Lesson:

Transform

PW SKILLS

Topics Covered

- CSS transition
- Translate
- Scale
- Rotate
- Skew
- matrix

CSS transition:

- This transform property allows you to translate, rotate, scale, and skew elements.
- Transformation is an effect that is used to change shape, size, and position.

Translate:

The translate property is used to move the element alone on the x-axis and y-axis.

Syntax:

```
translate(x, y)
```

The **x** and **y** parameters specify the distance that the element should be moved along the X and Y axis, respectively. They can be specified in various units, such as “px”, “em” or “%”.

By using **translateX()** we can move the element on the **x-axis** and by using **translateY()** we can move the element on the **y-axis**.

Here are some examples of how to use translate() in CSS:

1) Move the element 50 pixels to the right and 25 pixels down:
Index.html:

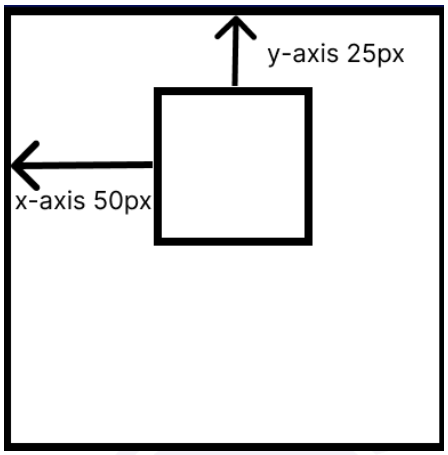
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="./style.css" />
    <title>Document</title>
  </head>
  <body>
      <div class="box"></div>
  </body>
</html>
```

Note: We will use the same HTML call for all translation examples.

Style.css

```
.box {
  border: solid;
  height: 50px;
  width: 50px;
  transform: translate(50px, 25px);
}
```

Browser output

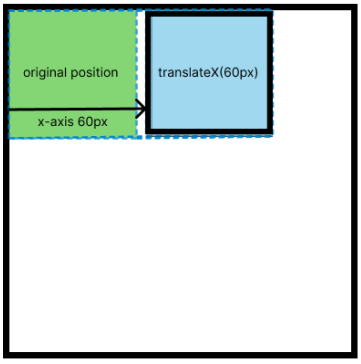


2) Move the element 60 pixels to the right using **translateX():**

style.css:

```
.box {
  border: solid;
  height: 50px;
  width: 50px;
  transform: translateX(60px);
}
```

Browser output- with some required information for better understanding

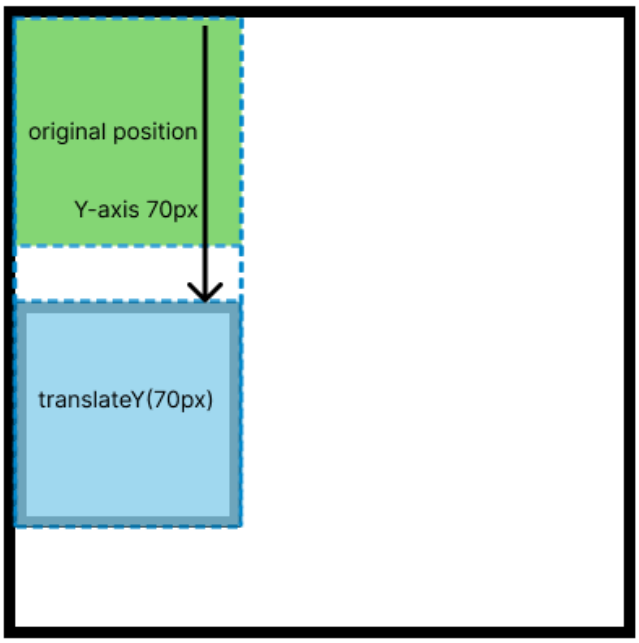


3) Move the element 70 pixels to the bottom using **translateY():**

Style.css

```
.box {
  border: solid;
  height: 50px;
  width: 50px;
  transform: translateY(70px);
}
```

Browser output with some required information for better understanding



Scale:

It is used to change the width and height of an element.

Syntax:

```
transform: scale(x, y);
```

where **x** and **y** are the scaling factors. A value of **1 represents** the **original size of the element**. Values **greater than 1** will **scale the element up**, while values **between 0 and 1** will **scale the element down**.

By using **scaleX()** we can **change the width** of an element and by using **scaleY()** we can **change the height** of an element.

Here are some examples of how to use scale() in CSS:

Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="./style.css" />
    <title>Document</title>
  </head>
  <body>
    <img class="image" src="https://picsum.photos/200" alt="image" />
  </body>
</html>
```

Note: We will use the same HTML code for all the scale() examples.

1) Scale an element to 120% of its original size in the horizontal direction and 80% of its original size in the vertical direction.

Style.css:

```
.image{  transform: scale(1.2, 0.8); }
```

Browser output:

before

after

2) Scale an element to 150% of its original size in both the horizontal and vertical directions:

Style.css:

```
.image{  transform: scale(1.5); }
```

Browser output:

before **after**

3) Increase the height and decrease the width of elements using scaleY() and scaleX():

style.css:

```
.image{
    transform: scaleX(1.2);
    transform: scaleX(0.5);
}
```

Browser output:

before **after**

Rotate:

It is used to rotate the element on the basis of an angle.

Syntax :

```
transform: rotate(angle);
```

where **angle** is the amount of **rotation in degrees**. **Positive values rotate** the element **clockwise**, while **negative values rotate** it **counterclockwise**.

