CSS Transitions

Transiciones y animaciones en CSS

Intro

El sistema de transiciones y animaciones en CSS es muy parecido a nivel conceptual, solo que la sintaxis cambia de manera importante. Las animaciones están más orientadas a ser bloques reutilizables.

Las transiciones podemos entenderlas tambien como animaciones con un inicio y un fotograma clave, mientras que las animaciones pueden tener 2 fotogramas clave o más.

Transition:
https://www.w3schools.com/css/css3_transitions.asp

Animation:
https://www.w3schools.com/css/css3_animations.asp

Vamos a ver las diferentes propiedades y sintaxis.

La propiedad transition

Es una propiedad resumen que recoje las siguientes propiedades:

- **Transition-duration**: La duración de la transición.
- **Transition-property**: La propiedad o propiedades que queramos transicionar.
- **Transition-delay:** El tiempo de retardo que pasa hasta que se dispara la transición.
- **Transition-timing-function**: La curva de interpolación de la transición.

```
.transition_element {
    transition-property: background-color;
    transition-duration: 350ms;
    transition-timing-function: ease-in;
    transition-delay: 100ms;
}
```

Transition property

Hacer referencia a la propiedad o propiedades que van a hacer transicion.

Para que exista una transición tiene que haber algún cambio en alguna propiedad del elemento HTML en cuestión, suele suceder en tres casos:

- Cambio de CSS desde JavaScript con la propiedad style
- Cambio de una clase
- Cambio por un pseudoselector como :focus o :hover

```css
button {
    background-color: #fff;
}
button:hover {
    background-color: #eee;
    transition-property: background-color;
    transition-duration: 350ms;
    transition-timing-function: ease-in;
}
```

Transition property

Una nota importante. No todas las propiedades son animables.

Animables son aquellas que tienen contenido numérico de cualquier tipo (relativo o absoluto o incluso con calc) como medidas, márgenes, paddings, bordes, o, contenido de color, opacidad, backgrounds, etc... etc...

Pero por ejemplo no podemos hacer una animación de display: none a display: block, ya que no hay pasos intermedios.

Transition property

```
button {
    background-color: #fff;
    color: #333;
}
button:hover {
    background-color: #222;
    color: #fff;
    transition-property: background-color, color;
    transition-duration: 350ms, 200ms;
    transition-timing-function: ease-in, ease;
}
```

Si queremos modificar varias propiedades podemos separarlas por comas.

Y si queremos que pueda ser animable cualquier propiedad que cambie se pone la palabra clave “all”.

```
.myClass {
    transition-property: background-color, color, padding, transform;
    transition-property: all;
}
```

Transition duration y Transition delay

Transition duration hace referencia al tiempo que tarda en completarse la transición.

Transition delay hace referencia al tiempo que tarda en comenzar la transición desde que se detecta el cambio de estado, ya sea hover, cambio de clase, o cambio de estilo por JS

Ambas propiedades se expresan en milisegundos (ms) o en segundos (s)

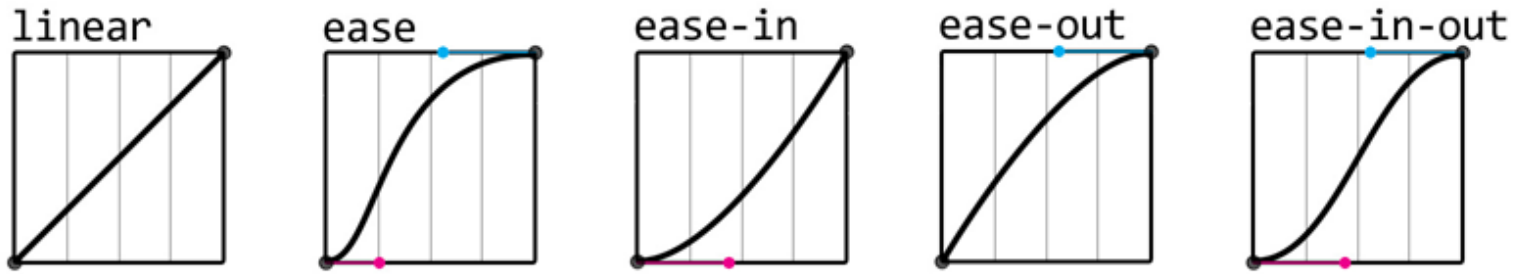
```
.transition_element {
    transition-duration: 350ms;
    transition-duration: 0.5s;
    transition-duration: 2s;
    transition-delay: 300ms;
}
```

Transition-timing-function

Esta propiedad hace referencia a la curva de interpolación entre dos fotogramas.

