir los distintos formatos de vídeo mediante aplicaciones sencillas que ofrecen multitud de opciones. También tenemos el mismo problema con los navegadores: **no existe un formato estándar**.

En **HTML5**, los formatos que más se usan tendrán los siguientes códecs:

- **MP4** = H.264 o MPEG-4
- **OGG** y **OGV** = Theora.
- **WEBM** = VP8 o VP9.
- **AV1=** AV1.

MPEG-4/H.264 video format - OTHER

Usage % of all users ?
Global 98.22%

Commonly used video compression format.

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
												2.1-2.2				
			2-20									[1] 2.3				
		3.1	21-34	10-24	6-8							3-4.3				
4-130	12-130	3.2-18.1	35-132	25-113	9-10		3.2-18.1	4-26		12-12.1		4.4-4.4.4				2.5
131	131	18.2	133	114	11	131	18.2	27	all	80	15.5	131	132	14.9	13.52	3.1
132-134		18.3-TP	134-136				18.3									

Ogg/Theora video format - OTHER

Usage % of all users ?
Global 4.98%

Free lossy video compression format.

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
	12-16		2-3	10.1												
4-119	17-121		3.5-129	11.5-105												
[1] 120-130	[1] 122-130	3.1-18.1	130-132	[1] 106-113	6-10		3.2-18.1	4-26		12-12.1		2.1-4.4.4				2.5
[1] 131	[1] 131	18.2	133	[1] 114	11	131	18.2	27	all	80	15.5	131	132	14.9	13.52	3.1
[1] 132-134		18.3-TP	134-136				18.3									

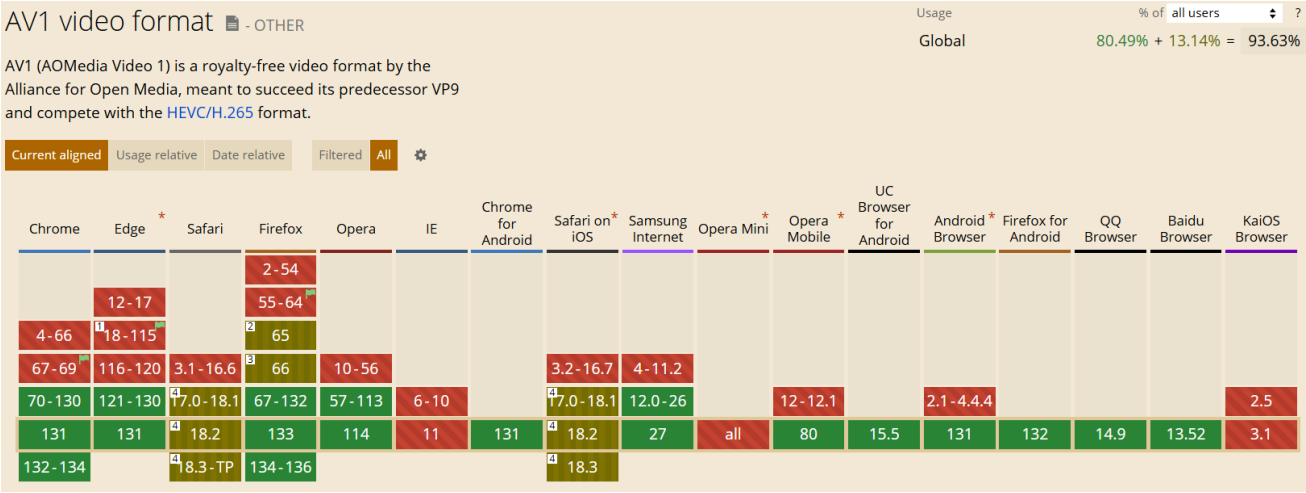
WebM video format - OTHER

Usage % of all users ?
Global 94.15% + 3.27% = 97.43%

Multimedia format designed to provide a royalty-free, high-quality open video compression format for use with HTML5 video. WebM supports the video codec VP8 and VP9.

Current aligned | Usage relative | Date relative | Filtered | All

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
		3.1-5.1														
		[3] 6-12														
		[4] 12.1-13.1					3.2-12.1									
4-5	12-13	[4 5 6] 14	2-3.6	10.1			[4] 12.2-13.7									
[1] 6-24	[1 2] 14-18	[7] 14.1-15.6	[1] 4-27	[1] 11.5-15	6-8		[4 5] 14-17.3	[1] 4				2.1-2.2				
25-130	79-130	16.0-18.1	28-132	16-113	[3] 9-10		17.4-18.1	5-26		12-12.1		[1] 2.3-4.4.4				2.5
131	131	18.2	133	114	[3] 11	131	18.2	27	all	80	15.5	131	132	14.9	13.52	3.1
132-134		18.3-TP	134-136				18.3									



Algunas **herramientas gratuitas** para la conversión pueden ser:

- **Miro Video Converter**: utilidad sencilla para convertir cualquier formato de vídeo, incluso OGV y WEBM.

- **FreeStudio**: es una potente herramienta que no solo trabaja en formatos de vídeo, sino que también con audio e imagen. Solo tiene versión para Windows.

- **AtubeCatcher**: además de poder convertir archivos en muchos formatos, nos da la posibilidad de descargar vídeos desde distintos sitios web, como YouTube, Google Videos o Vimeo.

2.3 Optimización de archivos de vídeo

Para optimizar el peso del archivo de vídeo será necesario editarlo para establecer algunos de los siguientes parámetros.

1) El **códec de compresión** de vídeo utilizado.
2) **Método de BitRate**. Utilizar un bitrate variable VBR puede optimizar la calidad del vídeo y repercutir en el peso final del archivo frente a un bitrate constante CBR.
3) **Velocidad de transmisión** (bitrate). Configurar bitrates más bajos: 1000 Kbps, 768 Kbps, 360 Kbps, etc.
4) **Dimensiones**. Cuanto más pequeña sea la altura y la anchura en píxeles de los fotogramas de un vídeo, menos tamaño ocupará su archivo.

5) **Velocidad de fotogramas**. Se puede reducir el número de fotogramas por segundo que mostrará el vídeo: 30, 24, 20, 16, etc.
6) **Fotogramas Clave**. Durante la compresión también se puede indicar cada cuánto se guardará un fotograma completo (fotograma clave): 24, 48, 96, 128, etc. Cuanto mayor sea esta cadencia más bajo será el peso del archivo resultante.

2.4 Sitios web para la descarga de vídeos

A continuación, se propone un listado de sitios web para la descarga de vídeos gratuitos libres de derechos de autor o con licencias Creative Commons para utilizarlos en proyectos propios.

- https://coverr.co/
- https://pixabay.com/
- https://www.videvo.net/
- https://es.videezy.com/
- https://lifeofvids.com/
- https://dareful.com/
- https://www.pexels.com/es-es/

3. Animaciones

Las **animaciones** son **transiciones entre elementos** que se definen mediante reglas de estilo en CSS. Se distinguen dos partes fundamentales en la creación de dichas animaciones:

1) Los **fotogramas**, que indican el estado inicial y final de estas.
2) El **estilo CSS**, que define la transición de la animación.

Hasta la aparición de este nuevo mecanismo, las técnicas de animación estaban basadas en scripts descritos en lenguaje **JavaScript**. Una de las **ventajas** fundamentales es que la animación **se muestra eficientemente en cualquier tipo de dispositivo**.

3.1 La regla @keyframes

La regla `@keyframes` especifica el código de animación. La podemos **aplicar**, con código CSS, a un elemento **html** o a **pseudoelementos** como por ejemplo, `hover`.

Si comparamos `@keyframes` con un lenguaje de programación, se podría decir que es como una **función** a la que podemos **invocar**. Esta se invoca con la propiedad `animation`, cuya sintaxis veremos en el apartado siguiente.

