Date .....

Gradient :-

It set colors as a background in any container or on content also like background-clip.

* Types :-

(i). Linear gradient :-

eg:-

```
div {
Bg-image : linear-gradient
(to right, red, yellow);
```

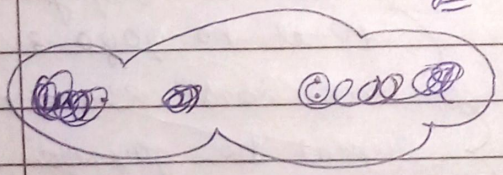
↳ Kithne tariko se (Bg-image & linear-gradient(value);) yahan pr value de sakte hai. ⇒

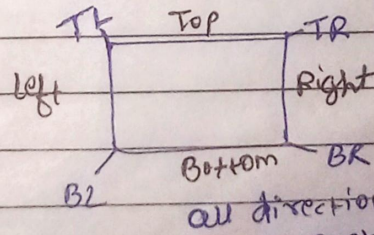
(i). Bgimage : linear-gradient(more two or one color name);

eg:- (red, yellow, pink, blue);

(ii). " : " (direction, color names);

eg:- (to right/left/top/Bottom, color);
(to Bottom Right etc./to Bottom left





all direction supported.

(iii). By angle :-

" : " (45/90/-45/180 deg, colors);

eg: (45 deg, red, yellow);

(iv). By percentage :-

» : » (to right, red, yellow, green 50%);

50% container pr green color lg jayega and bache hue 50% red, yellow me bat jayega.

(v). Transparency :-

» : » (to right, rgba(255,0,0,1), rgba(255,0,0,1));

2. Repeating-linear gradient :-

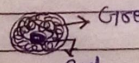
(i) » : Repeating-linear-gradient (to right, red, yellow 10%, green 20%);

10%+20% and red ka automatic set ho jayega to total ho gaya 30%. and baki space me repeat ho jayega yahi kino colour Right hi teraf.

3. Radial-gradient :-

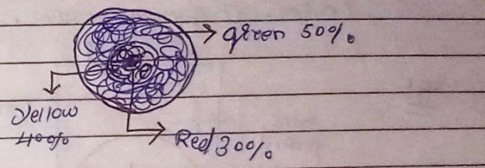
(i). » : Radical-gradient (red, green);

output :-



(ii) By - percentage :-

» : » (red 30%, yellow 40%, green 50%);

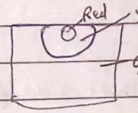


(iii). To make circle shape instead of ellipse :-

» : » (circle, red 30%, yellow 40%, green 50%);

↳(a) position of circle :-

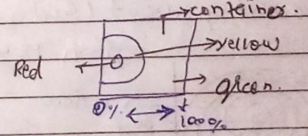
» : » (circle at top, colors);



/Bottom
/left
/Right
/TOP Right /TOP left /Bottom right/ Bottom left.

(b). circle position by % :- (x-axis)

» : » (circle at 0%, colors);



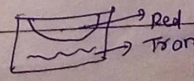
(c). circle position by % :- (both x and y axis).

» : » (circle at 25%, 30%, colors);

x-axis ↳ y-axis

(iv) (d). Transparency :-

» : » (ellipse at top, red, transparent);

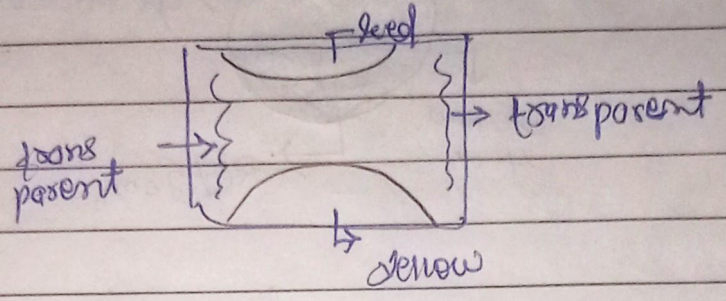


Date .....

