

index.html

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <link rel="stylesheet" href="style.css">
8      <title>CSS Animations</title>
9  </head>
10
11 <body>
12
13     <h1>CSS Animations</h1>
14
15     <p>CSS Animations are essential for controlling the movement and appearance of
16         elements on web pages. They consist of defining animation sequences using @keyframes
17         and applying these sequences to elements using various animation properties.</p>
18
19     <p>To understand CSS animations completely we need to master these things:</p>
20
21     <ol>
22         <li>Transforms</li>
23         <li>Transitions</li>
24         <li>Keyframes</li>
25
26     </ol>
27
28     <h2>Transform Properties</h2>
29
30     <ul>
31
32         <li><strong>Transform</strong>: It helps us to make changes to our HTML elements in a 2D plane by
33             allowing them to move from one position to other, resizing them, rotating them and
34             stretching them. There are four main aspects of transforms viz scale (resizing), rotate,
35             translate (moving) and skew (stretching).</li> <br>
36
```

Translate - Transform: It helps to move an HTML element either horizontally or vertically or both. To apply this transformation we use the transform rule. This rule takes multiple transformation actions space separated. To move in X axis direction or we can say horizontally we can use translateX(). To move in Y axis direction or we can say vertically we can use translateY().

<h4>We can translate our elements by using the following ways:**</h4>**

****translateX()****: It moves the element along the X axis.**
**

****translateY()****: It moves the element along the Y axis.**
**

****translateZ()****: It moves the element along the Z axis.**
**

****translate()****: This can be used for a 2D translation.**
**

****translate3d()****: This can be used for 3D translation.**
**

**
**

Rotation - Transform: It helps to rotate an HTML element. To apply this transformation we use the transform rule. This rule takes multiple transformation actions space separated. To rotate the element along the X, Y or Z axis we can use rotate().

<h4>We can rotate our elements by using the following ways:**</h4>**

****rotateX()****: It rotates the element along the X axis.**
**

****rotateY()****: It rotates the element along the Y axis.**
**

```
75         <li><strong>rotateZ()</strong>: It rotates the element along the Z axis (out of the screen).</li> <br>
76
77         <li><strong>rotate()</strong>: This can be used for a 2D rotation.</li> <br>
78
79         <li><strong>rotate3d()</strong>: This can be used for 3D rotation.</li> <br>
80
81     </ol>
82
83 </li>
84
85 <li><strong>Scale - Transform</strong>: It helps to resize an HTML element either small or large.
86
87     <ol>
88
89         <h4>We can resize our elements by using the following ways:</h4>
90
91         <li><strong>scaleX()</strong>: It resizes the element along the X axis.</li> <br>
92
93         <li><strong>scaleY()</strong>: It resizes the element along the Y axis.</li> <br>
94
95         <li><strong>scaleZ()</strong>: It resizes the element along the Z axis.</li> <br>
96
97         <li><strong>scale()</strong>: This can be used for a 2D scaling.</li> <br>
98
99         <li><strong>scale3d()</strong>: This can be used for 3D scaling.</li> <br>
100
101     </ol>
102
103 </li>
104
105 <br>
106
107 <li><strong>Skew - Transform</strong>: If you want to tilt or stretch an element, you can use skew().
108
109     <ol>
110
111         <h4>We can skew our elements by using the following ways:</h4>
112
```




```
113     <li><strong>skewX()</strong>: It tilts the element along the X axis.</li> <br>
114
115     <li><strong>skewY()</strong>: It tilts the element along the Y axis.</li> <br>
116
117     <li><strong>skew()</strong>: This can be used for a 2D skewing.</li> <br>
118
119     <li><strong>skew3d()</strong>: This can be used for 3D skewing.</li> <br>
120
121     </ol>
122
123 </li>
124
125 </ul>
126
127 <h2>Transition Properties</h2>
128
129 <p>CSS transitions enable web developers to control the smooth transition between two states of an element. For
130 instance, when a user hovers over a button, the background color of the element can change seamlessly using CSS
131 selectors and pseudo-classes.</p>
132
133 <ul>
134
135     <li><strong>transition-duration:</strong>: Defines the length of time a transition animation should take.</li>
136     <br>
137
138     <li><strong>transition-delay:</strong>: Defines the amount of time to wait before the transition animation
139         starts.</li> <br>
140
141     <li><strong>transition-timing-function:</strong>: Defines the speed curve of the transition animation.
142
143         <ol>
144
145             <h4>We can define the speed curve of the transition animation by using the following ways:</h4>
146
147             <li><strong>ease</strong>: This is a slow start and fast end.</li> <br>
148
149             <li><strong>ease-in</strong>: This is a slow start and fast end.</li> <br>
150
```

