ANIMACIONES Y TRANSICIONES

1. INTRODUCCIÓN

Uno de los aspectos más vistosos y atractivos de un sitio web es la inclusión de animaciones.

Una animación es un *proceso utilizado para dar la sensación de movimiento a imágenes o dibujos*. Para el caso concreto del diseño web, se puede completar esta definición añadiendo, además de imágenes, textos y audios que pueden visualizarse directamente desde un navegador, dando todo ello sensación de movimiento (animando la web).

El objetivo es conocer la manera de incluir y generar animaciones para un sitio web, mostrando la tecnología y las tendencias actuales en este campo.

Actualmente la web ofrece muchas alternativas para la creación de estas animaciones.

- Algunas de esas alternativas **son de pago** y **otras son libres.**
- Algunas **son implementadas por todos los navegadores**, otros son exclusivas de unos pocos.
- Algunas siguen el **estándar W3C y** otras no.

La variedad es mucha, y el desarrollador tiene que elegir en cada momento la solución que mejor se adapte a sus necesidades, a la tecnología empleada en el desarrollo del sitio web y a la tecnología que permiten los navegadores.

2. FORMATOS QUE SOPORTAN ANIMACIONES

La forma más sencilla de incluir animaciones es mediante archivos con formatos que permiten movimiento. En este apartado se explicarán los más comunes.

2.1. GIFS ANIMADOS

Un *gif* animado es un formato creado en 1987. Consiste, **en una serie de imágenes**, en formato *gif* que están colocados consecutivamente y **se muestran en pantalla durante un intervalo de tiempo** determinado. Esta secuencia se suele hacer en modo bucle (*loop*) para se repita indefinidamente.

Las **ventajas** de un *gif* animado son:

- Su rápida descarga.
- Nitidez.
- Permite usar transparencias.
- Se puede insertar en HTML como cualquier otra imagen.

Pero tiene como **desventajas:**

- Que las imágenes deben tener un número fijo de colores (un máximo de 256).
- Al tratarse de un formato de mapa de bits si la animación tiene muchas imágenes estáticas el tamaño del fichero resultante puede ser excesivo para que sea práctico.

En la red existen muchos sitios web que ofrecen *gifs* animados ya creados. Y también hay herramientas para crearlos y convertirlos a otros formatos *(.avi, .mpeg, .swf,* etc.). Algunas son las siguientes:

- **GIF Construction Set Professional**: Es una herramienta de escritorio muy utilizada por su gran funcionalidad. Permite crear *gifs* y conversión entre diferentes formatos.
- **GIMP**: Una herramienta de diseño gráfico Open Source (en el segmento de Adobe Photoshop) que permite, además de editar todo tipo de imágenes, crear *gif* animados poniendo capa imagen estática en una capa diferente.
- **Picasion**: Es una herramienta online que permite crear *gif* animados a partir de un máximo de 10 imágenes. Su funcionalidad es muy limitada, pero permite poner la velocidad de transmisión y la calidad final. Además, las imágenes pueden ser cogidas de repositorios de fotos como *Flickr.com*.

2.2. FLASH

Los ***archivos Flash .swf*** han sido el formato más atractivo de la web durante más de una década, tanto es que muchas webs han sido creadas con solo animaciones Flash (swf). Los archivos flash (*swf*) son creados, inicialmente, con la herramienta Adobe Flash (antes Macromedia Flash), aunque existen otras herramientas que también pueden crearlos, pero con menos funcionalidad.

La principal **ventaja de *.swf*** *es* que es un formato vectorial (ocupa poco espacio) e integra imagen, texto, vídeo *(Flash video) y* audio. Además, incluye un lenguaje programación *ActionScript* que hace que estas animaciones puedan interactuar, al reconocer, por ejemplo, eventos de ratón).

Actualmente, el uso de este formato en la web está viéndose **reducido** por dos razones principales:

- La **dificultad de que el contenido de los *.swf* puedan ser indexados** en su totalidad **por los motores de búsqueda**. Esto, desde el punto de vista SEO (*Search Engine Optimization.* Posicionamiento de buscadores) es una limitación importante.
- Por otro lado, desde **el auge del iPad** el uso de swf en páginas web se ha reducido considerablemente. El motivo es que, hasta la fecha, **Apple se niega** a introducir tecnología Flash en sus dispositivos (iPhone, iPad y iPod). Esto hace que los navegadores de estos dispositivos (Safari, por ejemplo) no puedan visualizar archivos *.swf,* con el

consiguiente quebradero de cabeza para los diseñadores. La solución adoptada ante este problema de intereses empresariales ha sido la aparición de HTML5 y *conversores* de *swf a* HTML5.

