



Frontend I

Clase 19

Animaciones



**Certified
Developer**
The Ultimate Tech Degree

DigitalHouse >
Coding School



Temas

1

Introducción

2

Repaso

3

Cierre

4

Actividad





Animaciones

<https://codepen.io/jaymar-g-aranas/pen/bGdRrQa>

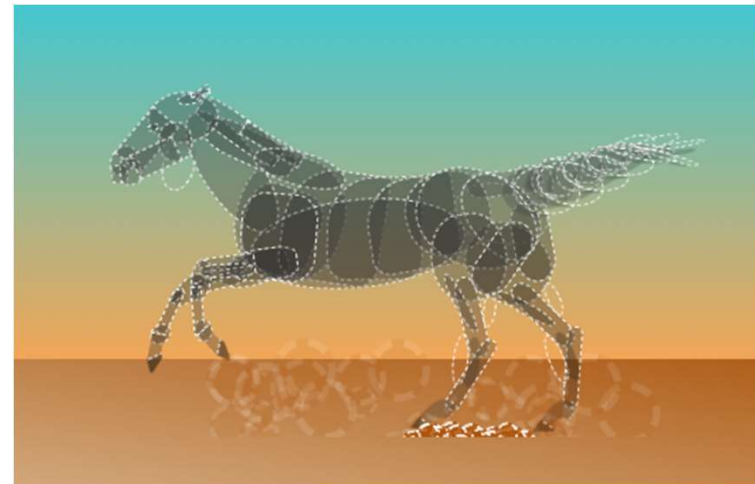




Animaciones

<https://codepen.io/jaymar-g-aranas/pen/bGdRrQa>

```
input[type=checkbox]:checked ~ label
{
  --outlines: white;
  --speed: 8s;
  --color-horse: rgba(50, 50, 50,
0.3);
  --color-horse-back: rgba(30, 30,
30, 0.3);
  --color-hair: rgba(70, 70, 70,
0.3);
  --color-hoof: rgba(0, 0, 0, 0.3);
}
```





Transform

DigitalHouse >
Coding School

Translate

```
transform: translate (20%);
```

```
transform: translate (20%, 30%);
```

DigitalHouse >
Coding School

Scale

```
transform: scale (2);
```

```
transform: translate (2,3);
```

DigitalHouse >
Coding School

Skew

```
transform: skew (10deg);
```

```
transform: skew (10deg, 5deg);
```

DigitalHouse >
Coding School

Rotate

```
transform: rotate (20deg)
```

```
transform: rotate (.20deg)
```



Transform-origin



transform-origin

```
transform-origin: bottom right (.60px)
```

```
transform: rotate (20deg)
```



CSS

```
@keyframes fantasma {  
    /* Aquí definiremos los puntos de la animación */  
}
```

```
@keyframes move{  
    0%{  
    50%{  
    100%{  
}
```

```
@keyframes move{  
    to{  
    from{  
}
```

- No necesito pseudoclasas para que funcione la animación.
- Se puede fraccionar lo que dura la animación con porcentajes para hacerla mucho más específica.
- Se puede seguir usando hover y focus pero no es condición necesaria.



Keyframes

Primero definimos y luego la usamos en donde la necesitemos

```
@keyframes move{  
  0%{  
    transform: rotate(0);  
  }  
  50%{  
    transform: rotate(40deg);  
  }  
  100%{  
    transform: rotate(300deg);  
  }  
}
```

```
animation: move 3s infinite alternate;  
}
```





Animation

Se pueden utilizar las siguientes propiedades para crear animaciones:

animation-name: Especifica el nombre de la animación que se va a aplicar.

animation-duration: Especifica la duración de la animación en segundos o milisegundos.

animation-timing-function: Especifica la función de tiempo que se utiliza para la animación (por ejemplo, ease-in, ease-out, linear, etc.).

animation-delay: Especifica el tiempo de retraso antes de que comience la animación.

animation-iteration-count: Especifica el número de veces que se debe repetir la animación (por ejemplo, 2, 3, infinite, etc.).

animation-direction: Especifica la dirección de la animación (por ejemplo, normal, reverse, alternate, alternate-reverse, etc.).

animation-fill-mode: determina cómo se aplican las propiedades CSS antes y después de la animación.

```
.box {  
  animation-name: myanimation;  
  animation-duration: 2s;  
  animation-timing-function: ease-in-out;  
  animation-delay: 0s;  
  animation-iteration-count: infinite;  
  animation-direction: alternate;  
  animation-fill-mode: both;  
  animation-play-state: running;  
}
```



animation-timing-function

Hay varias opciones frecuentes para el valor de animation-timing-function, que determinan cómo se debe reproducir la animación a lo largo del tiempo. Aquí hay algunos ejemplos:

ease: esta es la opción predeterminada y se utiliza para crear una animación suave y gradual al principio y al final, con una aceleración y desaceleración suave en el medio.

linear: esta opción produce una animación uniforme y constante a lo largo de toda la duración.

ease-in: esta opción se utiliza para crear una animación suave al principio y gradualmente se acelera hacia el final.

ease-out: esta opción se utiliza para crear una animación rápida al principio que gradualmente se desacelera hacia el final.

ease-in-out: esta opción se utiliza para crear una animación suave al principio y gradualmente se acelera hacia la mitad, luego se desacelera hacia el final.

cubic-bezier: esta opción permite definir una curva personalizada que controla cómo se reproduce la animación a lo largo del tiempo.





animation-direction

la propiedad animation-direction se utiliza para especificar la dirección en la que una animación debe ejecutarse. Los valores posibles para animation-direction son:

normal: La animación se reproduce en orden normal, desde el primer cuadro hasta el último.

reverse: La animación se reproduce en orden inverso, desde el último cuadro hasta el primero.

alternate: La animación se reproduce primero hacia adelante y luego hacia atrás, alternando entre los dos.

alternate-reverse: La animación se reproduce primero hacia atrás y luego hacia adelante, alternando entre los dos.