(v). two radial gradient at one time :- (for more than one)

)) :)) (elipse at top , red , transparent),
(elipse at Bottom, yellow, transparent);

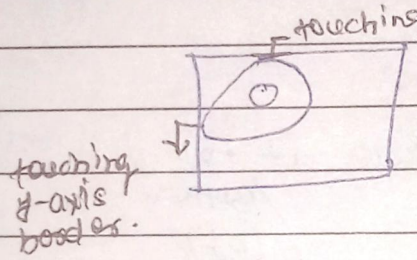
eg:-



(vi). size of elipse :-

)) :)) (closest-side at 30% 40% , red, yellow, green);

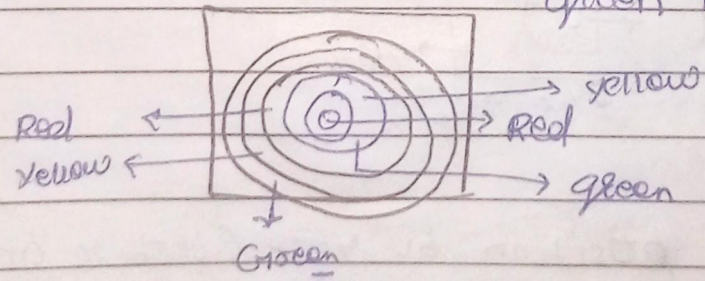
eg:-

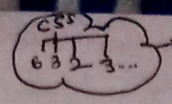


↳ closest-side / closest-corner / farthest-side / farthest-corner.

(vi). Repeating-radial gradient :-

)) : Repeating-radial-gradient (red, yellow 15%, green 15%);





Date .....

overlay effect :-

Ex:- (1).

<H1> My developers </h1>
<h2> it's overlay effect </h2>

.

(2) yahan pr, img class sabse niche tha to jab absolute diya to apni jagah se utr gaya and overlay aagaya to overlay ko v absolute diya to text align niche.

CSS =>

```
.container {
    height: 100vh;
    Bg-color: Dark_blue;
    display: flex;
    Justify-content: centre;
    align-item: Centre;
    → position: Relative;
}
```

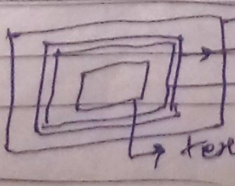
(3). Iss tarah img and overlay dono upr the background se and niche text tha to hamne text ko z-index 2 diya taaki sabse upar dikhe and overlay ko transparent kiya taaki img v dikhe

```
.container .overlay {
    height: 400px;
    width: 600px;
    bg-img: linear-gradient
    (to bottom, red, blue);
    → position: absolute;
    opacity: 0.7;
}
```

```
.container .img {
    height: 600px;
    width: 400px;
    bg-img: url('');
    bg-size: cover;
    bg-position: centre;
    → position: absolute;
}
```

```
.text {
    z-index: 2
    color: white;
    font-size: 30px;
}
```

output:-



Date .....

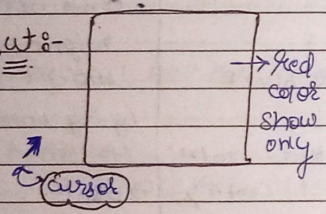
(2). overlay effect example-2 → text coming from side on hover.

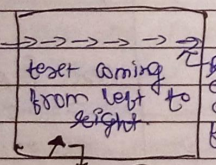
eg. text coming from left to right :-

```
<div class="container">
<img class="img">
<div class="overlay">
  <h1> text coming from left to right </h1> </div>
</div>
```

```
.container {
  height: auto;
  width: 25%;
  position: relative;
  bg-color: red;

.img {
  height: 100%;
  width: 100%;
  display: block;

.overlay {
  height: 100%;
  width: 0%  major cause.
  position: absolute;
  top: 0px;
  left: 0px
  bg-color: yellow;
  overflow: hidden;
  transition: width 0.5s;

.container:hover .overlay {
  width: 100%;
```

output :-





yellow color coming from left to right b/c width increases on hover.

↳ overlay me opacity ko use kr sakte hai for uske liye overflow: hidden ko hata denge. and height width sab 100% and hover me opacity k

Date .....

Note:- ① Jahan pr hamne width ko zero krke use hover me 100% kr diya to jab v cursor jata raha main pr tab overlay pr hover effect transition property ki wajah se 0.5s me lg raha tha and hame aisa lag raha tha ki text left se slide hokr aa raha hai jabki bss waha width increase ho raha tha.