Las curvas de interpolación describen las aceleraciones, deceleraciones, etc.. que se pueden dar en el proceso de interpolación entre dos puntos.

https://easings.net/



Transition-timing-function

- **ease**: Es el valor por defecto
- **linear**: Hace una transición totalmente linear.
- **ease-in**
- **ease-out**
- **ease-in-out**
- **step**(a). Hace la transición por pasos. A es el número de pasos
- **cubic-bezier**(a, b, c, d). Definen la curva bezier de transición. Mejor hacerlo visualmente. Para ello hay un par de herramientas interesantes: Una es la curva que aparece en el Chrome DevTools cuando declaramos la propiedad transition-timing-function, otra es en la página http://easings.net/es.

Transition

La propiedad transition a secas se suele usar como propiedad resumen para el resto y es la que se suele usar en la gran mayoría de los casos.

Antiguamente se solía prefijar la propiedad transition, pero según caniuse.com se puede usar sin prefijar en todos los navegadores sin problema.

```
.myClass {
    transition: background-color 350ms ease 0ms;
}
```

Transition aplicada a esconder algo

Para esconder algo no se usa display, si no que se usa visibility hidden, que no es un propiedad animable pero permite una transición entre ambos estados.

```
.modal_window {
    visibility: visible;
    opacity: 1;
    transition: visibility 500ms ease, opacity 500ms ease;

    &.closed {
        visibility: hidden;
        opacity: 0;
        transition: visibility 500ms ease, opacity 500ms ease;
    }
}
```

Animation

La propiedad animation tiene una sintaxis parecida a transition con algunas opciones más, en lugar de definir qué propiedades animamos en la etiqueta, declaramos qué animación se hace y en la animación se declaran los fotogramas clave y las propiedades que se animan.

En animation tenemos una doble sintaxis, la de la declaración en la regla CSS y la de la regla @keyframes.

```css
.myAnimation {
    animation-name: myAnimation;
    animation-duration: 350ms;
    animation-delay: 100ms;
    animation-timing-function: ease;
    animation-iteration-count: 4;
}

@keyframes myAnimation {
    from {
        background-color: #333;
        color: #fff;
    }
    to {
        background-color: #fff;
        color: #333;
    }
}
```

@keyframes

Es la definición de la animación.
La sintaxis es la siguiente.

En los valores se puede poner las palabras from, to, donde tendríamos dos fotogramas clave. Si no, se puede poner por porcentajes (0%, 99%, 100%)

From y to son dos atajos para decir 0% y 100%. Ambas maneras son posibles.

En la declaración entre llaves cambiamos el CSS necesario, siempre teniendo en cuenta que estén las mismas propiedades en todos los fotogramas.

```
@keyframes myAnimation {
    from {
        background-color: #333;
        color: #fff;
    }
    to {
        background-color: #fff;
        color: #333;
    }
}
@keyframes otherAnimation {
    0% {
        background-color: #333;
    }
    10% {
        background-color: rgb(204, 45, 45);
    }
    40% {
        background-color: rgb(45, 154, 204);
    }
    100% {
        background-color: rgb(204, 180, 45);
    }
}
```

Animation-duration, animation delay y animation-timing-function

Es el mismo fenómeno que en el caso de transition.

Habla de la duración o del retardo de la animación en segundos o milisegundos, igual que en la transición.

Y timing-function es igual también.

```
.myAnimation {
    animation-name: myAnimation;
    animation-duration: 350ms;
    animation-delay: 30ms;
    animation-timing-function: cubic-bezier(0.075, 0.82, 0.165, 1);
}

@keyframes myAnimation {
    from {
        background-color: #333;
        color: #fff;
    }
    to {
        background-color: #fff;
        color: #333;
    }
}
```

Animation-iteration-count

Define cuantas veces se va a repetir la animación.