Su **sintaxis** es la siguiente:

```
@keyframes animationname {keyframes-selector {css-styles;}}
```

Un **ejemplo** sería el siguiente:

```
@keyframes cambia_color {
      0%, 70% { background: blue; }
      50% {
            background: red;
            height: 200px; }
}
```

Debemos ver a `@keyframes` como si fueran **animaciones en la línea del tiempo**. Por tanto, tenemos que indicar qué va a pasar progresivamente a lo largo del tiempo sobre el elemento html sobre el que se va a lanzar `@keyframes`. Esto se hace definiendo **porcentajes** con respecto al tiempo definido en la animación.

Los `@keyframes` van cambiando **progresivamente** las propiedades del estado en el que se encuentran al que tienen que ir llegando; por tanto, **son transitorias** (no cambian su estado de forma repentina).

A no ser que se indique lo contrario, **al 0% y al 100%**, la animación tiene las propiedades que tiene **por defecto** el elemento html.

Además de usar porcentajes, también podemos utilizar `from {}` y `to {}` para indicar el estado **inicial y final**, respectivamente. Aunque para una mejor compatibilidad con los navegadores web, se debería definir siempre que sea posible, los selectores 0% y 100%.

Un **ejemplo** sería el siguiente:

```
@keyframes cambia_color {
      from { background: blue; }
      to { background: yellow;}
}
```

3.2 La propiedad 'animation'

La propiedad `animation` presenta diversos parámetros que permiten configurar su funcionamiento. Con esta propiedad, solo se determinan las propiedades de la animación, pero la apariencia de esta se define a través de `@keyframes`. Es decir, la propiedad `animation` invoca a `@keyframes`, que es el que realmente ejecuta la animación. Los **parámetros** más habituales son:

- `animation-delay`. Define el tiempo que transcurre entre el momento de carga del elemento y el momento de inicio de la animación.
- `animation-direction`. Define la dirección de reproducción de la animación tras finalizar. Existen dos opciones: que retroceda hasta el fotograma de inicio al finalizar la secuencia, o que comience siempre desde el principio.
- `animation-duration`. Tal y como su nombre indica, este parámetro define el tiempo que la animación tarda en completar su ciclo de acción.
- `animation-iteration-count`. Define el número de veces que se repite la animación; para indicar que se reproduzca indefinidamente toma el valor `infinite`.

- `animation-play-state`. Al utilizar esta propiedad, es posible parar y reanudar la secuencia de la animación.
- `animation-fill-mode`. Define qué valores toman las propiedades de la animación tras su finalización.
- `animation-name`. Como se ha visto, las características de distribución y apariencia de la animación se definen mediante la regla en CSS `@keyframes`. Ahora bien, este atributo de la propiedad '`animation`' permite dar **nombre** a la regla a la animación.

Estos parámetros se pueden utilizar de forma independiente, o en una sola línea de código con el parámetro `animation`, cuya **sintaxis** es la siguiente:

```
animation: name duration timing-function delay iteration-count direction
           fill-mode play-state;
```

En el siguiente **ejemplo**, el elemento de texto **'p' se desplaza por la pantalla comenzando**, aproximadamente, **en la zona central** de la pantalla (200%) **y terminando en el extremo** izquierdo. A través del parámetro `animation-duration` se establece que la animación tendrá una duración de 6 segundos desde el inicio al fin. Mediante el parámetro `animation-name` se define el nombre de la regla `@keyframes` que describe la apariencia de los fotogramas de la animación.

Además, **en la mitad de la secuencia, se indica que el valor de la letra aumente hasta un 200% y, a continuación, que decrezca** hasta llegar al tamaño inicial. El regreso al valor original del tamaño de letra se produce cuando el elemento llega a la posición marcada por `margin-left`.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Ejemplo animacion</title>

<style>
    h1{
        animation-duration:  6s;
        animation-name: estiloAnimacion;
    }
    @keyframes estiloAnimacion {
        from {
            margin-left: 50%;
            width: 200%;
        }
        50% {

            font-size: 200%;
            margin-left: 10%;
            width: 150%;
        }
        to {
            margin-left: 0%;
            width: 100%;
        }

    }
</style>
</head>
<body>
    <h1>ESTO ES UNA ANIMACIÓN</h1>

</body>
</html>
```

3.3 La propiedad 'transition'

Esta propiedad ya se vio en la unidad didáctica 2, la cual permite **modificar el ritmo al que se producen las transiciones** entre los estados definidos para cada animación, y que aportan la **sensación de movimiento**.

Para crear un efecto de transición, se deben especificar **dos** cosas:

1) La **propiedad CSS** a la que se quiere añadir un efecto. **El efecto de la transición comenzará cuando la propiedad CSS especificada cambie de valor**.

2) La **duración** del efecto. Si no se especifica la duración, la transición no tendrá efecto, porque el valor por defecto es 0.

Para realizar las transiciones tenemos a nuestra disposición las siguientes **propiedades**:

- ***transition-property***. Indica a qué propiedades queremos que se le aplique la transición. Por ejemplo, *all*, *width*, *height*, etc. Su **sintaxis** es la siguiente:

```
transition-property: none | all | property | initial | inherit;
```

- ***transition-duration***. Indica la **duración de la transición**, es decir, cuánto tiempo queremos que tarde el navegador en reproducir el cambio entre las características iniciales a las características que le hemos indicado (no solo a características de *transform*, sino también a características como pueden ser el color).

```
transition-duration: time | initial | inherit;
```

- ***transition-timing-function.*** Especifica la **curva de velocidad** del efecto de transición. Permite que un efecto de transición **cambie de velocidad** a lo largo de su duración, por ejemplo, primero más rápido y luego más lento.

```
transition-timing-function: linear | ease | ease-in | ease-out |
ease-in-out | step-start | step-end | steps(int,start|end)| cubic-
bezier(n,n,n,n) | initial | inherit;
```

  - *linear*: misma velocidad de principio a fin.
  - *ease*: Valor por defecto. Especifica un inicio lento, luego rápido y un final lento.
  - *ease-in*: comienzo lento.
  - *ease-out:* final lento.
  - *ease-in-out:* comienzo y final lentos.
  - *steps(int,start|end)*: permite **controlar el valor de las transiciones de los diferentes segmentos** que componen una animación. Esta propiedad hace que sea posible trabajar con diferentes momentos de la animación, ampliando así el espectro de posibilidades de diseño del movimiento y la animación. El primer parámetro especifica el número de **intervalos** de la función, y debe ser un número entero positivo (mayor que 0). El segundo parámetro, que es opcional, tiene el valor "start" o "end", y especifica el punto

en el que se produce el cambio de valores dentro del intervalo. Si se omite el segundo parámetro, se le da el valor "end".

- *step-start:* equivalente a steps(1, start).
- *step-end:* equivalente a steps(1, end).
- *cubic-bezier(n,n,n,n):* define sus propios valores en la función cubic-bezier. Los valores posibles son valores numéricos de 0 a 1.

- ***transition-delay***. Especifica el **retardo** (en segundos) para el efecto de transición.

```
transition-delay: time | initial | inherit;
```

- ***transition.*** Establece las **cuatro propiedades de transición anteriores en una sola propiedad**.

