10/01/23

# * CSS Gradients

CSS gradients let you display smooth transition b/w two or more specified colors.

There are three type

- → Linear Gradients [goes down / up / left / right / diagonally]
- → Radial Gradients [defined by their center]
- → Conic Gradients [rotated around a center point]

## ① Linear Gradients

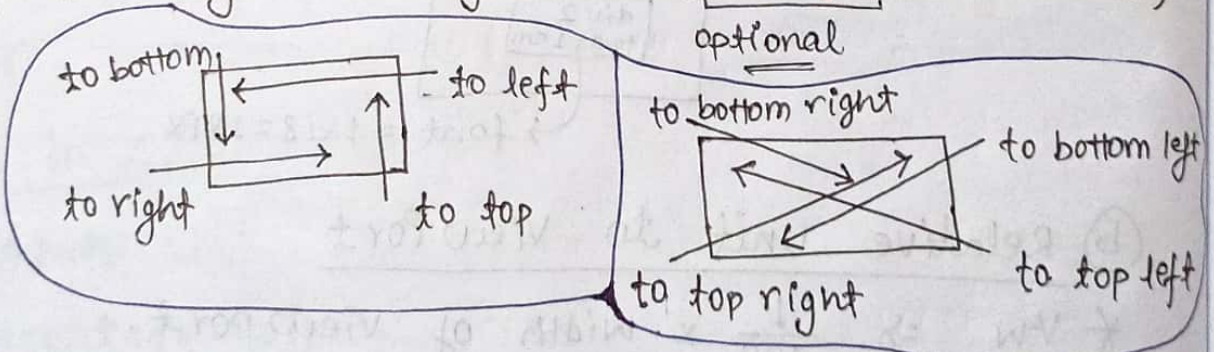
To create a linear gradients you must define at least two color stops. color stops are the colors you want to render smooth transition among.

You can also set a starting point and a direction or angle along with the gradient effect

Ex ⇒ background-image : linear-gradient (direction, color1, color2, color3 ---);

background-image : linear-gradient ( to right, red, blue);

optional




background-image : linear-gradient ( 210 deg, red, green, orange);

background-image : linear-gradient ( to bottom, rgba(255,0,0,0), rgba(0,0,255,1)

# By default direction :- to bottom ↓

## ② Radial Gradient

A radial gradient is defined by its center.
To create a radial gradient you must also define at least two color stops.

Snytax

```
background-image : radial-gradient ( shape size at
                   position , color1 , color2 ---);
```

```
background-image : radial-gradient (red, blue, green);
```

By default shape :- ellipse

## ③ Conic Gradient

A conic gradient is a gradient with color transition rotated around a center point.
To create a conic gradient you must define atleast two colors.

Syntax

```
background-image : conic-gradient (color1, color2--);
```

⇒ By default, angle is 0deg and position is center.
⇒ If no degree is specified, the colors will be spread equally around the center point.

## * CSS Shadow Effects

With CSS you can add shadow to text and to element.

- text-Shadow
- box-Shadow

## ① Text Shadow

The CSS text-shadow property applies shadow to text.

Syntax

```
text-shadow : value1 value2 value3
              horizontal shift   vertical shift   Blur Effect
```

Ex ⇒

```
p {
    text-shadow : 3px 3px 3px red, 5px 5px 5px
                  blue;
}
```

NOTE ⇒ we can apply more that one shadow in text.

⇒ By default shadow color is same as text color but we can apply different as per conventent.

## ② Box Shadow

The CSS box-shadow property is used to apply one or more shadow to an element.

Syntax

```
box-shadow : value1 value2 value3 ;
             horizontal shift   vertical shift   Blur Effect
```

Ex ⇒

```
.box {
    box-shadow : -10px 5px 5px green,
                 15px -5px 15px red;
}
```

NOTE ⇒ By default shadow color is same as text color.

Q ⇒ How can we add boarder using shadow?

we can add boarder using shadow by putting horizontal shift value and vertical shift value as 1PX.

⇒ multiple shadow can be added using comma.