El valor por defecto de animation-direction es normal.





animation-fill-mode

none: Esta es la opción por defecto. No se aplican los estilos definidos en la animación antes o después de la misma.

forwards: Los estilos definidos en la animación se aplican al elemento al final de la animación y se mantienen en ese estado.

backwards: Los estilos definidos en la animación se aplican al elemento en el momento en que se define la animación, antes de que ésta comience.

both: Los estilos definidos en la animación se aplican tanto al inicio como al final de la misma.

Estas opciones determinan cómo se aplican las propiedades CSS definidas en la animación antes y después de la misma, y permiten personalizar el comportamiento de la animación.





Steps()

La función **steps()** es una función que se utiliza para crear animaciones por saltos o por pasos en lugar de animaciones fluidas.

La función `steps()` **especifica cuántos pasos de animación deben ocurrir durante la duración de la animación y cómo se deben repartir estos pasos**. La función toma dos valores como argumentos: el número de pasos y la posición en la animación en la que deben ocurrir los pasos. Por ejemplo, la función `steps(4, start)` crearía una animación por pasos en la que se producen 4 pasos, todos al comienzo de la animación.

La sintaxis completa para utilizar `steps()` en una animación CSS es: **`steps(<n>, [start | end])`**.

La función `steps()` es útil para crear animaciones que se asemejan a animaciones por fotogramas clave o por cuadros de animación. Puede crear animaciones que simulan el movimiento de un objeto que se mueve de una posición a otra en intervalos discretos, en lugar de una animación fluida y continua.





Animation forma abreviada

Cuando se utiliza la propiedad animation de forma abreviada para especificar todas las propiedades de la animación en una sola línea, las propiedades deben aparecer en el siguiente orden:

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

```
css .fantasma {  
    animation: fantasma 3s infinite;  
}
```





Transition

La propiedad transition en CSS permite agregar **efectos de transición suaves** a los cambios de estilo de un elemento. Cuando se aplica la propiedad **transition** a un elemento, se especifican las propiedades CSS que deben tener una transición suave y el tiempo que debe durar la transición.

La propiedad transition tiene varios valores que se pueden establecer:

transition-property: especifica la propiedad CSS o propiedades que deben tener una transición suave.

transition-duration: especifica la duración de la transición.

transition-timing-function: especifica la función de tiempo que se utiliza para controlar la velocidad de la transición.

transition-delay: especifica un retraso antes de que comience la transición.





Transition versión abreviada

Estos valores se pueden establecer en una sola línea, utilizando la propiedad abreviada transition, o por separado utilizando las propiedades individuales correspondientes.


```
css
button {
  background-color: #eaeaea;
  color: #000;
  transition:
    background-color 0.5s,
    color 0.5s;
}
```

```
css
button {
  background-color: #1a73e8;
  color: #fff;
  transition: all 0.5s
}
```




Transition

```
# estilos.css > #nombre
1  .box{
2      width: 100px;
3      height: 100px;
4      background-color: coral;
5      transition: transform 1.5s;
6  }
7  .box:hover{
8      transform: rotate(45deg);
9  }
10 #nombre{
11     transition: transform 1.5s;
12 }
13 #nombre:focus{
14     transform: scaleX(2);
15 }
16
```





spinner

```
/* Hacemos girar a spinner */  
.spinner {  
  width: 64px;  
  height: 64px;  
  position: fixed;  
  top: calc(50% - 32px);  
  left: calc(50% - 32px);  
  border-radius: 50%;  
  z-index: 2;  
  background-color: transparent;  
  border-top: 10px solid yellow;   
  border-right: 10px solid blue;   
  border-bottom: 10px solid red;   
  border-left: 10px solid transparent;   
}  
/*
```

```
@keyframes spinner{  
  from{transform rotate(0)}  
  to{transform rotate(1turn)}  
}
```





Páginas

<https://animista.net/play/basic/flip-scale>

<https://projects.verou.me/animatable/#border-width/3>





Actividad



https://drive.google.com/file/d/1PIU6HjZeja6wY6Hn4-AbqPpwOmLz_zkh/view





Conclusiones

En conclusión, las animaciones en CSS ofrecen una forma sencilla y eficaz de crear efectos visuales dinámicos en sitios web. Con la capacidad de personalizar la duración, el retraso y otros parámetros de la animación, es posible crear una amplia variedad de efectos de animación, desde simples transiciones hasta animaciones complejas. Aprender a utilizar las animaciones de CSS puede mejorar significativamente la calidad visual de un sitio web y hacer que la experiencia de usuario sea más atractiva e interesante.



DigitalHouse>
Coding School