② To isi tarah se agr ham width ki jagah height ko 0px krke fr hover se badhayenge to aisa lagega ki text top se bottom ki taraf slide ho raha hai.

3. (i) third example of, overlay effect from left to right like second example.

```
<div class="container">
  <img src="..." class="image">
  <div class="overlay">
    <div class="text"> Left to Right </div>
  </div>
</div>
```

```
.container { width: 200px;
  height: 200px;
  position: relative; }

.image { width/height: 200px;
  display: block; }

.overlay { position: absolute;
  transition: all 0.3s ease;
  opacity: 0;
  bg-color: gray; }

.text { color: white; font-family: serif;
  position: absolute; top: 50%; left: 50%;
  transform: translate(-50%, -50%); }

.container:hover .overlay {
  opacity: 1;
  width: 100%; }
```

overlay me left

~~.overlay { height: 100%;~~
~~width: 0;~~
~~top: 0;~~
~~position: 0;~~ }

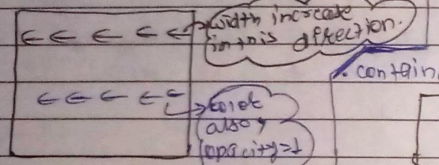
Date .....

(3)(iv). overlay from right to left :-

```
<div class="container">
  <img class="image">
  <div class="overlay overlay-right">
    <div class="text">
      Right to left
    </div>
  </div>
</div>
```

rest of code is same,

```
.overlay-right {
  height: 100%;
  width: 0;
  top: 0;
  Right: 0;
}
```



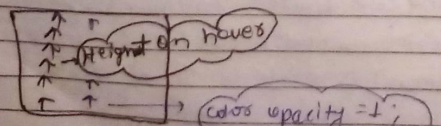

```
.container:hover .overlay-right {
  width: 100%;
  opacity: 1;
}
```

3(iii). From bottom to top;

```
<div class="overlay overlay-bottom">
```

```
.overlay-bottom {
  width: 100%;
  height: 0%;
  bottom: 0;
  left: 0;
}

.container:hover .overlay-bottom {
  height: 100%;
  opacity: 1;
}
```

output:-



Spiral

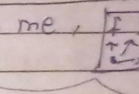
(3)(iv). From cross :-

```
.overlay-cross {
  width: 0;
  height: 0;
  top: 0;
  left: 0
}
```

↳ Right cross ke liye (right:0);
↳ bottom cross ke liye (bottom:0);

values = {(top:0, left:0;) (top:0; Right:0),
(bottom:0,») (bottom:0, »)}

↳ bottom:0; left:0; me,



To reach effect layega

```
.container:hover .overlay-cross {
  height/width: 100%;
}
```

(3)(v). fade effect :-

```
.overlay-fade {
  width/height: 100%;
  top:0; left:0;
  bg-color: red;
}

.container:hover .overlay-fade {
  opacity: 0.7;
}
```

Spiral

Date .....

Transform - property :-

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

(1). Translate :-

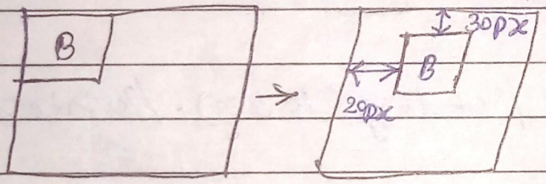
pehle wale div ke relative age home usko kahi° bhejna ho to ham isko use krte hai°.

eg:-

```
<div>
    <div>B</div>
</div>
```

```
div {
    transform : translate
    (20px) or (20px, 30px)
}
```

(x-axis) (y-axis)

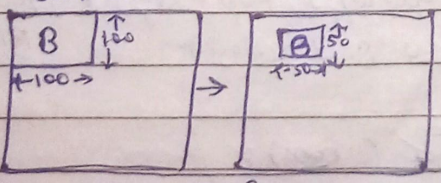


(2). Scale :-

To decrease or increase the size of div by its previous size.

```
<div>
    <div>B</div>
</div>
```

output :-




```
div {
    width : 100px;
    height : 100px;
    transform : scale (0.5 or 50%);
}
```

Is se width 50px and height v 50px ho jayega.