Actualmente hay navegadores que ya han dejado o tienen previsto no dar más soporte al formato Flash próximamente.

3. ANIMACIONES CON HTML5 Y CSS3

El lenguaje HTML5, combinado con CSS3 y JavaScript, permite crear animaciones tan potentes como Flash.

HTML5 tiene como ventaja respecto a Flash que es casi soportado por los navegadores actuales de forma nativa y que la web tiene en un futuro a ser implementado completamente en todos. Indudablemente dejará de lado a Flash en un futuro no muy lejano, facilitando a los desarrolladores la creación de webs animadas para cualquier navegador.

El uso de HTML5, CSS3 y JavaScript tiene una estructura muy parecida a lo mostrado en el punto anterior. Cualquier desarrollador con conocimientos de estos lenguajes puede escribir directamente el código para animar sus creaciones. Además, si se apoya en la multitud de páginas web disponibles en Internet con trucos para hacer animaciones o en librerías de alto nivel (como jQuery), el resultado es aún más rápido.

Para crear una secuencia de animación CSS, se emplea la propiedad **animation** y sus subpropiedades. Con ellas podemos no solo configurar el ritmo y la duración de la animación, sino otros detalles sobre la secuencia de la animación. Con ellas no configuramos la apariencia actual de la animación. Para ello disponemos de **@keyframes** como se describirá más adelante.

Las subpropiedades de animation son:

- **animation-name:** Especifica el nombre de la regla @keyframes. Ejemplo: `animation-name: animacion1`
- **animation-duration:** Cuánto tiempo tarda en ejecutarse la animación. Se expresa en segundos. Ejemplo: `animation-duration: 2s`
- **animation-timing-function:** Indica el ritmo de la animación. Es decir, cómo se muestran los fotogramas de la animación, estableciendo curvas de aceleración. A continuación, se explican algunos de los valores posibles:
  - ✓ `animation-timing-function: ease`: Define un efecto de animación con un comienzo lento, luego rápido y finalmente termina lento (cuando no definimos la función elige esta por defecto)
  - ✓ `animation-timing-function: linear`: Define un efecto de animación con la misma velocidad de inicio a fin.
  - ✓ `animation-timing-function: ease-in`: Define un efecto de animación con un comienzo lento.
  - ✓ `animation-timing-function: ease-out`: Define un efecto de animación con un final lento.
  - ✓ `animation-timing-function: ease-in-out`: Define un efecto de animación con un comienzo lento y un final lento.

