Animaciones con HTML y JavaScript

Propiedades transition y transform de CSS

La propiedad transition permiten realizar una simple animación cuando se cambia alguna de las propiedades de un elemento, aunque no todas las propiedades se pueden animar.

Tenemos las siguientes:

- transition-property: propiedad CSS que queremos animar. Admite varias propiedades separadas por comas, all para todas y none para ninguna
- transition-duration: el tiempo en segundos que durará la transición
- transition-delay: el tiempo en segundos de retraso antes de que comience la animación, siendo por defecto cero
- transition-timing-function: dispone de diferentes valores para cambiar la velocidad de la animación, pudiendo indicar que comience más rápido y luego más lento, al revés, ... Sus posibles valores son ease, ease-in, ease-out, ease-in-out, linear, step-start y step-stop. Incluso podemos indicar valores en alguna de ellas. Más información en https://developer.mozilla.org/en-US/docs/Web/CSS/transition-timing-function
- transition: para poner todo lo anterior en una sola propiedad

Ejemplo:

```
p {
    font-size: 3rem;
    color: red;
    transition-duration: .5s; /* Medio segundo */
    transition-property: color;
}
```

En este caso indicamos que queremos realizar una transición de medio segundo cuando cambie la propiedad color. Para ello en algún momento, ya sea con CSS o con JavaScript, deberemos cambiar el color del párrafo para que se reproduzca la animación.

Ejemplo:

```
p:hover {
    color: green;
}
```

Como hemos comentado no todas las propiedades se pueden animar y algunas de ellas solo funcionarán si le hemos dado antes un valor a la propiedad.

Ejemplo:

```
p {
    transition-duration: .5s; /* Medio segundo */
    position: relative;
    transition-property: left;
}
p:hover {
    left:100px;
}
```

Esta animación no funcionará pues left no tiene un valor dado. Para solucionarlo bastaría con darle left: 0;

Propiedad transform

Esta propiedad nos permite escalar, rotar, mover, deformar, ... un elemento.

Algunos de sus posibles valores son:

- **scale**(valor): para agrandar o empequeñecer un elemento. 1 es el valor normal, por debato de 1 la empequeñecemos, por encima de 1 la agrandamos

  Ejemplo:

  ```
  transform: scale(1.5); /* La agrandamos un 50% */
  transform: scale(2); /* La agrandamos un 100% */
  ```

- **scaleX**(valor): escala en el eje X
- **scaleY**(grados): escala en el eje Y
- **rotate**(grados): rota un elemento hacia la izquierda (valor negativo) o la derecha (valor positivo). Debe poner el número seguido de deg

  Ejemplo:

  ```
  transform: rotate(30deg);
  ```

- **rotateX**(grados): rotar en el eje X
- **rotateY**(grados): rotar en el eje Y
- **rotateZ**(grados): rotar en el eje Z
- **rotate**(grados): igual a rotateZ
- **translateX**(movimiento): desplaza el elemento en el eje X. Podemos indicar píxeles, rem, ...
- **translateY**(movimiento): desplaza el elemento en el eje Y. Podemos indicar píxeles, rem, ...
- **translateZ**(movimiento): desplaza el elemento en el eje Z. Podemos indicar píxeles, rem, ...
- **translate**(movimientoX, movimientoY): desplaza el elemento en el eje X e Y. Podemos indicar píxeles, rem, ... Si solo se indica un valor es igual a translateX
- **skew**(gradosX, gradosY): deforma un elemento en el eje X e Y. Si solo se indica un valor, lo hará en el eje X
- **skewX**(grados): deforma un elemento en el eje X
- **skewY**(grados): deforma un elemento en el eje Y

Podemos realizar más de una transformación.

Ejemplo:

```
transform: translateX(10px) rotate(10deg) translateY(5px);
```

Animaciones CSS

Las animaciones van un paso más allá de las transiciones, pudiendo crear una serie de cambios a lo largo de toda la animación, cambios que pueden ser de propiedades diferentes. A cada una de esas series se las llama keyframe (cuadro clave) encargándose el navegador de calcular todos los pasos para llegar a esta.

Para crear una animación usaremos un elemento **@keyframe** en los estilos CSS al que daremos un nombre que la identifique y luego usaremos la propiedad **animation** en el elemento deseado para usar esa animación.

Dentro de este elemento indicaremos los diferentes keyframes de varios modos que vamos a ir viendo mediante ejemplos, siendo el más sencillo el uso de porcentajes, siendo 0% el comienzo de la animación y 100% su final.

