index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>CSS Animations</title>
</head>

<body>

    <h1>CSS Animations</h1>

    <p>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.</p>

    <p>To understand CSS animations completely we need to master these things:</p>

    <ol>
        <li>Transforms</li>
        <li>Transitions</li>
        <li>Keyframes</li>

    </ol>

    <h2>Transform Properties</h2>

    <ul>

        <li><strong>Transform</strong>: 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).</li> <br>

        <li><strong>Translate - Transform</strong>: 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().

            <ol>

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

                <li><strong>translateX()</strong>: It moves the element along the X axis.</li> <br>

                <li><strong>translateY()</strong>: It moves the element along the Y axis.</li> <br>

                <li><strong>translateZ()</strong>: It moves the element along the Z axis.</li> <br>

                <li><strong>translate()</strong>: This can be used for a 2D translation.</li> <br>

                <li><strong>translate3d()</strong>: This can be used for 3D translation.</li> <br>

            </ol>

        </li>

        <br>

        <li><strong>Rotation - Transform</strong>: 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().

            <ol>

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

                <li><strong>rotateX()</strong>: It rotates the element along the X axis.</li> <br>

                <li><strong>rotateY()</strong>: It rotates the element along the Y axis.</li> <br>

                <li><strong>rotateZ()</strong>: It rotates the element along the Z axis (out of the screen).</li> <br>

                <li><strong>rotate()</strong>: This can be used for a 2D rotation.</li> <br>

                <li><strong>rotate3d()</strong>: This can be used for 3D rotation.</li> <br>

            </ol>

        </li>

        <li><strong>Scale - Transform</strong>: It helps to resize an HTML element either small or large.

            <ol>

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

                <li><strong>scaleX()</strong>: It resizes the element along the X axis.</li> <br>

                <li><strong>scaleY()</strong>: It resizes the element along the Y axis.</li> <br>

                <li><strong>scaleZ()</strong>: It resizes the element along the Z axis.</li> <br>

                <li><strong>scale()</strong>: This can be used for a 2D scaling.</li> <br>

                <li><strong>scale3d()</strong>: This can be used for 3D scaling.</li> <br>

            </ol>

        </li>

        <br>

        <li><strong>Skew - Transform</strong>: If you want to tilt or stretch an element, you can use skew().

            <ol>

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

                <li><strong>skewX()</strong>: It tilts the element along the X axis.</li> <br>

                <li><strong>skewY()</strong>: It tilts the element along the Y axis.</li> <br>

                <li><strong>skew()</strong>: This can be used for a 2D skewing.</li> <br>

                <li><strong>skew3d()</strong>: This can be used for 3D skewing.</li> <br>

            </ol>

        </li>

    </ul>

    <h2>Transition Properties</h2>

    <p>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.</p>

    <ul>

        <li><strong>transition-duration:</strong>: Defines the length of time a transition animation should take.</li>
        <br>

        <li><strong>transition-delay:</strong>: Defines the amount of time to wait before the transition animation
            starts.</li> <br>

        <li><strong>transition-timing-function:</strong>: Defines the speed curve of the transition animation.

            <ol>

                <h4>We can define the speed curve of the transition animation by using the following ways:</h4>

                <li><strong>ease</strong>: This is a slow start and fast end.</li> <br>

                <li><strong>ease-in</strong>: This is a slow start and fast end.</li> <br>

```

```html
                <li><strong>ease-out</strong>: This is a fast start and slow end.</li> <br>

                <li><strong>ease-in-out</strong>: This is a slow start, fast middle and slow end.</li> <br>

                <li><strong>linear</strong>: This is a slow start and fast end.</li> <br>

                <li><strong>steps-end</strong>: This is with a constant start and fast end.</li> <br>

                <li><strong>steps-start</strong>: This is with a fast start and constant end.</li> <br>

                <li><strong>cubic-bezier()</strong>: This can be used to define the custom curve of the transition
                    animation.</li> <br>