  - ✓ `animation-timing-function: cubic-bezier (n,n,n,n):` Definimos la velocidad de la animación en base a una curva cúbica de Bézier. Para ello incorporamos cuatro valores "n" que serán números decimales entre el 0 y el 1.
  - ✓ `animation-timing-function: step-start:` Salto de la situación inicial a la final al comienzo. Equivalente a steps(1, start).
  - ✓ `animation-timing-function: step-end:` Salto de la situación inicial al último estado al final. Equivalente a steps(1, end).
  - ✓ `animation-timing-function: steps (numeroDePasos, end):` El parámetro numeroDePasos (debe ser mayor de cero) establece el número de pasos intermedios entre la situación inicial y la final, lo que equivale a que la animación se vea como “pequeños saltos”.
- **animation-delay:** Es el tiempo de retardo en el comienzo de la animación. Ejemplo: `animation-delay: 2s.`
- **animation-iteration-count:** El número de iteraciones de la animación. Puede ser un valor entero mayor que cero o `infinite`, que repite la animación mientras el navegador siga abierto o el usuario la interrumpa. Ejemplos: `animation-iteration-count: 1` o `animation-iteration-count: infinite`
- **animation-direction:** Indica si la animación debe retroceder hasta el fotograma de inicio cuando se finalice la secuencia, o si debe comenzar desde el principio al llegar al final. Ejemplos posibles valores:
  - ✓ `animation-direction: normal:` Cada vez que termina un ciclo, la animación se reinicia al estado inicial y comienza desde el principio. Este es el comportamiento por defecto.
  - ✓ `animation-direction: alternate:` La animación, al terminar un ciclo, invierte su dirección. Es decir, los pasos de la animación se ejecutan al revés. En definitiva, en cada ciclo se ejecuta la animación en una dirección, empezando por el comportamiento por defecto.
  - ✓ `animation-direction: reverse:` Cada ciclo de la animación se reproduce al revés. Cada vez que comienza un ciclo de animación, ésta se posiciona en el estado final y comienza desde ahí.
  - ✓ `animation-direction: alternate-reverse:` Es similar a `alternate` pero la animación se reproduce al revés. Es decir, la animación se posiciona en el estado final, comienza a reproducirse al revés y cuando llega al inicio vuelve a reproducirse de forma normal hasta llegar al final de la secuencia. Y vuelve otra vez a repetirse. En definitiva, es como `alternate`, pero en este caso se comienza la reproducción al revés.
- **animation-fill-mode:** Especifica el modo en que una animación CSS aplica sus estilos, estableciendo su persistencia y estado final tras su ejecución. Ejemplos de posibles valores:
  - ✓ `animation-fill-mode: none:` Es el valor por defecto. La animación no aplicará los estilos antes ni después de su ejecución.
  - ✓ `animation-fill-mode: forwards:` El objeto sobre el que se aplica la animación quedará con los valores y estilos que le aplique el último keyframe de la ejecución de la animación. El último valor dependerá del valor de `animation-direction` y `animation-iteration-count`.
  - ✓ `animation-fill-mode: backwards:` La animación aplicará los valores definidos en el primer keyframe tan pronto como se aplique al objeto, y los

retendrá durante el tiempo de `animation-delay`. El primer keyframe dependerá del valor de `animation-direction`.

- ✓ `animation-fill-mode: both:` La animación seguirá las reglas de las opciones forwards y backwards, extendiendo las propiedades de la animación en ambas direcciones.

- **animation-play-state:** Determina si una animación está en ejecución o en pausa. Puede ser consultada para determinar si la animación se está ejecutando. Además, su valor se puede establecer para pausar y reanudar una animación. Ejemplos: `animation-play-state: running` si se está ejecutando o `animation-play-state: paused` si la animación está pausada. Se pueden cambiar estos valores mediante clases de CSS como hover o vía JavaScript.

Las propiedades se animation se pueden especificar de forma abreviada. Por ejemplo: `animation: mymove 5s infinite` (respectivamente name, duration e iteration-count).

Ya hemos visto cómo configurar la animación, pero falta conocer la apariencia de la misma. Decíamos anteriormente que esto se hace mediante la regla **@keyframes**.

La regla arroba `@keyframes` permite a los autores controlar los pasos intermedios en una secuencia de animación CSS mediante el establecimiento de determinados puntos que se deben alcanzar durante la animación. Esto le da un control más específico sobre los pasos intermedios de la secuencia de animación que se obtiene al dejar que el navegador maneje todo automáticamente.

Para utilizar keyframes, se crea una regla de `@keyframes` con un nombre que es utilizada por la propiedad `animation-name` para que coincida con una animación de keyframe a su lista. Cada regla `@keyframes` contiene una lista de estilo de selectores de keyframe, cada una de los cuales está compuesto de un porcentaje a lo largo de la animación en la que se produce el keyframe así como un bloque que contiene la información de estilo para ese keyframe.

Puede listar los keyframes en cualquier orden, éstos serán tratados en el orden en que los porcentajes especificados indican que debe ocurrir.

Ejemplo:

```
@keyframes identifier {
     0% { top: 0; left: 0; }
     30% { top: 50px; }
     50% { left: 50px; }
     100% { top: 100px; left: 100%; }
}
```

En el ejemplo anterior, los porcentajes se refieren a cada instante dentro de la duración de la animación. Es decir, si la animación dura 4 segundos, el 50% se refiere a los dos segundos. Se trata de estilos que se añaden a los ya existentes durante la ejecución de la animación. El nombre del keyframe “identifier” será aquel que haya que asignar a `animation-name`.