```
transition: transition-property | transition-duration | transition-
timing-function | transition-delay | initial | inherit;
```

A continuación, se muestran un **ejemplo**:

```html
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition-property: all;
  transition-duration: 5s;
  transition-timing-function: ease-in-out;
  transition-delay: 2s;
}
div:hover {
  transform: rotate(240deg) skew(20deg);
  background: green;
  width: 300px;
}
</style>
</head>
<body>

<h1>La propiedad transition</h1>
<p>Al poner el ratón sobre el elemento, este rotará 240º
y se deformará 20º. Al mismo tiempo se pondrá verde y
aumentará su ancho a 300px:</p>
<div></div>

</body>
</html>
```

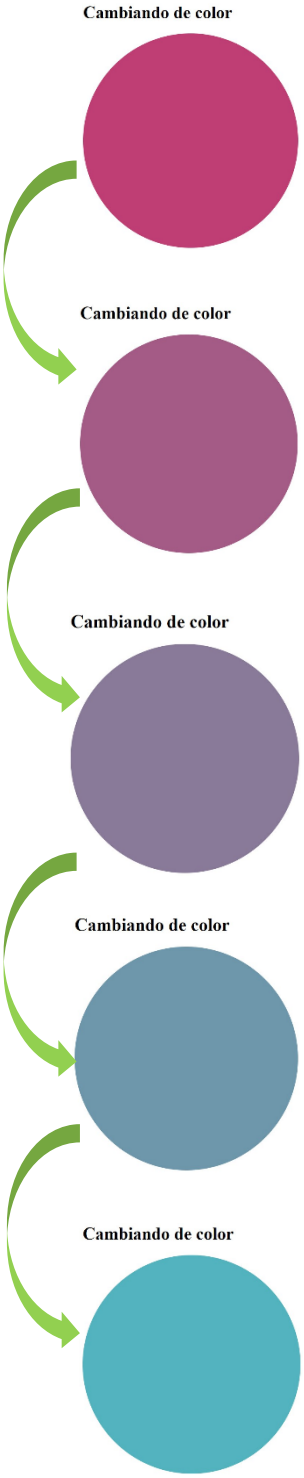
En el siguiente **ejemplo**, se representa un círculo que cambia de color cuando el cursor del ratón pasa por encima de este elemento. En la regla marcada mediante la *expresión* `section .circle {}` se definen los parámetros de diseño del círculo (color, tamaño, radio,

etc) y, a través de la propiedad ‘`transition`’, se indica que la animación va a modificar el color de fondo del círculo (`background`) en cuatro momento, es decir, en cuatro pasos (‘`steps(4)`’).

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Propiedad steps</title>

    <style>
        section .circle{
            background: rgba(191,63,116,1);
            width: 400px;
            height: 400px;
            border-radius: 200px;
            margin: 100 auto;
            display: block;
            transition: background 1s steps(4);
        }

        section .circle:hover{
            background: rgba(83, 180, 191, 1);
        }
    </style>
</head>
<body>
    <header><h1>Cambiando de color</h1></header>
    <section>
        <div class="circle"></div>
    </section>
</body>
</html>
```



3.4 Ejemplos de animaciones 2D

A continuación, se muestran algunos ejemplos sencillos de animaciones 2D.

3.4.1 La animación FLASH

La **animación flash** consiste en hacer parpadeos como si se tratara del flash de una cámara de fotos. En este ejemplo, la animación va a empezar cuando posemos el ratón sobre el elemento html en cuestión.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion Flash</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_flash.css">
</head>
<body>
    <div class="flash">
        <p>Flash</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 15px 2px;
    padding: 5px 40px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.flash:hover{
    animation: 0.7s flash 6;
}

@keyframes flash {
    50% {
        opacity: 1;
    }
    25%, 75% {
        opacity: 0;
    }

}
```

3.4.2 La animación CLICK

La **animación click** (o efecto click) reproduce el movimiento que tiene una caja o elemento html cuando se hace click sobre él. Esto es, se va para “adentro” y luego vuelve. Para ello tenemos que jugar con la escala.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion Click</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_click.css">
</head>
<body>
    <div class="click">
        <p>Click</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 15px 2px;
    padding: 5px 40px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.click:hover{
    animation: 0.7s click 1;
}

@keyframes click {
    25%, 75% {
        transform: scale(0.8);
    }
    50% {
        transform: scale(1.1);
    }

}
```

3.4.3 La animación CHICLE

El efecto chicle se consigue haciendo que la caja o elemento html se encoja y estire, como si de un chicle se tratara. Para ello vamos a tener que jugar con su escala en ambos ejes (horizontal y vertical).

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion chicle</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_chicle.css">
</head>
<body>
    <div class="chicle">
        <p>Chicle</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 40px;
    padding: 5px 40px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.chicle:hover{
    animation: 1s chicle 1;
}

@keyframes chicle {
    30% {
        transform: scaleX(1.25) scaleY(0.75);
    }
    50% {
        transform: scaleX(0.75) scaleY(1.25);
    }
    60% {
        transform: scaleX(1.15) scaleY(0.85);
    }

}
```

3.4.4 La animación VIBRADOR

La animación vibrador consiste en mover la caja o el elemento html hacia todos los lados o en el sentido que nosotros queramos. Para ello debemos usar la propiedad *transform: translateX* para movimientos horizontales, y *transform: translateY* para movimientos verticales.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion vibrador</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_vibrador.css">
</head>
<body>
    <div class="vibrador">
        <p>Vibrador</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 15px 2px;
    padding: 5px 40px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}
.vibrador:hover{
    animation: 1s vibrador infinite;
}
@keyframes vibrador {
    5%, 25%, 45%, 65%, 85% {
        transform: translateY(-10px);
    }
    15%, 35%, 55%, 75%, 95% {
        transform: translateY(10px);
    }
    10%, 30%, 50%, 70%, 90% {
        transform: translateX(-10px);
    }
    20%, 40%, 60%, 80% {
        transform: translateX(10px);
    }
}
```

3.4.5 La animación CARTEL

La animación cartel imita el movimiento que realizaría un cartel colgado en una puerta de una tienda. Por tanto, emula un vaivén que va perdiendo fuerza hasta que se queda en su posición original. Esto se consigue usando la propiedad `transform: rotate`, indicando que el origen de la rotación sea “arriba en el centro” o top center.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion cartel</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_cartel.css">
</head>
<body>
    <div class="cartel">
        <p>Cartel</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 15px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}
.cartel:hover{
    animation: 1s cartel 1;
    transform-origin: top center;
}
@keyframes cartel {
    20% {
        transform: rotate(15deg);
    }
    40% {
        transform: rotate(-10deg);
    }
    60% {
        transform: rotate(5deg);
    }
    80% {
        transform: rotate(-3deg);
    }
}
```

3.4.6 La animación BALANCEO

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion balanceo</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_balanceo.css">
</head>
<body>
    <div class="balanceo">
        <p>balanceo</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 15px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}
.balanceo:hover{
    animation: 1s balanceo 1;
}
@keyframes balanceo {
    20% {
        transform: scale(0.7) rotate(-6deg);
    }
    30%, 70% {
        transform: scale(1.1) rotate(6deg);
    }
    50%, 90% {
        transform: scale(1.1) rotate(-6deg);
    }
}
```

3.4.7 La animación VAIVÉN

Esta animación es muy similar al anterior, pero combinando diferentes propiedades. No sólo se desplaza horizontalmente, si no que se rota. Por tanto, se van a combinar las propiedades *transform: rotate* y *transform: translateX*.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion vaiven</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_vaiven.css">
</head>
<body>
    <div class="vaiven">
        <p>vaiven</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 15px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}
.vaiven:hover{
    animation: 1s vaiven 1;
}
@keyframes vaiven {
    15% {
        transform: rotate(-10deg) translateX(-25%);
    }
    30% {
        transform: rotate(5deg) translateX(20%);
    }
    45% {
        transform: rotate(-3deg) translateX(-10%);
    }
    60% {
        transform: rotate(2deg) translateX(5%);
    }
    75% {
        transform: rotate(-1deg) translateX(-3%);
    }
}
```

3.4.8 La animación CAÍDA o DESCOLGADO

Esta animación simula el **descuelgue** de una caja o elemento html, haciendo un movimiento de vaivén para simular que está cayendo, y finalmente se simula su caída con un desplazamiento vertical (o hacia abajo).

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Animacion caida</title>
    <link rel="stylesheet" type="text/css" href="estilos_caida.css">
</head>
<body>
    <div class="caida">
        <p>caida</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.caida{
    animation: 1s caida 1;
}

@keyframes caida {
    20%, 60% {
        transform: rotate(80deg);
    }
    40% {
        transform: rotate(60deg);
    }
    80% {
        opacity: 1;
        transform: rotate(60deg);
    }
    100% {
        transform: translateY(400px);
        opacity: 0;
    }

}
```