(i) scalex (2) {x-direction me inc/dec hoga} (ii). scaley (1) {south me v use hoga}

(iii) scale (2);

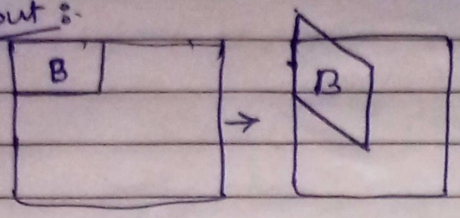
(3). Skew :-

Kisi particular angle me skew kr sakte hai.

output iska aayega Date.....

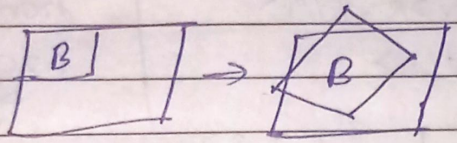
(i) transform : skew (20 deg, 10 deg) or (0 deg, 20 deg) or (20 deg).
x-axis's y-axis's

output :-



(4) Rotate :-

(i) transform : rotate (20 deg) → z - direction



(ii) transform : rotatex / rotatey (10 deg) . / rotate z.

Notes all together :-

transform : rotate (20 deg) scale (1.2) skew (30 deg)
translate (45px);

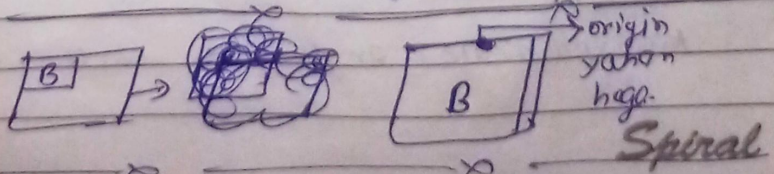
Transform - origin :- (By default (centre))

ye batata hai ki element kahan se scale, skew, rotate and translate hoga.

eg:- Top, Bottom, Top-left, Top-Right, Bottom left/Right
Right/left. (all 8 direction).

eg:-

```
transform : scale (2);
transform - origin : TOP ;
                     ↳ left, centre, Right, length, %.
```



Spiral

Date .....

15. Animations :-

↳ koi naam variable ki tarah.

(1). Animation-name : ____ ;

(i) box

(ii) container

(2). Animation duration : ____ ;

(i). s

(ii). ms.

(3). Animation-timing-function : ____ ;

(i) ease = slow start + fast + slow end

(ii). ease-in = slow start + linear.

(iii). ease-out = linear + slow end

(iv). ease-in-out = slow start + linear + slow end.

(v). Linear = same speed.

4. Animation-delay : ____ ;

(i) s

(ii) ms.

5. Animation-iteration-count : ____ ;

(i). 1

(ii) (2)

(iii) 2 (up to).

(6). Animation-direction : ____ ;

default (i) normal : start → forward. end

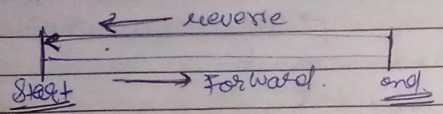
(ii) reverse : start ← reverse. end

(iii) alternate : start → forward end ← reverse

Spiral

Date .....

(iv). alternate - reverse :



(7). Animation - play - state : _____ ;

↳ hover me like same hai jab hover ho pause ho jaaye animations

(i). running (default).
(ii). pause.

** Shortcut :-

↳ Animation : Animation name | Anim. duration | Anim. - tim-fn. | Anim. delay

Anim. - iteration - count | Anim. - direction | Anim. play - state. ;

How to implement :-

(1). using from and to →

@ keyframes (Animation-name) { identifier

from { // CSS code } start

to { // CSS code } End.

}

Spiral

Date .....