Los valores 0% y 100% se pueden sustituir por `from` y `to`

```
@keyframes identifier {
      from { top: 0; left: 0; }
      30% { top: 50px; }
      50% { left: 50px; }
      to { top: 100px; left: 100%; }
}
```

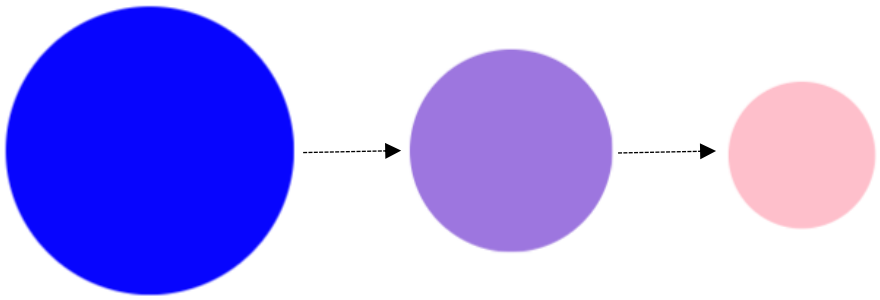
Veamos ahora varios ejemplos de animaciones que engloban la propiedad animation y los keyframes. En primer lugar, el desplazamiento del recuadro rojo de debajo de izquierda a derecha en 4 segundos mientras el navegador sigue abierto:

Los estilos necesarios para ello serían:

```
div.recuadro{
    width:100px;
    height:100px;
    background:red;
    position:relative;
    animation: mymove 5s infinite;
}

@keyframes mymove{
      from {left:0px;}
      to {left:200px;}
}
```

Desplazamiento de un círculo de izquierda a derecha mientras cambia de color. Además, en este caso se emplea un retardo de 3 segundos en los que se cambia al estilo inicial de la animación con backwards. La animación se ejecuta mientras el navegador sigue abierto.

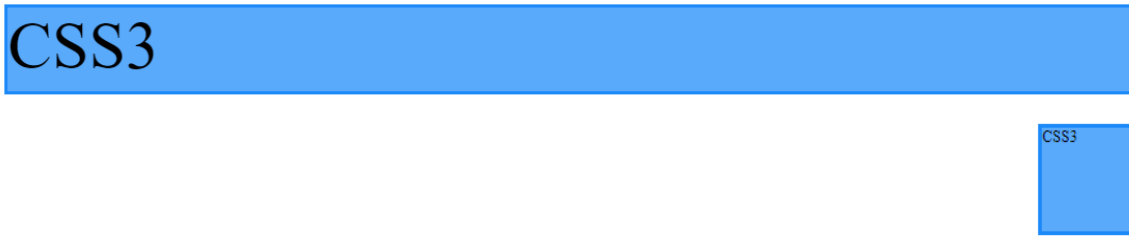


```
.ball {
      width: 200px;
      height: 200px;
      background-color: red;
      border-radius: 50%;
      animation: myAnim 2s backwards 3s infinite;
}
```

```
@keyframes myAnim {
	from {
		background-color: blue;
	}

	to {
		background-color: pink;
		transform: translateX(500px) scale(.5);
	}
  }
```

Un último ejemplo es el texto a continuación que aparece y desaparece alternando la dirección y cambiando el tamaño de texto.



En este caso se especifican las propiedades individualmente y los keyframes con porcentajes.

```
div {
	width: 80px;
	height: 80px;
	background-color: #5aaafc;
	border: 3px solid #1c89f9;
	animation-duration: 6s;
	animation-iteration-count: infinite;
	animation-name: MiAnimacion;
	animation-direction: alternate;
}

@keyframes MiAnimacion {
	0% {
		font-size:0%;
		margin-left:100%;
	}

	50% {font-size: 400%;}

	100% {font-size: 400%;
		margin-left: 20%;
		width: 100%;
	}
}
```

4. TRANSICIONES

Las transiciones CSS, parte del borrador de la especificación CSS3, proporcionan una forma de animar los cambios de las propiedades CSS, en lugar de que los cambios surtan efecto de manera instantánea. Por ejemplo, si se cambia el color de un elemento de blanco a negro, normalmente

el cambio es instantáneo. Al habilitar las transiciones CSS, el cambio sucede en un intervalo de tiempo que puedes especificar, siguiendo una curva de aceleración que puedes personalizar.

