Animations

Introduction

CSS animations allow you to add movement and transitions to your elements. With CSS, you can define keyframes that specify the animation's starting and ending points, as well as intermediate steps for smooth transitions.

👉 Animations play a critical role in improving the experience of users as they assist in providing improved visual feedback and helps in making interactions with the website striking.

CSS 3 has an abundance of in-built properties that substantially help in animating elements, and that too across all the major browsers.

But — With great power, comes greater responsibility.

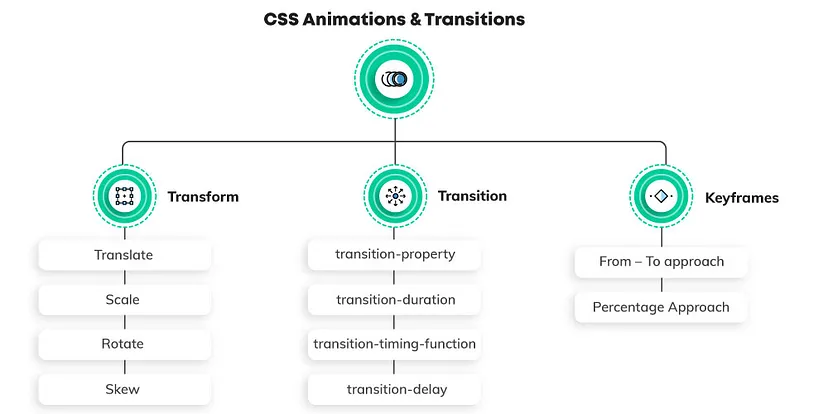
Animations need to be used wisely, or you could end up creating a page that is more distracting than interactive. The entire purpose of animations is to serve as an aid, rather than a hindrance while using web applications.

CSS animations have 3 significant aspects to it:

Transforms

Transitions

Keyframes



**Transform:**

Transforms helps to change your web elements in all kinds of wonderful ways — from moving the element to re-sizing it, from rotating the element to tilting it.

**And the best part —** You can alter anything and everything without changing the document flow.

**There are four significant aspects of transforms:**

1. Translate
2. Scale
3. Rotate
4. Skew

1.Translate

1. <style>
2. .translatable {
3. width: 100px;
4. height: 100px;
5. background-color: lightblue;
6. transition: transform 0.5s ease; /\* Adding a smooth transition effect \*/
7. }
8. .translatable:hover {
9. transform: translate(50px, 50px);
10. /\* You can also use individual axis translations like translateX(50px) translateY(20px) \*/
11. }
12. </style>
13. </head>
14. <body>
15. <div class="translatable">Hover me to translate!</div>
16. </body>

2.Scale

 <style>

        .scalable {

            width: 100px;

            height: 100px;

            background-color: lightgreen;

            transition: transform 0.5s ease; /\* Adding a smooth transition effect \*/

        }

        .scalable:hover {

            transform: scale(1.5, 0.5);

            /\* You can also use individual axis scaling like scaleX(1.5) scaleY(0.5) \*/

        }

    </style>

</head>

<body>

    <div class="scalable">Hover me to scale!</div>

</body>

3.Rotate

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Rotate Example</title>

    <style>

        .rotatable {

            width: 100px;

            height: 100px;

            background-color: lightcoral;

            transition: transform 0.5s ease; /\* Adding a smooth transition effect \*/

        }

        .rotatable:hover {

            transform: rotate(180deg);

            /\* You can use different degree values for the amount of rotation \*/

        }

    </style>

</head>

<body>

    <div class="rotatable">Hover me to rotate!</div>

</body>

</html>

1. 4. Skew
2. <!DOCTYPE html>
3. <html lang="en">
4. <head>
5. <meta charset="UTF-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1.0">
7. <title>Skew Example</title>
8. <style>
9. .skewable {
10. width: 100px;
11. height: 100px;
12. background-color: rgb(232, 17, 17);
13. transition: transform 0.5s ease; /\* Adding a smooth transition effect \*/
14. }
15. .skewable:hover {
16. transform: skew(45deg, 45deg);
17. /\* You can also use individual axis skewing like skewX(45deg) skewY(80deg) \*/
18. }
19. </style>
20. </head>
21. <body>
22. <div class="skewable">Hover me to skew!</div>
23. </body>
24. </html>

**One can control the transitions with the help of the following properties:**

* transition-property
* transition-duration
* transition-timing-function
* transition-delay

transition-property: Specifies the name of the CSS property to which the transition should be applied. It can be one or more properties separated by commas.

transition-duration: Specifies the duration over which the transition should occur. It is expressed in seconds (s) or milliseconds (ms).

transition-timing-function: Specifies the timing function that defines how the intermediate values of the transition are calculated. It can be values like ease, linear, ease-in, ease-out, ease-in-out, or a cubic-bezier function.

transition-delay: Specifies a delay before the transition effect starts. It is expressed in seconds (s) or milliseconds (ms).

The keyframes help in changing the animation from one to another at certain times.

### One can play with keyframes in two ways:

1. From — To approach
2. Percentage Approach

## 1. From — To Approach

In this approach, there are only 2 states during the animation — start state and end state.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Car Animation - Horizontal Movement</title>

    <style>

        @keyframes moveCar {

            0% {

                left: 0;

            }

            100% {

                left: 80%; /\* You can adjust the destination position as needed \*/

            }

        }

        .car {

            position: absolute;

            width: 100px;

            height: 50px;

            background-color: #3498db; /\* Car color \*/

            left: 0; /\* Initial position \*/

            top: 50%; /\* Center vertically \*/

            transform: translate(0, -80%);

            animation: moveCar 5s linear infinite alternate;

        }

    </style>

</head>

<body>

    <div class="car">Car</div>

</body>

</html>

**2. Percentage Approach**

We cannot use from-to in **@keyframes** as there is more than 1 state now. The percentage approach breaks down the animation into various intermediate states including a start and end state. The start state is indicated by 0%, and the end state is indicated by 100%. There can be as many intermediate states you want, but it is advised to keep the segregation of states uniform throughout.

**Example**: 0% — 25% — 50% — 75% — 100%

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Bouncing Ball Animation</title>

    <style>

        @keyframes bounce {

            0%, 20%, 50%, 80%, 100% {

                transform: translateY(0);

            }

            40% {

                transform: translateY(0px);

            }

            60% {

                transform: translateY(30px);

            }

            80% {

                transform: translateY(100px);

            }

            100% {

                transform: translateY(200px);

            }

        }

        .container {

            position: relative;

            width: 200px;

            height: 200px;

            margin: 50px auto;

            border: 2px solid #333;

            overflow: hidden;

        }

        .ball {

            width: 50px;

            height: 50px;

            background-color: red;

            border-radius: 50%;

            position: absolute;

            animation: bounce 2s infinite;

        }

    </style>

</head>

<body>

    <div class="container">

        <div class="ball"></div>

    </div>

</body>

</html>