```
151     <li><strong>ease-out</strong>: This is a fast start and slow end.</li> <br>
152
153     <li><strong>ease-in-out</strong>: This is a slow start, fast middle and slow end.</li> <br>
154
155     <li><strong>linear</strong>: This is a slow start and fast end.</li> <br>
156
157     <li><strong>steps-end</strong>: This is with a constant start and fast end.</li> <br>
158
159     <li><strong>steps-start</strong>: This is with a fast start and constant end.</li> <br>
160
161     <li><strong>cubic-bezier()</strong>: This can be used to define the custom curve of the transition
162         animation.</li> <br>
163
164     </ol>
165
166 </li> <br>
167
168 </ul>
169
170 <ol>
171
172     <h4>CSS animations involve several key properties:</h4>
173
174     <li><strong>@keyframes:</strong>: Defines the style of the element at different points of time.</li> <br>
175
176     <li><strong>animation-name:</strong>: Defines the name of the animation.</li> <br>
177
178     <li><strong>animation-duration:</strong>: Defines the length of time the animation should take.</li> <br>
179
180     <li><strong>animation-timing-function:</strong>: Defines the speed curve of the animation.</li> <br>
181
182     <li><strong>animation-delay:</strong>: Defines the amount of time to wait before the animation starts.</li> <br>
183
184     <li><strong>animation-iteration-count:</strong>: Defines the number of times the animation should be
185         played.</li> <br>
186
187     <li><strong>animation-direction:</strong>: Defines whether the animation should be played forwards, backwards
188         or in alternate cycles.
```

```
189
190     <ol>
191
192         <h4>We can define the direction of the animation by using the following ways:</h4>
193
194         <li><strong>Normal</strong>: The animation will be played forwards.</li> <br>
195
196         <li><strong>Reverse</strong>: The animation will be played backwards.</li> <br>
197
198         <li><strong>Alternate</strong>: The animation will be played forwards then backwards.</li> <br>
199
200         <li><strong>Alternate-reverse</strong>: The animation will be played backwards then forwards.</li>
201
202     </ol>
203
204 </li> <br>
205
206 <li><strong>animation-fill-mode:</strong>: Defines how the animation should apply to the element once it
207     has finished.</li> <br>
208
209 <li><strong>animation-play-state:</strong>: Defines whether the animation is running or paused.</li> <br>
210
211 </ol>
212
213 <h2>Transitions Example</h2>
214
215 <div id="parent">
216
217     <div id="red">
218
219         <div id="line"></div>
220
221     </div>
222
223 </div>
224
225 <h2>Animations Example</h2>
226
```

```
227     <div id="box">
228
229     </div>
230
231 </body>
232
233 </html>
```

style.css

```
1  * {
2    letter-spacing: 0.5px;
3  }
4
5  #parent {
6    width: 200px;
7    height: 200px;
8    background-color:  blue
9    display: flex;
10   justify-content: center;
11   align-items: center;
12   margin: 0 auto;
13 }
14
15 /* Transform Properties */
16
17 #red {
18   width: 100px;
19   height: 100px;
20   background-color:  red
21   display: flex;
22   justify-content: center;
23   align-items: center;
24   transform: translate(10px, 10px);
25   transform: rotate(50deg);
26   transform: scale(1.5, 1.5);
27   transform: skew(10deg, 10deg);
28   transition-duration: 1s;
29   transition-delay: 0.5s;
30   transition-timing-function: ease-in;
31 }
32
33 #line {
34   width: 50px;
35   height: 5px;
36   background-color:  black
```



```
37 }
38
39 h1,
40 h2 {
41     text-align: center;
42 }
43
44 p,
45 li {
46     font-size: 18px;
47 }
48
49 #red:hover {
50     background-color:  green
51     cursor: pointer;
52 }
53
54 #box {
55     width: 100px;
56     height: 100px;
57     background-color:  red
58     margin: 0 auto;
59 }
60
61 /* Animation Properties */
62
63 #box:hover {
64     animation-name: change-color;
65     animation-duration: 2s;
66     animation-delay: 1s;
67     animation-iteration-count: infinite;
68     animation-direction: alternate;
69     animation-timing-function: ease-in-out;
70     cursor: pointer;
71 }
72
73 @keyframes change-color {
74
```