⇒ color of the shadow can be changed.

⇒ Spread radius can be changed.

## * CSS Dimension Property

⇒ Height

⇒ width

⇒ max-height ⇒ content ke sath div ki height inc krega max height tk fir overflow ho jaye ga content.

⇒ min-height ⇒ min height tk div rhega, content km ho, tb v aur content badh ne par div ka height badhega.

⇒ Max-width

⇒ Min-width

## * Overflow Property

This property specifies whether to clip content or to add scrollbar when an element's content is too big to fit in a specified area.

NOTE ⇒ The overflow property only works for block elements with a specified height.

Syntax

overflow : visible / hidden / clip / scroll / auto

Values in overflow

# * CSS Position Property

The position property specifies the type of positioning method used for an element.

## ① Static (By default)

An element with position: static; is not positioned in any special way it is always positioned to the normal flow of page.

## ② Relative

An element with position: relative; is position relative to its normal position.

Setting the top, right, bottom, and left properties of a relative-positioned element will cause it to be adjusted away from its normal position.

```
position : relative;
left     : 15PX;
top      : 10PX
```

## ③ Fixed

An element with position: fixed; is positioned relative to the viewport which means it always stays in the same place even if the page is scrolled.

The top, right, bottom and left properties are used to position the element.

```
position: fixed;
```

## ④ Absolute

An element with position: absolute, is positioned relative to the nearest positioned ancestor.

However, if an absolute positioned element has no positioned ancestors, it uses the document body and move along with the page scroll.

NOTE:) Absolute positioned element are removed from the normal flow, and can overlap element.

```
position: absolute;
top  : 100px
left : 15px
```

⑤ sticky

The element is positioned based on the user's scroll position.

A sticky element toggles b/w relative & fixed depending on the scroll position.

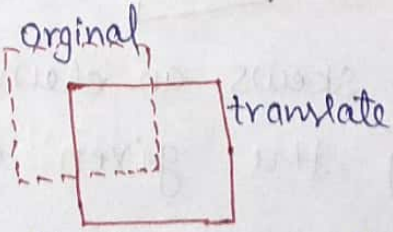
```
position: sticky;
top  : 50px;
left : 10px
```

* 2D Transform

Transform property you can use following method-

- translate ()
- rotate ()
- Scale X ()
- Scale Y ()
- Scale ()
- Skew X ()
- Skew Y ()
- Skew ()
- matrix ()

① translate

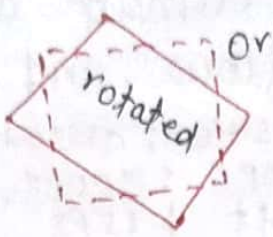



The translate method moves an element from its current position according to its given parameters

Syntax

```
translate ( value 1, value 2);
```

## ② rotate()




The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

Ex ⇒
```
div {
    transform: rotate(20deg);
}
```

## ③ Scale()


The scale() method increases or decreases the size of an element according to the given parameters for the width and height.

Ex ⇒
```
div {
    transform: scale(2, 3);
}
```
(2 → scaleX, 3 → scaleY)

NOTE ⇒ We can also use scaleX & scaleY seperately

## ④ Skew()


The skew() method skews an element along with X and Y-axis by the given angle.

Ex ⇒
```
div {
    transform: skew(20deg, 10deg);
}
```
(20deg → skewX, 10deg → skewY)

We can also use skewX & skewY seperately.

⑤ matrix()

The matrix() method combines all the 2D transform method into one.

The parameters are as follow:

matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())

Ex⇒
```
div {
    tranform: matrix( 1, -0.3, 0, 1, 0, 0);
}
```

* 3D Transforms

It works on the z-axis like 2D transform which works on x-axis and y-axis. Not possible to explain in 2D transform.

Ex ⇒
```
.transform-inclass {
    transform: persepective (15PX)
               translateZ (-10PX);
}
```

```
#card {
    transform: persepective (350PX)
               Scale Z (40deg)
               rotate X (45 deg)
               rotate Y (30 deg);
}
```