3.4.9 Animación ENTRADA y SALIDA ROTATIVOS

Con este efecto logramos la entrada y salida de un elemento de forma rotativa (rota sobre sí mismo). Para ello combinaremos las propiedades *transform: translateX* con *transform: rotate*.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Animacion IO Rotativo</title>
    <link rel="stylesheet" type="text/css" href="estilos_iorotativo.css">
</head>
<body>
    <div class="inRotativo">
        <p>inRotativo</p>
    </div>

    <div class="outRotativo">
        <p>outRotativo</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}
.inRotativo{
    animation: 1s inRotativo 1;
}
@keyframes inRotativo {
    0% {
        opacity: 0;
        transform: translateX(-100%) rotate(-260deg);
    }
}
.outRotativo{
    animation: 1s outRotativo 1;
}
@keyframes outRotativo {
    100% {
        opacity: 0;
        transform: translateX(100%) rotate(-260deg);
    }
}
```

3.4.10 Animación REBOTE

Esta animación trata de imitar el efecto producido cuando una pelota bota contra el suelo, se eleva y vuelve a bajar.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion Rebote</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_rebote.css">
</head>
<body>
    <div class="rebote">
        <p>rebote</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.rebote{
    animation: 1s rebote 1;
}

@keyframes rebote {
    20%, 50%, 80% {
        transform: translateY(0px);
    }
    40% {
        transform: translateY(-45px);
    }
    60% {
        transform: translateY(-25px);
    }
}
```

3.4.11 Animación REBOTEIN

Esta animación va a hacer el **efecto de aparición** de un elemento en escena, entrando como si fuera un rebote progresivo hacia el usuario. Para ello se empleará la propiedad `transform`: `scale`.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion ReboteIn</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_rebotein.css">
</head>
<body>
    <div class="rebotein">
        <p>reboteIn</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.rebotein{
    animation: 1s rebotein 1;
}

@keyframes rebotein {
    0% {
        opacity: 0;
        transform: scale(0.3);
    }
    50% {
        opacity: 1;
        transform: scale(1.1);
    }
    70% {
        transform: scale(0.9);
    }
}
```

3.4.12 Animación REBOTEOUT

Esta animación va a emular el efecto contrario a la animación ReboteIn, es decir, en vez de ser entrante, es saliente.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion ReboteOut</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_reboteout.css">
</head>
<body>
    <div class="reboteout">
        <p>reboteOut</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.reboteout{
    animation: 1s reboteout 1;
}

@keyframes reboteout {
    25% {
        transform: scale(0.9);
    }
    50% {
        opacity: 1;
        transform: scale(1.1);
    }
    100% {
        opacity: 0;
        transform: scale(0);
    }
}
```

3.4.13 Animación FADEIN

Este efecto consiste en emular la aparición progresiva de un elemento html.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion FadeIn</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_fadein.css">
</head>
<body>
    <div class="fadein">
        <p>FadeIn</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.fadein{
    animation: 10s fadein 1;
}

@keyframes fadein {
    0% {
        opacity: 0;
    }
    50% {
        opacity: 0.8;
    }
}
```

3.4.14 Animación VELOZIN

Esta animación emulará la sensación de que el elemento html tenga mucha velocidad en su entrada, afectando incluso a su apariencia.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion velozIn</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_velozin.css">
</head>
<body>
    <div class="velozin">
        <p>VelozIn</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 15px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.velozin{
    animation: 1s velozin 1;
}

@keyframes velozin {
    0% {
        opacity: 0;
        transform: translateX(-200%) skewX(30deg);
    }
    60% {
        opacity: 1;
        transform: translateX(-20%) skewX(-30deg);
    }
    80% {
        transform: translateX(0%) skewX(15deg);
    }
}
```

3.4.15 Animación VELOZOUT

Esta animación es similar a la anterior, sólo que la sensación de velocidad se produce a la salida del elemento html. En esta animación, el elemento sale sobre el eje vertical.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion velozOut</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_velozout.css">
</head>
<body>
    <div class="velozout">
        <p>VelozOut</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 15px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.velozout{
    animation: 1s velozout 1;
}

@keyframes velozout {
    15% {
        opacity: 1;
    }

    100% {
        transform: translateY(-500px) skewY(-70deg);
        opacity: 0;
    }
}
```

3.4.16 Animación GIRO

Esta animación hace que la aparición del elemento html sea rotando sobre sí mismo.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion Giro</title>
    <link rel="stylesheet" type="text/css" href="estilos_giro.css"
    >
</head>
<body>
    <div class="giro">
        <p>Giro</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 15px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.giro{
    transform-origin: center;
    animation: 5s giro 1;
}

@keyframes giro {
    0% {
        opacity: 0;
        transform: rotate(-200deg);
    }
}
```

3.4.17 Animación ARROW

En esta animación, cuando se pasa el ratón sobre el elemento html, aparece un pequeño triángulo por el lateral, se gira, y luego desaparece. Esta animación se utiliza para llamar la atención del usuario, con el objetivo de que haga click sobre él.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Animacion arrow</title>
    <link rel="stylesheet" type="text/css" href="estilos_arrow.css">
</head>
<body>
    <div class="arrow">
        <p>Arrow</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.arrow::before {
    transition-duration: 0.3s;
    content: '';
    /*Caracteristicas que forman el triangulo #a8cf45 */
    border-color: transparent #a8cf45 transparent transparent;
    border-style: solid;
    border-width: 15px 15px 15px 0;

    /*Posicionamiento del triángulo*/
    left: 5px;
    position: absolute;
    top: 50%;
    margin-top: -15px;
}
.arrow:hover::before {
    animation: 1.5s moverflecha infinite;
}

@keyframes moverflecha {
    25%{
        left: -15px;
        transform: rotate(0deg);
    }
    65%{
        left: -15px;
        transform: rotate(180deg);
    }
    90%{
        transform: rotate(180deg);
    }
}
```

3.4.18 Animación SOMBRA MÓVIL

Esta animación hace que el elemento html empiece a elevarse y a descender. A medida que se produce ese cambio de altura con respecto al suelo que se está simulando, la sombra se pronuncia más o se pronuncia menos. Este efecto se consigue combinando pseudoelementos con `@keyframes`.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1">
    <title>Animacion sombra movil</title>
    <link rel="stylesheet" type="text/css" href="
    estilos_sombra.css">
</head>
<body>
    <div class="sombra">
        <p>Sombra</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 15px 2px;
    padding: 5px 40px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}
.sombra {
    transform: translateX(0);
}
.sombra:hover{
    transform: translateY(-6px);
    animation: 1.5s movercaja infinite;
}
@keyframes movercaja {
    50%{
        transform: translateY(-3px);
    }
}
.sombra::before {
    position: absolute;
    content: '';
    top: 100%;
    left: 5%;
    height: 10px;
    width: 90%;
    opacity: 0;
    background: -webkit-radial-gradient(center, ellipse,
            rgba(0, 0, 0, 0.35) 0%, rgba(0, 0, 0, 0) 80%);
    transition-duration: 0.3s;
}
.sombra:hover::before {
    opacity: 0.4;
    transform: translateY(6px);
    animation: 1.5s moversombra infinite;
}
@keyframes moversombra {
    50%{
        opacity: 1;
        transform: translateY(3px);
    }
}
```

3.5 Ejemplos de animaciones 3D

A continuación, se muestran algunos ejemplos sencillos de animaciones 3D.

