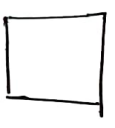
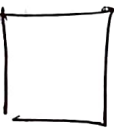
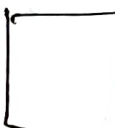

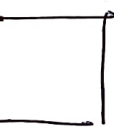






Grid & Responsiveness

	line1	line2	line3	line4
	col1	col2	col3	
line1 →				
line2 →				
line3 →				

✓ grid-area: — / — / — / —
 ↑ ↑ ↑ ↑
 grid row grid grid grid
 Start column row column
 Start end end
 ┌────────────────────────────────┐
 Single line giving
 values of rows &
 columns in grid ✓

Grid-area:-

The grid-area property specifies a particular area or set of rows & columns that a grid-item occupies. It is applied to the grid item itself with CSS eg:-

• item { grid-area: 1/2/3/3 }

Grid-template-area:

grid-template-area is the property used to name the rows & columns of a grid & to set its layout. It could look like this

• container {

display: grid;

grid-template-columns:
300px 300px 300px;

grid-template-rows:
250px 600px

grid-template-area:

"hd hd hd hd hd hd hd"

"sd sd sd main main main main main"

"ft ft ft ft ft ft ft ft";

• header {
grid-area: hd;
}

✓ naming
& adding.

for adding eg:-

#1 header {

grid-area: hd;

}

grid-template
-area
working

Row →	col 1	col 2
row 1	hd	hd
row 2	side	side
row 3	ft	ft

↑ This is how
naming is done
& when we give value
that occupies the
space named ✓

eg:- #header {
grid-area: hd;

↑

so this will
occupy hd space ✓

Q) If we have
rows & columns
defined in our grid
& can name them with
(grid-template-area)
& then give more data (box)
than defined what will
happen to the extra
boxes

→ They all will
squeeze below
the container

Advanced Grid Concepts :-

- 1) Fr unit \rightarrow ^{boxes are} equally divided using fr
eg: 1fr 1fr 1fr
- 2) Repeat Function \rightarrow repeat (3, 1fr)
 \uparrow 3 times \leftarrow equally divided
- 3) Grid - auto-rows: minmax()
 \uparrow
when there are unknown number of rows we use this property \leftarrow minimum & maximum size of the rows

Grid properties :-

- | | |
|---|--|
| <ul style="list-style-type: none">✓ justify-content✓ Align-content✓ justify-items✓ Align-items | <ul style="list-style-type: none">For parent<ul style="list-style-type: none">✓ justify-self✓ Align-self✓ place-items✓ place-selfFor grid-items (child)<ul style="list-style-type: none">horizontalvertically |
|---|--|

① justify-content
 \rightarrow content placed in main axis \longleftrightarrow
(Start, end, center, s-b, s-q, s-e)

② Align-~~content~~ items

\rightarrow content placed in vertical axis



- ✓ justify-content \nwarrow changes position with value without change in width.
- ✓ justify-items \nwarrow changes ~~height~~ width according to content
- ✓ justify-self \nwarrow change in specific box

Media Queries :- (Responsiveness)
css style in every device

- 1) "View port" — The area of the window in which web content can be seen.
- 2) Media queries are used to set different style rules for different devices or sized screens. we use breakpoints to set the condition of a media query.
The logic is
@media (feature: value)
- 3) Essentially, media query breakpoints are pixel values that a developer/designer can define in CSS. When a responsive website reaches those pixel values, a transformation (such as the one detailed above) occurs so that the website offers an optimal user experience.

For Multiple break points

we use and keyword

```
@media (min-width: 600px) and  
      (max-width: 900px) {
```

```
  container {
```

```
    // rules to change size.
```

```
  }
```

```
}
```

Nested Grids

Nesting CSS grids is simple & can be done simply by using the `display: grid` rule for both a parent & child element

```
* container {
```

```
  display: grid;
```

```
  // ...
```

```
}
```

```
# one {
```

```
  display: grid
```

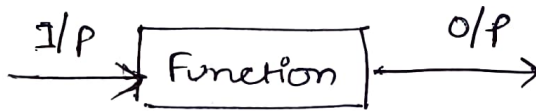
```
}
```

Animations In CSS



- ✓ CSS Style 1 to CSS Style 2
gradual change b/w them is
called Animations.