(2). using percentage % :-

@ keyframes (Animation-name) { identifier

start { 0% { // CSS code } }

middle { 50% { // CSS code } }

end { 100% { // CSS code } }

Examples :-

```
// Boiler plate
* {
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}
// body
body {
    width: 100vw;
    height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
}
```

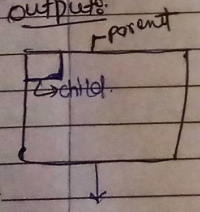
```
// parent
.parent {
    width: 1100px;
    height: 600px;
    bgcolors: green;
    border: 3px solid black;
    position: relative;
}
// child
.child {
    width: 150px;
    height: 150px;
    bg color: violet;
```

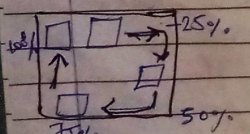
Spiral

Date .....

```
animation-name: box;
animation-duration: 3s;
animation-timing-function: ease-in-out;
animation-delay: 1s;
"  - iteration-count: infinite;
"  - direction: alternate;
position: absolute;
}

@keyframes box {
    0% {
        left: 0;
        top: 0;
    }
    25% {
        left: calc(100% - 150px);
        top: 0;
    }
    50% {
        left: calc(100% - 150px);
        top: calc(100% - 150px);
    }
    75% {
        left: 0;
        top: calc(100% - 150px);
    }
    100% {
        left: 0;
        top: 0;
    }
}
```

output:





Date .....

Example-2 :-

```
// child
.child {
    animation: box 2s ease-in-out 1s infinite running;
    animation-fill-mode: backwards;
    position: absolute;
    left: 0;
    top: 0;
}

@keyframes box {
    from {
        left: -10px;
        top: -10px;
        bg color: blue;
    }

    to {
        left: calc(100% - 150px);
        top: 0; // rotate: 360deg;
        bg color: green;
    }
}
```

Date .....

(16). CSS units :-

yahan pr do types ke units hote hai°.

(1). Absolute units / fixed units →

(1). pixel →

- Logical pixel : jo screen hame visual hota hai and usme jo pixels hote hai° usko logical pixel kehte hai° and ye roughly sabhi screens ke same hi hote hai°.

- physical pixel → actual hardware pixels hote hai°.

- 1 pixel = $\frac{1}{96}$ inch.

(2). pc (picas) →

- 1pc = 12px

(3). pt (points) →

- 1pt = 1.33 pixels
- 12pt = 1pc = 12 pixels

(4) inch
(5) cm
(6) mm.

(2). Relative units →

(1). percentage (%) :

Spiral

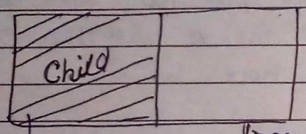
Date .....

- ye parent se relative hota hai.

eg:-
```
<div class="parent">
    <div class="child"></div>
</div>
```

```
.parent{
    width: 200px;
}
.child{
    width: 50%
}
```

output:-

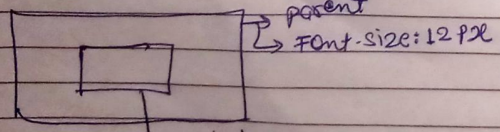


→ child
→ parent
→ 200px
~~200~~ / 2 × 50% / ~~100~~
= 100 px

- parent container body ka child hoga agr parent me 50% likha means body ka 50% parent ko mil jayega width.

(2). em :-
- ye v apne parent ke hi relative hota hai.

eg:-



→ 4em (4×12px = 48px)
→ Font-size).

- yahan v parent ka parent body hoga jisk by default 16px hota hai font-size.

sirf font-size ke liye hi test kiya hai rem , em ko parent and root element ke relative.

Date .....

sirf font size ke liye hi pata hai.

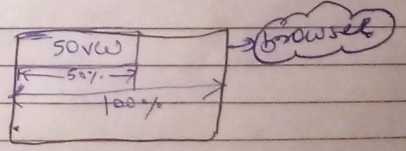
(3). rem :-
- ye root element ko relative maante hai

eg:-
```
html{
    Font-size: 16px (by default)
}
div{
    Font-size: 3rem (3×16=48px)
}
```
(hum change v kar sakte hai).
↳ font size)

(4). vw (viewport height) :-
- browser ka jo visible area hota hai uska width.

eg:-
```
div{
    width: 50vw (50% of viewport width)
    20vw (20% of viewport width)
    etc.
}
```

output:-



(5) vh (viewport height) →
- browser ka jo visible area hota hai uska height.

eg:- same like vw.