3.5.1 GIROX

Esta animación 3D trata de emular una caja o elemento html que rota sobra su eje horizontal. Este efecto se consigue usando las siguientes propiedades:

- *perspective ()*. Esta propiedad representa cuán lejos o cerca estamos de la caja o elemento html sobre el cual se reproduce el 3D. Si ponemos pocos píxeles de distancia, los efectos van a verse más pronunciados porque estamos mucho más cerca.
- *backface-visibility: visible*. Para que se vea la parte trasera del elemento html cuando gire.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Animacion GiroX</title>
    <link rel="stylesheet" type="text/css" href="estilos_girox.css">
</head>
<body>
    <div class="girox">
        <p>GiroX</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.girox{
    animation: 1.5s girox 1;
    backface-visibility: visible;
}
@keyframes girox {
    0% {
        opacity: 0.5;
        transform: perspective(200px) rotateX(150deg);
    }
    40%{
        transform: perspective(200px) rotateX(-20deg);
    }
    70%{
        transform: perspective(200px) rotateX(20deg);
    }
}
```

3.5.2 GIROY

Esta animación es igual a la anterior, con la diferencia de que el elemento html rota sobre su eje vertical.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Animacion GiroY</title>
    <link rel="stylesheet" type="text/css" href="estilos_giroy.css">
</head>
<body>
    <div class="giroy">
        <p>GiroY</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.giroy{
    animation: 1.5s giroy 1;
    backface-visibility: visible;
}
@keyframes giroy {
    0% {
        opacity: 0.5;
        transform: perspective(200px) rotateY(90deg);
    }
    40%{
        transform: perspective(200px) rotateY(-20deg);
    }
    70%{
        transform: perspective(200px) rotateY(20deg);
    }
}
```

3.5.3 GIRO OUT 3D

Esta animación hace que la caja o elemento html rote sobre el eje vertical y desaparezca. Esto se consigue combinando las propiedades *transform: perspective* y *transform: rotateY*. Si quisiéramos que rote sobre el eje horizontal, tendríamos que utilizar *transform: rotateX* en vez de *transform: rotateY*.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Animacion GiroOutY</title>
    <link rel="stylesheet" type="text/css" href="estilos_giroouty.css">
</head>
<body>
    <div class="giroouty">
        <p>GiroOutY</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}

.giroouty{
    animation: 1.5s giroouty 1;
}

@keyframes giroouty {
    100% {
        opacity: 0.3;
        transform: perspective(100px) rotateY(90deg);
    }
}
```

3.5.4 TRANSLATE Z

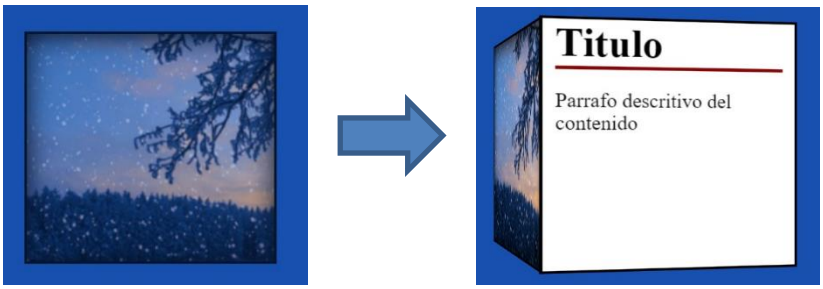
Esta animación consiste en girar la caja o elemento html, de manera que parezca que viene hacia el usuario y que se sale de la pantalla. Esto se hace añadiendo la propiedad *transform: translateZ* junto con *transform: perspective* y *transform: rotate*. En este caso, hay que tener cuidado ya que el resultado final depende del orden en el que se escriban estas propiedades.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Animacion Giro3D</title>
    <link rel="stylesheet" type="text/css" href="estilos_giro3d.css">
</head>
<body>
    <div class="giro3d">
        <p>Giro3D</p>
    </div>
</body>
</html>
```

```css
div {
    font-family: 'Arial';
    background-color: #a8cf45;
    margin: 50px 0px;
    position: relative;
    left: 20%;
    padding: 5px 20px;
    text-align: center;
    font-weight: bolder;
    width: 80px;
    display: inline-block;
    vertical-align: middle;
}
.giro3d{
    animation: 1.5s giro3d 1;
    backface-visibility: visible;
}
@keyframes giro3d {
    40% {
        transform: perspective(300px) translateZ(120px) rotateY(170deg);
    }
    50%{
        transform: perspective(300px) translateZ(120px) rotateY(190deg);
    }
    80%{
        transform: perspective(300px) translateZ(0px) rotateY(360deg);
    }
    100%{
        transform: perspective(300px) rotateY(360deg);
    }
}
```

3.5.5 CUBO

En esta animación vamos a tener un **cubo** que, al pasar el ratón por encima, **se gira y muestra otra de sus caras**. Por tanto, vamos a tener **dos caras**, la cara frontal que contendrá una imagen, y la cara trasera que contendrá la descripción (o texto) de lo que esa imagen representa.




```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Cubo 3D</title>
    <link rel="stylesheet" type="text/css" href="estilos_cubo.css">
</head>
<body>
    <div class="box">
        <div class="front"></div>
        <div class="back">
            <h1>Titulo</h1>
            <p>Parrafo descritivo del contenido</p>
        </div>
    </div>
</body>
</html>
```

```css
body {
    background: #1750AF;
}

.box {
    display: inline-block;
    margin: 50px 20px;
    width: 200px;
    height: 200px;
    position: relative;
}

.box > div {
    border: 2px solid black;
    position: absolute;
    width: 100%;
    height: 100%;
}

.front {
    opacity: 0.75;
    box-shadow: inset 0px 2px 20px black;
}
.back {
    background: white;
    transform: translateZ(-100px) rotateY(90deg);
    left: 100px;
}

.box {
    transition-duration: 0.5s;
    transform-style: preserve-3d;
    transform-origin: left center;
}
.box:hover {
    transform: perspective(800px) rotateY(-80deg);
}
.box:hover .front {
    opacity: 1;
}

.box:nth-child(1) .front {
    background: url('winter.gif');
}

.back h1 {
    margin: 0px;
    width: 90%;
    margin-left: 5%;
    border-bottom: 3px solid #870606;
}

.back p {
    width: 90%;
    margin-left: 5%;
}
```

3.5.6 BOLA DEL MUNDO

Esta animación simula una bola del mundo que está girando constantemente.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bola del mundo</title>
    <link rel="stylesheet" type="text/css" href="estilos_bola.css">
</head>
<body>
    <div class="content">
    </div>
</body>
</html>
```

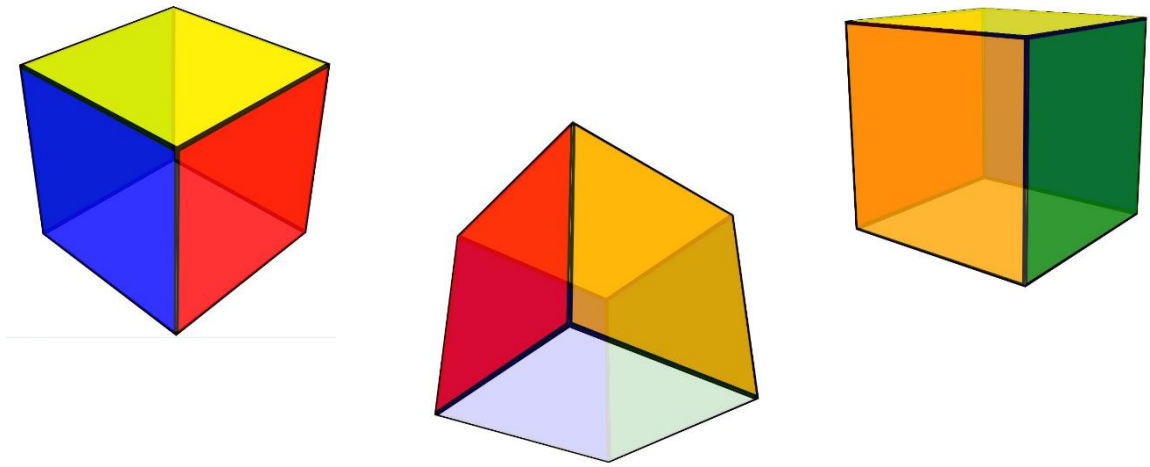
```css
body {
    background: #FFF;
}
.content {
    width: 92px;
    height: 92px;
    margin: 100px;
    background: url('mapa.jpg');
    background-size: cover;
    border-radius: 50%;
    animation: rotation 10s infinite linear;
    box-shadow: 0px 5px 15px #888;
}