Acepta un número entero, o acepta la palabra clave infinite, que hace que se repita la animación una y otra vez.

```
.myAnimation {
    animation-name: myAnimation;
    animation-duration: 350ms;
    animation-delay: 100ms;
    animation-timing-function: ease;
    animation-iteration-count: 4;
}

@keyframes myAnimation {
    from {
        background-color: #333;
        color: #fff;
    }
    to {
        background-color: #fff;
        color: #333;
    }
}
```

Animation-direction

Permite ejecutar la animación en diferentes direcciones, es decir colocando los valores from, to, o numéricos de manera simétrica. Se hace con palabras clave.

- Normal: Ejecuta la animación como está definida en el @keyframes.
- Reverse: Ejecuta la animación al contrario de como está definida en el @keyframes.
- Alternate: Ejecuta la animación en las veces impares de modo normal y las pares como reverse. Este valor tiene sentido en el momento en el que la animación tenga varias iteraciones.
- Alternate-reverse: Ejecuta la animación en las veces pares de modo normal y las impares como reverse.

```
.myAnimation {
    animation-name: myAnimation;
    animation-duration: 350ms;
    animation-delay: 30ms;
    animation-timing-function: ease;
    animation-iteration-count: 4;
    animation-direction: reverse;
}
@keyframes myAnimation {
    from {
        background-color: #333;
        color: #fff;
    }
    to {
        background-color: #fff;
        color: #333;
    }
}
```

Animation-fill-mode y Animation-play-state

Animation fill mode tiene varias opciones, pero se usan dos: forwards o backwards.

- Forwards hace que cuando termine la animación, las propiedades modificadas se queden aplicadas.
- Backwards hace que una vez terminada la animación se vuelva a los datos anteriores.

```
.myAnimation {
    animation-name: myAnimation;
    animation-duration: 350ms;
    animation-delay: 30ms;
    animation-timing-function: ease;
    animation-iteration-count: 4;
    animation-direction: reverse;
    animation-fill-mode: forwards;
    animation-play-state: paused;
}
```

Animation play state permite parar o reanudar una animación que está en ejecución.

Animation

Es la propiedad resumen de todas las propiedades anteriores. Si alguna de las propiedades la dejamos en su valor por defecto se puede omitir. El orden sería:

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

```
.myAnimation {
    animation-name: myAnimation;
    animation-duration: 350ms;
    animation-delay: 30ms;
    animation-timing-function: ease;
    animation-iteration-count: 4;
    animation-direction: reverse;
    animation-fill-mode: forwards;
    animation-play-state: running;
}
.myAnimation {
    animation: myAnimation 350ms ease 0ms infinite normal forwards running;
}
```

Transform

Transform es una propiedad de CSS3 que nos permite meter cambios en cualquier elemento como si se tratara de un elemento individual y modificarlo.

Para usar transiciones y animaciones que tengan que ver con posicionamiento conviene usar transform en lugar de otras opciones.

La sintaxis es un poco especial, y consta de 3 partes: translate, rotate, scale. Tiene más propiedades más complejas, tridimensionales, etc.. pero no las vemos.

```
.myElement {
    transform: translate(X, Y) rotate(deg) scale(factor);
    transform: translate(40px, 3%);
    transform: rotate(90deg);
    transform: translate(-50%, -50%) scale(1.1);
}
```

Disclaimer

Las transiciones y animaciones hoy en día las vemos en todos lados dentro de la web y de las aplicaciones web. En cierto modo cada vez podríamos decir que hay más cosas animándose y con más complejidad.

Las técnicas que hemos explicado hoy valen para transicionar de manera sencilla o hacer alguna animación sencilla de algún elemento.

En caso de que queramos meter nociones de motion design y animaciones complejas, sincronizadas, aunque se pueden hacer con CSS, conviene hacerlas con JS con librerías como GSAP o AnimeJS. Estas librerías cambian los parámetros de CSS desde JS basadas en un stream de tiempo. Y permite auntenticas maravillas.

En caso de querer hacer animaciones ultracomplejas con character design, recomiendo familiarizarse con el software de animación After Effects y la librería BodyMovin.

Este tema en realidad daría para un MOOC en sí.