28/4/21

* Sizing in CSS →

Sizing given to an element according to which it will be covering the space.

We have →

1. Px vs em vs rem
2. Percentage
3. vh
4. vw.

① Pixel (Px) → Our complete screen is made up of pixel and when we say an element will occupy 2px or 3px it will be occupying that much space only on the screen.

→ It is a a fixed unit.

② em - (Element) → Depends upon the parent element.

Ex →

1 em = Size of pixel of Parent.

if font size of Parent = 32 px

then font size of child will be

{1 em = 32 px.}

```
body {
    font-size : 32px;
}
.child { font-size : 1em; }
```

(3) Root - element (rem) → It depends on the root element or html element.

Ex → html {
font-size: 64px;
}

.size-heading {
font-size: 1rem;
}

→ So, this 1rem is now equal to 64px;

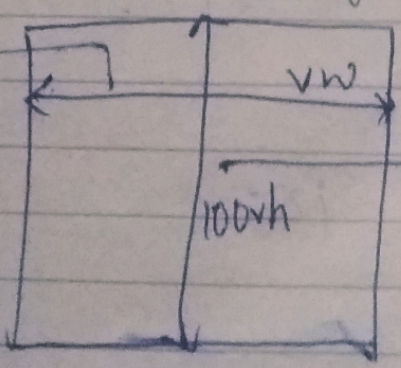
(4) Percentage → In order to make % to px ↓

100% = 80px;

200% = 160px; and so on can be calculated.

(5) vh → Complete height of the viewport

view-port height - Complete width of the browser.



→ Complete height of the browser.

↳ view-port width

(6) vw → Complete width of the view-port

{* We use vh and vw in terms of margins, width & height. }

* Transitions → To move from one state to another, or how we can make these moves smooth or effects can be added to that particular change.

Shorthand: transition.
transition - property
transition - duration
transition - timing-function
transition - delay

Syntax →

transition: width 1s 2s ;

(property) (duration) (delay)

* In transition - timing function we have →

(1) Linear → Works normally acc. to duration.

(2) Ease → In beginning starts fast and then becomes slow - starts

(3) ease-in → ~~Ends~~ slow and ends up comparitively fast

(4) ease-out → starts normally and ends up slow.

(5) ease-in-out → In this case we have slow start as well as slow end.

Syntax →

{ transition: width 2s ease; }

- So, now in-order to control the speed of transitions on your own we can use →

→ Cubic-bezier() → It has 4 properties.

Cubic-bezier(P_0, P_1, P_2, P_3):

So, all these 4 values come together and decide the transition from start to end.

So, P_0 and P_2 are the x-coordinates whereas, P_1 and P_3 are the y-coordinates.

So, we can say that →

P_0 and P_1 are the x and y coordinates for start: effect
whereas, P_2 and P_3 are for the end effect.

Generally the values in there goes from "0-1"

* In order to know about the values and how exactly the effect, we can visit → ("cubic-bezier.com")

* We can also put "all" in transition →

Ex → transition: all 2s;

but in here it will be applied to all the properties present inside that class.

* Transformations → for positioning of elements in terms of scale or rotation, or get a 2D or 3D effect out of it.

→ Shorthand: transform
→ scale(x,y), scaleX(), scaleY() → increases width & height of element
→ skew, skewX(), skewY() →
→ translateX(), translate(x,y) translateY() → It changes the position.
→ rotate
→ translate3D(x,y,z), translateZ
→ scale3D(x,y,z), scaleZ(z)

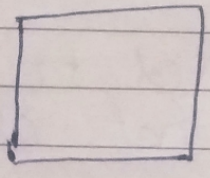
We can use → 'deg, turn'

Syntax →

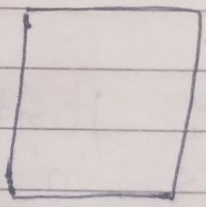
```
transform: rotate(30deg);
           rotate(0.5 turn);
           skew(30deg);
```

* Rotation always happen in clockwise direction.

o In translate →



Original Position



after translatex it shifts from it's position towards right horizontally.

When we do,

① transform: translate X (45px);

② translate [transform Y] makes it shift vertically or we can do downwards.

③ translate (45px, 45px) → makes it shift towards right as well as towards bottom.

{Similarly, we can have (-ve) values applied..}

* Animation →

Shorthand: animation.

{ animation: name (mandatory) duration timing-function delay
iteration-count direction fill-mode; }

Ex → animation-name: heading;

@keyframes heading {
100% {transform: scale(2.5, 2.5)}
}

{Key word for animation..}