@keyframes rotation {
    0% {
        background-position: 0px 0px;
    }
    100% {
        background-position: -160px 0px;
    }

}
```

3.5.7 CUBO RUBYK

Esta animación simula un cubo de colores, que estará continuamente girando sobre el eje horizontal y vertical, de manera que se verán todas sus caras.

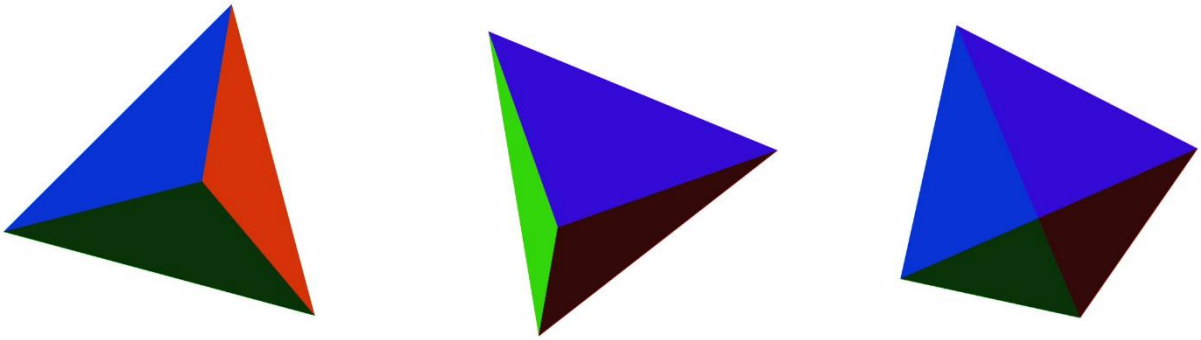


```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Cubo Rubyk</title>
    <link rel="stylesheet" type="text/css" href="estilos_cubo.css">
</head>
<body>
    <div class="contenedor">
        <div class="cubo">
            <div class="cara"></div>
            <div class="cara"></div>
            <div class="cara"></div>
            <div class="cara"></div>
            <div class="cara"></div>
            <div class="cara"></div>
        </div>
    </div>
</body>
</html>
```

```css
.contenedor {
    perspective: 1000px;
    perspective-origin: 50% 50%;
}
.cubo {
    transform-style: preserve-3d;
    transform: rotateY(-30deg);
    margin: 200px auto;
}
.cubo, .cubo > div {
    height: 290px;
    width: 290px;
}
.cara {
    position: absolute;
    border: 5px solid black;
    opacity: 0.8;
}
.cubo .cara:nth-child(1) {
    background: red;
    transform: translateZ(150px);
}
.cubo .cara:nth-child(2){
    background: green;
    transform: translateZ(-150px);
}
.cubo .cara:nth-child(3){
    background: blue;
    transform: translateX(-150px) rotateY(-90deg);
}
.cubo .cara:nth-child(5){
    background: yellow;
    transform: translateY(-150px) rotateX(-90deg);
}
.cubo .cara:nth-child(6){
    background: white;
    transform: translateY(150px) rotateX(-90deg);
}
.cubo {
    animation: giro 20s infinite linear;
}
@keyframes giro {
    0% {
        transform: rotateX(-45deg) rotateY(0deg);
    }
    50% {
        transform: rotateX(45deg) rotateY(360deg);
    }
    100% {
        transform: rotateX(-45deg) rotateY(720deg);
    }
}
```

3.5.8 PIRÁMIDE 3D

Esta animación simula una pirámide de colores, que estará continuamente girando sobre el eje horizontal y vertical, de manera que se verán todas sus caras.



```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Piramide 3D</title>
    <link rel="stylesheet" type="text/css" href="estilos_piramide.css">
</head>
<body>
    <div id="piramide">
        <div></div>
        <div></div>
        <div></div>
        <div></div>
    </div>
</body>
</html>
```

```css
#piramide {
    position: relative;
    margin: 80px auto;
    height: 500px;
    width: 100px;
    transform-style: preserve-3d;
    transform-origin: 116px 200px 116px;
    transform: rotateY(30deg);
    animation: giro 10s infinite linear;
}
#piramide > div {
    position: absolute;
    border-style: solid;
    border-width: 200px 0px 200px 346px;
    transform-origin: 0 0;
}
#piramide > div:nth-child(1) {
    border-color: transparent transparent transparent
                  rgba(0, 0, 0, 0.8);

    transform: rotateX(-60deg)
               rotateY(19.5deg)
               translateX(-116px)
               translateY(-200px)
               translateZ(326px);
}
#piramide > div:nth-child(2) {
    border-color: transparent transparent transparent
                  rgba(255, 0, 0, 0.8);

    transform: rotateY(-19.5deg)
               rotateX(180deg)
               translateY(-400px);
}
#piramide > div:nth-child(3) {
    border-color: transparent transparent transparent
                  rgba(0, 255, 0, 0.8);

    transform: rotateY(90deg)
               rotateZ(60deg)
               rotateX(180deg)
               translateY(-400px);
}
#piramide > div:nth-child(4) {
    border-color: transparent transparent transparent
                  rgba(0, 0, 0, 255, 0.8);

    transform: rotateX(60deg)
               rotateY(19.5deg);
}

@keyframes giro {
    from {
        transform: rotateX(0deg) rotateY(0deg) rotateZ(0deg);
    }
    to {
        transform: rotateX(360deg) rotateY(360deg) rotateZ(360deg);
    }
}
```

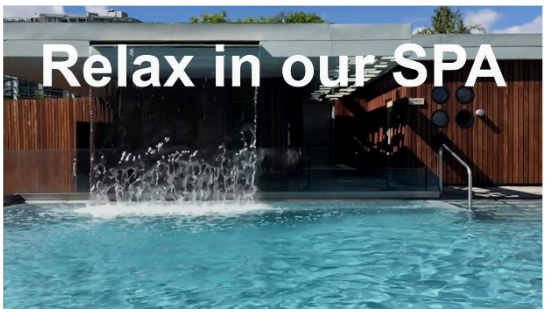
3.6 Integración de vídeo en una animación

Para integrar vídeo en una animación, simplemente tendremos que utilizar la etiqueta `<video>` en nuestro código html.

Existen bancos de **vídeos** en Internet que podemos usar, tales como https://pixabay.com/, https://www.pexels.com/, …

En el siguiente ejemplo se ha emulado la aparición progresiva de las letras (efecto **FadeIn**) sobre un vídeo de fondo.

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Animacion FadeIn con Video</title>
    <link rel="stylesheet" type="text/css" href="estilos_fadein.css">
</head>
<body>
    <h1 class="fadein">Relax in our SPA</h1>
    <video class="videosize" src="Waterfall.mp4" autoplay loop muted>
</body>
</html>
```




```css
h1 {
    font-family: 'Arial';
    color: #FFF;
    margin: 50px 0px;
    position: relative;
    text-align: center;
    font-weight: bolder;
    font-size: 110px;
    vertical-align: middle;
}
.videosize {
    position:absolute;
    z-index:-1;
    top:0;
    left:0;
    width:100%;
    height:100vh;
}
.fadein{
    animation: 15s fadein 1;
}
@keyframes fadein {
    0% {
        opacity: 0;
    }
    50% {
        opacity: 0.8;
    }
}
```

3.7 Integración de sonido en una animación

Para integrar sonido en una animación y que se corresponda con el movimiento de nuestra animación ya es algo **más complejo**. Para ello tendremos que utilizar **JavaScript**, el cual nos ayudará a controlar el sonido, por ejemplo, para iniciar el sonido cuando el usuario haga click en el botón de play, o pararlo si el usuario hace click en el botón de pausa.

Por otro lado, debemos tener en cuenta la **duración del sonido**, ya que esta deberá ser la misma que la de la animación para que ésta sea coherente.