            </ol>

        </li> <br>

    </ul>

    <ol>

        <h4>CSS animations involve several key properties:</h4>

        <li><strong>@keyframes:</strong>: Defines the style of the element at different points of time.</li> <br>

        <li><strong>animation-name:</strong>: Defines the name of the animation.</li> <br>

        <li><strong>animation-duration:</strong>: Defines the length of time the animation should take.</li> <br>

        <li><strong>animation-timing-function:</strong>: Defines the speed curve of the animation.</li> <br>

        <li><strong>animation-delay:</strong>: Defines the amount of time to wait before the animation starts.</li> <br>

        <li><strong>animation-iteration-count:</strong>: Defines the number of times the animation should be
            played.</li> <br>

        <li><strong>animation-direction:</strong>: Defines whether the animation should be played forwards, backwards
            or in alternate cycles.

            <ol>

                <h4>We can define the direction of the animation by using the following ways:</h4>

                <li><strong>Normal</strong>: The animation will be played forwards.</li> <br>

                <li><strong>Reverse</strong>: The animation will be played backwards.</li> <br>

                <li><strong>Alternate</strong>: The animation will be played forwards then backwards.</li> <br>

                <li><strong>Alternate-reverse</strong>: The animation will be played backwards then forwards.</li>

            </ol>

        </li> <br>

        <li><strong>animation-fill-mode:</strong>: Defines how the animation should apply to the element once it
            has finished.</li> <br>

        <li><strong>animation-play-state:</strong>: Defines whether the animation is running or paused.</li> <br>

    </ol>

    <h2>Transitions Example</h2>

    <div id="parent">

        <div id="red">

            <div id="line"></div>

        </div>

    </div>

    <h2>Animations Example</h2>

    <div id="box">

    </div>

</body>

</html>
```

style.css

```css
* {
    letter-spacing: 0.5px;
}

#parent {
    width: 200px;
    height: 200px;
    background-color: 🟦 blue
    display: flex;
    justify-content: center;
    align-items: center;
    margin: 0 auto;
}

/* Transform Properties */

#red {
    width: 100px;
    height: 100px;
    background-color: 🟥 red
    display: flex;
    justify-content: center;
    align-items: center;
    transform: translate(10px, 10px);
    transform: rotate(50deg);
    transform: scale(1.5, 1.5);
    transform: skew(10deg, 10deg);
    transition-duration: 1s;
    transition-delay: 0.5s;
    transition-timing-function: ease-in;
}

#line {
    width: 50px;
    height: 5px;
    background-color: ⬛ black
```

```css
}

h1,
h2 {
    text-align: center;
}

p,
li {
    font-size: 18px;
}

#red:hover {
    background-color: 🟩 green
    cursor: pointer;
}

#box {
    width: 100px;
    height: 100px;
    background-color: 🟥 red
    margin: 0 auto;
}

/* Animation Properties */

#box:hover {
    animation-name: change-color;
    animation-duration: 2s;
    animation-delay: 1s;
    animation-iteration-count: infinite;
    animation-direction: alternate;
    animation-timing-function: ease-in-out;
    cursor: pointer;
}

@keyframes change-color {

    0% {
        background-color: 🟥 red
    }

    50% {
        background-color: 🟩 green
    }

    100% {
        background-color: 🟦 blue
    }
}
```

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:

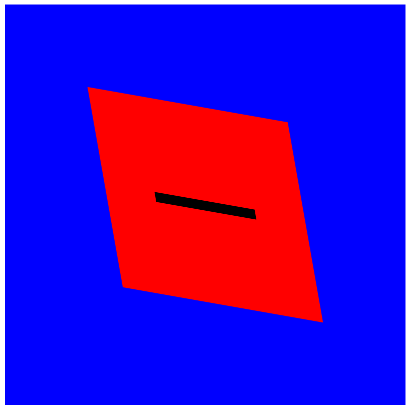
1. **@keyframes:**: Defines the style of the element at different points of time.
2. **animation-name:**: Defines the name of the animation.
3. **animation-duration:**: Defines the length of time the animation should take.
4. **animation-timing-function:**: Defines the speed curve of the animation.
5. **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