Ejemplo:

```
@keyframes rotar {
  0% {
    transform: rotate(10deg);
  }
  50% {
    transform: rotate(-10deg);
  }
  100% {
    transform: rotate(0);
  }
```

Notas:

- el nombre de la animación es rotar
- al comienzo de esta, 0%, rotamos el elemento 10 grados hacia abajo
- a su mitad, 50%, lo rotamos 10 grados hacia arriba
- al final de esta, 100%, lo volvemos a colocar en su posición inicial
- podemos poner todos los keyframes que deseemos
- se puede usar from en lugar de 0%
- se puede usar to en lugar de 100%

Luego indicaremos el uso de esa animación en el selector CSS deseado.

Ejemplo:

```
div {
  animation: 0.3s rotar;
}
```

Donde indicamos los segundos que durará la animación.

Podemos indicar varias o diferentes propiedades en cada keyframe.

Ejemplo:

```
@keyframes rotar {
  0% {
    transform: rotate(10deg);
    color: red;
  }
  50% {
    transform: rotate(-10deg);
  }
  100% {
    transform: rotate(0);
    color:green;
  }
```

En este ejemplo hemos usada la versión corta animation y solo hemos indicado dos valores, la duración y el nombre. Pero tenemos varias propiedades, siendo algunas de ellas:

- **animation-duration**: la duración en segundos de la animación
- **animation-name**: el nombre de la animación

- **animation-delay**: retraso en segundos para que comience la animación
- **animation-iteration-count**: número de veces que se repite la animación, siendo por defecto 1, pudiendo indicar el valor infinite para que se repita siempre. También es posible indicar valores con decimales, por ejemplo 0.5 para que solo se haga la mitad de la animación
- **animation-direction**: si ejecutar la animación en el orden establecido (norma), al revés (reverse), primero normal y luego al revés (alternate) y primero al revés y luego normal (alternate-reverse)
- **transition-timing-function**: marca como se realiza la animación, a la misma velocidad todo el tiempo (linear), más despacio al comienzo (ease-in), más despacio al final (ease-out), más rápido hacia la mitad (ease-in-out), ...
- **animation-play-state**: running para poner en marcha la animación y paused para pausarla, pudiendo continuar luego desde donde estaba

En todas estas propiedades, incluyendo animation, podemos indicar valores para más de una animación separada por comas.

En el siguiente ejemplo ejecutamos dos animaciones, rotar y escalar, las cuales se ejecutarán simultáneamente:

```
animation: 3s rotar, 3s escalar;
```

Si queremos que la segunda comience después de la primera, basta con ponerle un retraso igual a la duración de esa:

```
animation: 3s rotar, 3s 3s escalar;
```

Si ponemos las propiedades por separados también usaremos comas si tenemos más de una animación, debiendo usar el mismo orden en todas.

```
animation-duration: 3s, 5s;
animation-name:  rotar, escalar;
```

No es recomendable animar propiedades del modelo de cajas (márgenes, paddings, height y width pues consumen bastante CPU para los cálculos

Usando animaciones en JavaScript

Bastas con añadir el estilo deseado con style o añadir una clase donde esté definido el estilo.

Ejemplo:

```
    function animar() {
    // Suponemos que tenemos un elemento llamado parrafo
    const p = document.getElementById("parrafo");
    p.style.animation = "3s rotar, 3s escalar";
}

document.getElementById("boton").addEventListener("click", animar);
```

Una vez que se aplique el estilo o la clase, debemos poner el atributo animation a "" o eliminar la clase, si queremos que se vuelva a reproducir posteriormente.

Eventos de transiciones
Disponemos de los siguientes eventos:

- **transitionstart**: cuando la animación ha comenzado (después del delay / espera si lo hay)
- **transitionrun**: cuando la animación ha comenzado (antes del delay / espera si lo hay)
- **transitionend**: cuando la animación ha finalizado. No se ejecuta si se aborta la transición (por ponerle display: none o asignándole otro valor a la propiedad)

Eventos de animaciones
Disponemos de los siguientes eventos:

- **animationstart**: cuando la animación ha comenzado (después del delay / espera si lo hay)
- **animationend**: cuando la animación ha terminado
- **animationiteration**: cuando una iteración de la animación ha finalizado y va a comenzar otra, es decir, cuando indicamos que se repita dos o más veces

Bibliotecas de animación

Algunas bibliotecas conocidas para crear animaciones son:

- https://animejs.com/
- https://threejs.org/ (para 3D)
- http://velocityjs.org/
- https://gsap.com/