Here are some examples of how to use rotate() in CSS;

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./style.css">
    <title>Document</title>
</head>
<body>
    <img class="image" src="https://picsum.photos/200" alt="image"></img>
</body>
</html>
```

Note: We will use the same HTML code for all rotate() examples.

1) Rotate an element 45 degrees clockwise:

Style.css:

```
.image { transform: rotate(45deg); }
```

Browser output:

2) Rotate an element 45 degrees anticlockwise:

style.css:

```
.image {  transform: rotate(-45deg); }
```

Browser output:

Skew:

It specifies the skew transformation along the X and Y axis corresponding to the skew angles.

Syntax :

```
transform: skew(x-angle, y-angle);
```

where **x-angle** and **y-angle** are the **angles of skew in degrees**. **Positive values skew** the element **in the direction of the positive axis**, while **negative values skew** it in the **opposite direction**.

By using **skewX()** we can skew the element along the x-axis and by using **skewY()** we can skew the element along the x-axis.

Here are some examples of how to use skew() in CSS;
Index.html

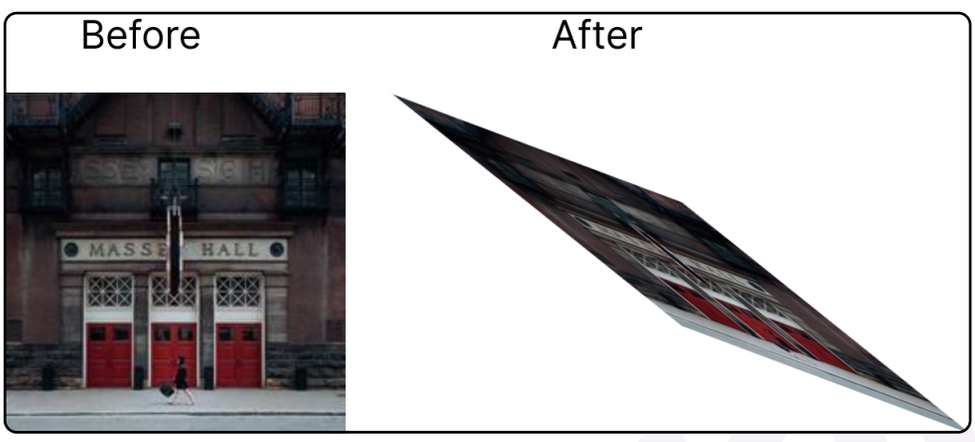
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="./style.css">
    <title>Document</title>
</head>
<body>
    <img class="image" src="https://picsum.photos/200" alt="image"/>
</body>
</html>
```

Note: We will use the same HTML code for all skew examples.

1) skew an element 45 degrees in the X direction and 25 degrees in the Y direction.
style.css:

```
image{
 transform : skew(45deg,25deg)
}
```

Browser output:

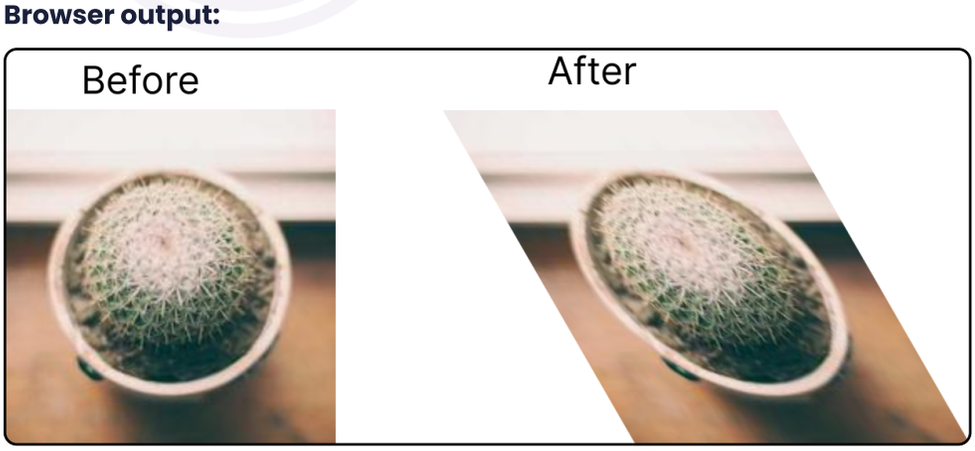


2) Skew an element 30 degrees in the X direction using **skewX():**

style.css

```
image{
 transform: skewX(30deg)
}
```

Browser output:



3) Skew an element 45 degrees in the Y direction using **skewY():**

style.css

```
image{
 transform: skewY(45deg)
}
```

Browser output:

Matrix:

The matrix() defines a homogeneous 2D transformation matrix.
It takes six parameters, containing mathematical functions that allow you to rotate, scale, move (translate), and skew elements.

Syntax :

```
transform: matrix(a, b, c, d, tx, ty);
// matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(),
translateY())
//
```

where **a, b, c, d, tx,** and **ty** are the values of the matrix elements. These values represent the following transformation matrix:

```
| a  c  tx |
| b  d  ty |
| 0  0  1  |
```

Here's an explanation of what each matrix element does:

- **“a”** represents the **horizontal scaling** of the element.
- **“b”** represents the **vertical skewing** of the element.
- **“c”** represents the **horizontal skewing** of the element.
- **“d”** represents the **vertical scaling** of the element.
- **“tx”** represents the **horizontal translation** of the element.
- **“ty”** represents the **vertical translation** of the element.

Here are some examples of how to use matrix() in CSS:

1) Scale the element in horizontal and vertical directions:

Style.css

```
 .image{
 transform: matrix(2, 0, 0, 2, 0, 0);
}
```

Browser output:

Before

After