Existen multitud **bancos de sonidos o efectos** de uso libre en Internet que podemos utilizar, tales como https://freesound.org/, https://www.pexels.com/, https://pixabay.com/...

A continuación, se presentan algunos **ejemplos** de código de animaciones con sonido.

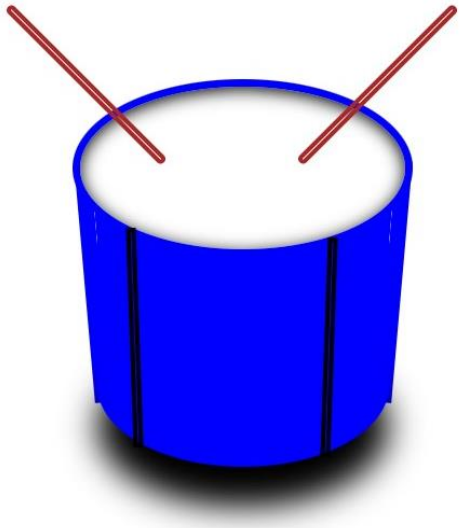
3.7.1 Animación de un TAMBOR

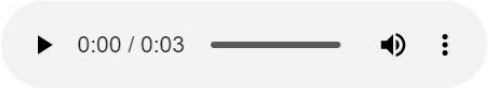
Esta animación simula que se está tocando un tambor, con su correspondiente sonido.

```html
<!DOCTYPE html>
<html>
<head>
    <style>
        #drum{
            width:50%;
            margin:auto;
            position:relative;
            margin-top:100px;
        }
        audio{
            position:relative;
            left:50%;
            transform:translate(-50%);
        }
        h2{
            text-align:center;
            margin:50px
        }
        .top{
            border:5px solid blue;
            width:200px;
            margin:auto;
            height:100px;
            border-radius:100%;
            background:white;
            box-shadow:0px 2px 10px black inset;
            position:relative
        }
        .mid{
            width:180px;
            background:blue;
            height:150px;
            margin:auto;
            position:relative;
            top:-60px;
            z-index:-1;
        }
        .bottom{
            width:180px;
            margin:auto;
            height:100px;
            border-radius:100%;
            background:blue;
            position:relative;
            top:-120px;
            box-shadow:0px 25px 20px black;
        }
        .mid::before{
            content:"";
            position:absolute;
            left:-8px;
            height:130px;
            border:10px solid blue;
            transform:rotate(-5deg);
        }
        .mid::after{
            content:"";
            position:absolute;
            right:-8px;
            height:130px;
            border:10px solid blue;
            transform:rotate(5deg);
        }
        .bar{
            height:130px;
            border:3px solid;
            width:0px;
            display:inline-block;
            position:relative;
        }
        .b1{
            left:20px;
            transform:rotate(-2deg);
            top:-48px;
        }
        .b2{
            left:130px;
            transform:rotate(2deg);
            top:-42px;
        }
```

```css
.stick{
    height:130px;
    border:3px solid brown;
    width:0px;
    display:inline-block;
    position:absolute;
    border-radius:50px;
    top:-70px;
}
.s1{
    transform:rotate(-45deg);
}
.s2{
    left:180px;
    transform:rotate(45deg);
}
.active1{
    animation : tap1 2.6s ;
}
.active2{
    animation : tap2 2.6s ;
}
@keyframes tap1{
    9%{
        transform:rotate(-45deg);
        height:130px;
    }
    24.6%{
        transform:rotate(-45deg);
        height:130px;
    }
    39%{
        transform:rotate(-45deg);
        height:130px;
    }
    54%{
        transform:rotate(-45deg);
        height:130px;
    }
    68%{
        transform:rotate(-45deg);
        height:130px;
    }
    87.7%{
        transform:rotate(-45deg);
        height:130px;
    }
    0%{
        transform:rotate(-60deg);
        height:90px;
    }
    17.3%{
        transform:rotate(-60deg);
        height:90px;
    }
    31.5%{
        transform:rotate(-60deg);
        height:90px;
    }
    47%{
        transform:rotate(-60deg);
        height:90px;
    }
    62%{
        transform:rotate(-60deg);
        height:90px;
    }
    78.4%{
        transform:rotate(-60deg);
        height:90px;
    }
    95%{
        transform:rotate(-60deg);
        height:90px;
    }
}
@keyframes tap2{
    0%{
        transform:rotate(45deg);
        height:130px;
    }
    17.3%{
        transform:rotate(45deg);
        height:130px;
    }
    31.5%{
        transform:rotate(45deg);
        height:130px;
    }
```

```
            47%{
                transform:rotate(45deg);
                height:130px;
            }
            62%{
                transform:rotate(45deg);
                height:130px;
            }
            78.4%{
                transform:rotate(45deg);
                height:130px;
            }
            95%{
                transform:rotate(45deg);
                height:130px;
            }
            9.9%{
                transform:rotate(60deg);
                height:90px;
            }
            24.6%{
                transform:rotate(60deg);
                height:90px;
            }
            39%{
                transform:rotate(60deg);
                height:90px;
            }
            54%{
                transform:rotate(60deg);
                height:90px;
            }
            68%{
                transform:rotate(60deg);
                height:90px;
            }
            87.7%{
                transform:rotate(60deg);
                height:90px;
            }
        }
    </style>
</head>
<body>
    <h1>TAMBOR</h1>
    <div id='drum'>
        <div class="top">
            <div class="stick s1"></div>
            <div class="stick s2"></div>
        </div>
        <div class="mid"></div>
        <div class="bottom">
            <div class="bar b1"></div>
            <div class="bar b2"></div>
        </div>
        <audio src="Bongos.wav" controls>
        </audio>
    <script>
        var audio = document.getElementsByTagName('audio')[0];
        audio.onplay=()=>{
            document.getElementsByClassName('stick')[0].classList.add('active1');
            document.getElementsByClassName('stick')[1].classList.add('active2');
            console.log("hi")
        }
        audio.onpause=()=>{
            document.getElementsByClassName('stick')[0].classList.remove('active1');
            document.getElementsByClassName('stick')[1].classList.remove('active2');
        }
    </script>
</body>
</html>
```