Functions in CSS :-



piece of code where we get output
when we give Input to it

eg:- `rgb ()`
 ↑
 i/p parameters
`repeat ()`
 ↑
 name

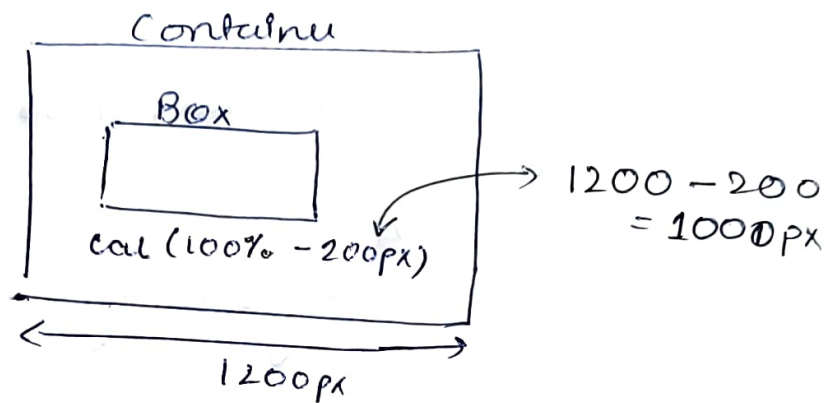
`Scale()`, `translate()`, `gradient`
`rgba()`, `url()`, `minmax()` etc etc.

Now, :-

Math Functions in CSS :-

① `calc()` (for width)

↖ To solve expression (+, -, *, / etc)
& use the value



`calc (parent value expression value)`
→ `calc (100% - 200px)`

② `min (-,-)` (for width)


③ `max (-,-)` (for width)

④ `minmax ($\frac{-}{\uparrow}$, $\frac{-}{\uparrow}$)` provides range b/w min & maximum
min to max

Variables in CSS :-

When you have a long block of code (layout) & you want to add styles, but when you want to change style then from variables you can change all value from single place.

✓ variables works only inside the Blocks. when declared ~~in~~ locally.

✓ When variable is declared in the :root  then ~~also~~ it will work

:root {

--dark-red: #981a2c;

}

↑ globally defined variable
can be used anywhere.

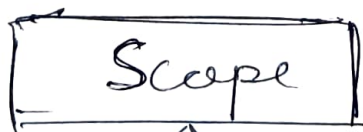
but.

container {

--dark-red: #981a2c;

}

↑ local variable
only be used
within blocks.



Global
variable

↑
in root
element

Local
variable

↑
in specific
element

✓ New.

Find diff b/w

```
:root {  
  }  
}
```

&

```
* {  
  }  
}
```

Animations :-

Transitions in CSS :-

- 1) Transition Timing (duration)
- 2) Transition function
- 3) Transition Delay (time after which transition occurs)
- 4) Transition property, (in which property we apply transition)

we can animate our objects using two properties.

Transition
property

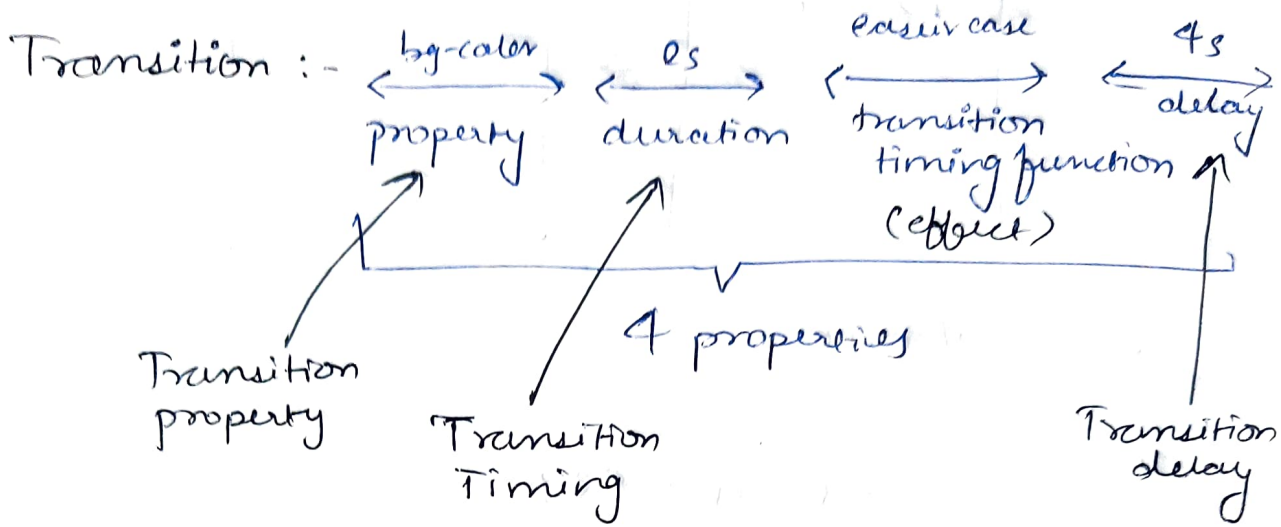
✓

Animation
& Key
frames

✓

① Transition property :-

Transition shorthand property :-



② Animation property :- & Keyframes :-

give properties inside the class you want to have animation

Then to run animation

@keyframes Name {

properties :-

animation-name
animation-duration
animation-iteration-count
animation-timing-function
animation-delay
animation-direction
animation-fill-mode

we write these properties inside the class CSS where we want animation to run.