```
75  0% {  
76      background-color:  red  
77  }  
78  
79  50% {  
80      background-color:  green  
81  }  
82  
83  100% {  
84      background-color:  blue  
85  }  
86  }
```

CSS Animations

CSS Animations are essential for controlling the movement and appearance of elements on web pages. They consist of defining animation sequences using `@keyframes` and applying these sequences to elements using various animation properties.

To understand CSS animations completely we need to master these things:

1. Transforms
2. Transitions
3. Keyframes

Transform Properties

- **Transform:** It helps us to make changes to our HTML elements in a 2D plane by allowing them to move from one position to other, resizing them, rotating them and stretching them. There are four main aspects of transforms viz scale (resizing), rotate, translate (moving) and skew (stretching).
- **Translate - Transform:** It helps to move an HTML element either horizontally or vertically or both. To apply this transformation we use the transform rule. This rule takes multiple transformation actions space separated. To move in X axis direction or we can say horizontally we can use `translateX()`. To move in Y axis direction or we can say vertically we can use `translateY()`.

We can translate our elements by using the following ways:

1. **translateX():** It moves the element along the X axis.
2. **translateY():** It moves the element along the Y axis.
3. **translateZ():** It moves the element along the Z axis.
4. **translate():** This can be used for a 2D translation.

5. **translate3d()**: This can be used for 3D translation.

- **Rotation - Transform**: It helps to rotate an HTML element. To apply this transformation we use the transform rule. This rule takes multiple transformation actions space separated. To rotate the element along the X, Y or Z axis we can use rotate().

We can rotate our elements by using the following ways:

1. **rotateX()**: It rotates the element along the X axis.
 2. **rotateY()**: It rotates the element along the Y axis.
 3. **rotateZ()**: It rotates the element along the Z axis (out of the screen).
 4. **rotate()**: This can be used for a 2D rotation.
 5. **rotate3d()**: This can be used for 3D rotation.
- **Scale - Transform**: It helps to resize an HTML element either small or large.

We can resize our elements by using the following ways:

1. **scaleX()**: It resizes the element along the X axis.
 2. **scaleY()**: It resizes the element along the Y axis.
 3. **scaleZ()**: It resizes the element along the Z axis.
 4. **scale()**: This can be used for a 2D scaling.
 5. **scale3d()**: This can be used for 3D scaling.
- **Skew - Transform**: If you want to tilt or stretch an element, you can use skew().

We can skew our elements by using the following ways:

1. **skewX()**: It tilts the element along the X axis.
2. **skewY()**: It tilts the element along the Y axis.
3. **skew()**: This can be used for a 2D skewing.
4. **skew3d()**: This can be used for 3D skewing.

Transition Properties

CSS transitions enable web developers to control the smooth transition between two states of an element. For instance, when a user hovers over a button, the background color of the element can change seamlessly using CSS selectors and pseudo-classes.

- **transition-duration::** Defines the length of time a transition animation should take.
- **transition-delay::** Defines the amount of time to wait before the transition animation starts.
- **transition-timing-function::** Defines the speed curve of the transition animation.

We can define the speed curve of the transition animation by using the following ways:

1. **ease**: This is a slow start and fast end.
2. **ease-in**: This is a slow start and fast end.
3. **ease-out**: This is a fast start and slow end.
4. **ease-in-out**: This is a slow start, fast middle and slow end.
5. **linear**: This is a slow start and fast end.

6. **steps-end**: This is with a constant start and fast end.
7. **steps-start**: This is with a fast start and constant end.
8. **cubic-bezier()**: This can be used to define the custom curve of the transition animation.

CSS animations involve several key properties:

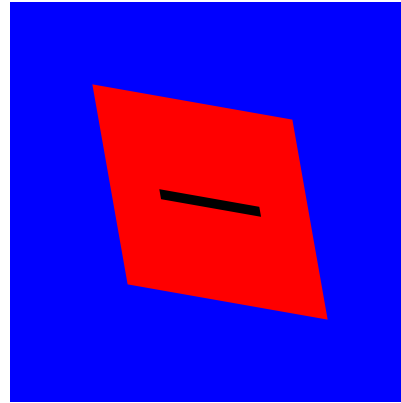
1. **@keyframes::** Defines the style of the element at different points of time.
2. **animation-name::** Defines the name of the animation.
3. **animation-duration::** Defines the length of time the animation should take.
4. **animation-timing-function::** Defines the speed curve of the animation.
5. **animation-delay::** Defines the amount of time to wait before the animation starts.
6. **animation-iteration-count::** Defines the number of times the animation should be played.
7. **animation-direction::** Defines whether the animation should be played forwards, backwards or in alternate cycles.

We can define the direction of the animation by using the following ways:

1. **Normal**: The animation will be played forwards.
2. **Reverse**: The animation will be played backwards.
3. **Alternate**: The animation will be played forwards then backwards.
4. **Alternate-reverse**: The animation will be played backwards then forwards.
8. **animation-fill-mode::** Defines how the animation should apply to the element once it has finished.

9. **animation-play-state::** Defines whether the animation is running or paused.

Transitions Example



Animations Example