Las transiciones CSS se controlan mediante la propiedad abreviada **transition**. Es preferible utilizar este método porque de esta forma se evita que la longitud de la lista de parámetros sea diferente, lo que puede dar lugar a tener que emplear un tiempo considerablemente largo en depurar el código CSS.

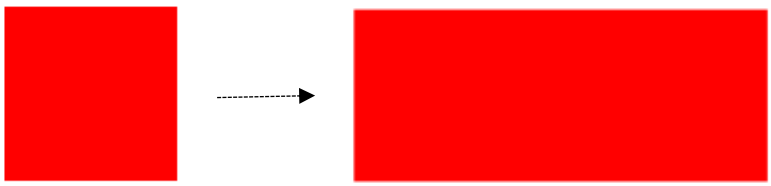
Puedes controlar los componentes individuales de la transición usando las siguientes subpropiedades:

- **transition-property:** Especifica el nombre o nombres de las propiedades CSS a las que deberían aplicarse las transiciones. Sólo las propiedades que se enumeran aquí son animadas durante las transiciones; los cambios en el resto de las propiedades suceden de manera instantánea como siempre. Ejemplos: `transition-property: width, height` o el valor `transition-property: all` que se aplicará a todas las propiedades susceptibles de aplicar una transición.
- **transition-duration:** Especifica la duración en la que sucederán las transiciones. Puedes especificar una única duración que se aplique a todas las propiedades durante la transición o valores múltiples que permitan a cada propiedad de transición un período de tiempo diferente. Se expresa en segundos (s) o milisegundos (ms). Ejemplos: `transition-duration: 6s` o `transition-duration: 1s, 15s` (que en el ejemplo anterior se podría aplicar respectivamente a width y height);
- **transition-timing-function:** Especifica la curva cúbica bézier que se usa para definir cómo se computan los valores intermedios para las propiedades. Admite valores similares a las animaciones. También permite más de un valor que se aplica respectivamente a cada propiedad. Por ejemplo: `transition-timing-function: ease-in` (se refiere a un comienzo lento y la velocidad aumenta progresivamente) o `transition-timing-function: ease, step-start` (ease se corresponde a aumentar la velocidad en el medio de la transición y disminuir al final, mientras que step-start se refiere a que la transición hace una parada al comienzo y luego se va directamente al estado final).
- **transition-delay:** Define el tiempo de espera entre el momento en que se cambia una propiedad y el inicio de la transición. El formato es igual que en `transition-duration`.

Sin embargo, como cada transición puede aceptar valores diferentes, suele ser más cómodo emplear la propiedad resumida **transition**. Por ejemplo, `transition: width 4s, height, 2s;`

Veamos varios ejemplos de definición de transiciones. La mayoría se activarán al usar el selector :hover, ya que permite cambiar el tamaño de un elemento dinámicamente y así ver el efecto de las transiciones.

El ejemplo de debajo ralentiza el cambio del ancho de un elemento y en lugar de hacerse inmediatamente se lleva a cabo en 2 segundos. Los 2 segundos se refieren a la duración, ya que con la propiedad abreviada si solo hay un valor de transition-duration y transition-delay, se opta por transition-duration.



```
div.recuadro {
   width: 100px;
   height: 100px;
   background: red;
   transition: width 2s;
}

div.recuadro:hover {
   width: 300px;
}
```

Otro ejemplo con los cuatro atributos de transition aplicados a dos propiedades sería el de debajo similar la anterior pero añadiendo rotación. En este caso, el retardo para el ancho es de 2 segundos, mientras que para transform de 1 segundo. Además, se ha definido transition-timing-function con el valor linear.

```
div.recuadro {
   width: 100px;
   height: 100px;
   background: red;
   transition: width 2s linear 2s, transform 4s linear 1s;
}

div.recuadro:hover {
   width: 300px;
   transform: rotate(180deg);
}
```

Y por último cinco divs, a los cuales se les aplica la misma transición de nuevo, pero con diferentes intervalos de velocidad.

```
div.recuadro {
   width: 100px;
   height: 100px;
   background: red;
   transition: width 2s;
}

div.recuadro:hover {
   width: 300px;
}

#div1 {transition-timing-function: linear;}
#div2 {transition-timing-function: ease;}
#div3 {transition-timing-function: ease-in;}
#div4 {transition-timing-function: ease-out;}
#div5 {transition-timing-function: ease-in-out;}
```