3.7.2 Animación PASO A NIVEL

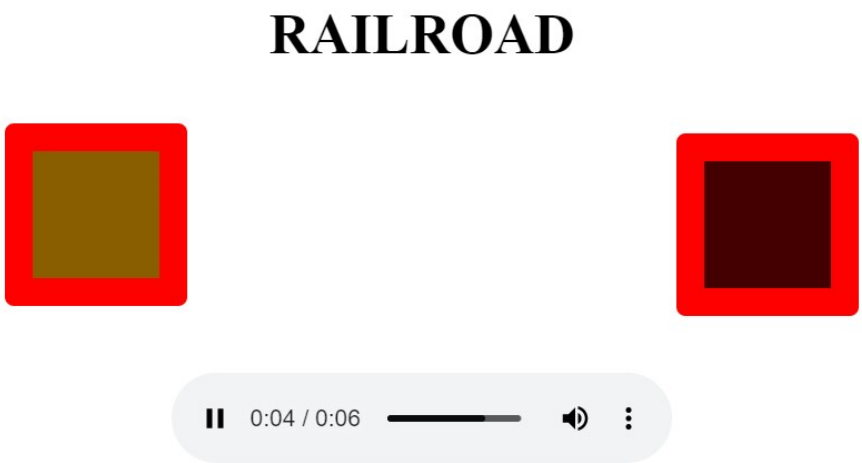
Esta animación simula las **luces que hay en un paso a nivel** en una vía de tren, con su correspondiente sonido.

```html
<!DOCTYPE html>
<html>
    <style>
        .signal {
            font-size: 1.8em;
            height: 400px;
            margin: 2em auto 1em;
            position: relative;
            text-align: center;
            width: 500px;
        }
        .lights,
        .lights::after {
            background-color: #400000;
            border: 0.6em solid #f00;
            border-radius: 5%;
            height: 2.6em;
            left: 0;
            position: relative;
            width: 2.6em;
        }
        .lights::after {
            content: "";
            left: 515%;
            right: 0;
            top: -0.4em;
            width: 2.6em;
            height: 2.6em;
            position: absolute;
        }
        .signal.active .lights {
            animation: r1 6.38s linear 0s 1;
        }
        .signal.active .lights::after {
            animation: r2 6.38s linear 0s 1;
        }

        @keyframes r1 {
            0.0% {
            background-color: #400;
            }
            1.4% {
            background-color: #400;
            }
            1.5% {
            background-color: #FF0;
            }
            8.8% {
            background-color: #400;
            }
            16.3% {
            background-color: #400;
            }
            16.4% {
            background-color: #FF0;
            }
            23.7% {
            background-color: #400;
            }
            31.7% {
            background-color: #400;
            }
            31.8% {
            background-color: #FF0;
            }
            39.0% {
            background-color: #400;
            }
            46.9% {
            background-color: #400;
            }
            47.0% {
            background-color: #FF0;
            }
            54.4% {
            background-color: #400;
            }
            62.1% {
            background-color: #400;
            }
            62.2% {
            background-color: #FF0;
            }
            69.7% {
            background-color: #400;
            }
            77.5% {
            background-color: #400;
            }
            77.6% {
            background-color: #FF0;
            }
            85.0% {
            background-color: #400;
            }
            92.4% {
            background-color: #400;
            }
            92.5% {
            background-color: #FF0;
            }
            100.0% {
            background-color: #400;
            }
        }
        @keyframes r2 {
            0.0% {
            background-color: #400;
            }
            8.8% {
            background-color: #400;
            }
            8.9% {
            background-color: #FF0;
            }
            23.7% {
            background-color: #400;
            }
            23.8% {
            background-color: #FF0;
            }
            31.7% {
            background-color: #400;
            }
            39.0% {
            background-color: #400;
            }
            39.1% {
            background-color: #FF0;
            }
            46.9% {
            background-color: #400;
            }
            54.4% {
            background-color: #400;
            }
            54.5% {
            background-color: #FF0;
            }
            62.1% {
            background-color: #400;
            }
            69.7% {
            background-color: #400;
            }
            69.8% {
            background-color: #FF0;
            }
            77.5% {
            background-color: #400;
            }
            85.0% {
            background-color: #400;
            }
            85.1% {
            background-color: #FF0;
            }
            92.4% {
            background-color: #400;
            }
            100.0% {
            background-color: #400;
            }
        }
```

```
        audio{
            margin:40px;
        }
    </style>
    <body>
        <div id="signal" class="signal">
        <h3>RAILROAD</h3>
        <div class="lights"></div>
        <audio src="railroad-crossing-signal.wav" controls preload="auto">
                Your browser does not support the
                <code>audio</code> element.
            </audio>
        </div>

        <script>
            var audio = document.getElementsByTagName('audio')[0];
            audio.onplay=()=>{
                if (!signal.classList.contains('active')) {
                    signal.classList.add('active');
                }
            }
            audio.onpause=()=>{
                signal.classList.remove('active');
            }
        </script>
    </body>
</html>
```



3.7.3 Animación LATIDO DE CORAZÓN

Esta animación simula el latido de un corazón, con su correspondiente sonido.

CÓDIGO HTML y JS

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" type="text/css" href="heartbeat_estilos.css">
  <title>HEARTBEAT</title>
</head>
<body>

<h2>HEARTBEAT</h2>

<div id="heart" class="heart">
  <audio id="heartbeat" src="heartbeat.mp3" preload="auto">
    Your browser does not support the <code>audio</code> element.
  </audio>
</div>

<div class="buttons">
  <button id="play">Play</button>
  <button id="reset">Reset</button>
</div>

<script>
  var playBtn = document.getElementById('play'),
  resetBtn = document.getElementById('reset'),
  heart = document.getElementById('heart'),
  hearbeat = document.getElementById('heartbeat');

playBtn.addEventListener('click', function() {
  if (!heart.classList.contains('active')) {
    heart.classList.add('active');
    heartbeat.play();
  }
}, false);

resetBtn.addEventListener('click', function() {
  heart.classList.remove('active');
  heartbeat.pause();
  heartbeat.currentTime = 0;
}, false);

</script>


</body>
</html>
```

CÓDIGO CSS

```
body {
  background-color: black;
  color: #eaeaea;
  margin: 0;
  padding: 1em;
  text-align: center;
}

.heart {
  background-color: black;
  height: 200px;
  margin: 1em auto;
  position: relative;
  text-align: center;
  width: 200px;
}

.heart::before,
.heart::after {
  background-color: red;
  box-shadow: -8px -14px 10px 0 black inset;
  content: "";
  height: 80%;
  left: 25%;
  position: absolute;
  top: 2%;
  width: 50%;
  border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  transform: translateZ(0) rotate(-45deg);
  transform-origin: 50% 68.75% 0;
}
.heart::after {
  transform: translateZ(0) rotate(45deg);
  box-shadow: -15px 10px 14px 0 black inset;
}

.heart.active::before {
  animation: heartbeats-before 5.85s linear 0s 1;
}

.heart.active::after {
  animation: heartbeats-after 5.85s linear 0s 1;
}

@keyframes heartbeats-before {
  3% {
    background-color: red;
    transform: rotate(-45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  8.2% {
    background-color: orange;
    transform: rotate(-45deg) scale(0.8);
    border-radius: 70% 70% 70% 0 / 70% 70% 10% 0;
  }
  13.7% {
    background-color: red;
    transform: rotate(-45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  25% {
    background-color: red;
    transform: rotate(-45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  30% {
    background-color: orange;
    transform: rotate(-45deg) scale(0.8);
    border-radius: 70% 70% 70% 0 / 70% 70% 10% 0;
  }
  36% {
    background-color: red;
    transform: rotate(-45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  44.4% {
    background-color: red;
    transform: rotate(-45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  49.6% {
    background-color: orange;
    transform: rotate(-45deg) scale(0.8);
    border-radius: 70% 70% 70% 0 / 70% 70% 10% 0;
  }
  56.4% {
    background-color: red;
    transform: rotate(-45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  66.7% {
    background-color: red;
    transform: rotate(-45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  71.8% {
    background-color: orange;
    transform: rotate(-45deg) scale(0.8);
    border-radius: 70% 70% 70% 0 / 70% 70% 10% 0;
  }
  77% {
    background-color: red;
    transform: rotate(-45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
}

@keyframes heartbeats-after {
  3% {
    background-color: red;
    transform: rotate(45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  8.2% {
    background-color: orange;
    transform: rotate(45deg) scale(0.8);
    border-radius: 70% 70% 0 70% / 70% 70% 0 10%;
  }
  13.7% {
    background-color: red;
    transform: rotate(45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  25% {
    background-color: red;
    transform: rotate(45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  30% {
    background-color: orange;
    transform: rotate(45deg) scale(0.8);
    border-radius: 70% 70% 0 70% / 70% 70% 0 10%;
  }
  36% {
    background-color: red;
    transform: rotate(45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  44.4% {
    background-color: red;
    transform: rotate(45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  49.6% {
    background-color: orange;
    transform: rotate(45deg) scale(0.8);
    border-radius: 70% 70% 0 70% / 70% 70% 0 10%;
  }
  56.4% {
    background-color: red;
    transform: rotate(45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  66.7% {
    background-color: red;
    transform: rotate(45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
  71.8% {
    background-color: orange;
    transform: rotate(45deg) scale(0.8);
    border-radius: 70% 70% 0 70% / 70% 70% 0 10%;
  }
  77% {
    background-color: red;
    transform: rotate(45deg) scale(1);
    border-radius: 50% 50% 0 0 / 37.5% 37.5% 0 0;
  }
}

button {
  color: black;
  padding: 4px 16px;
}
.p {
  text-align: center;
  font-size: 13px;
  padding-top: 120px;
}

.p a {
  color: lightblue;
}
```