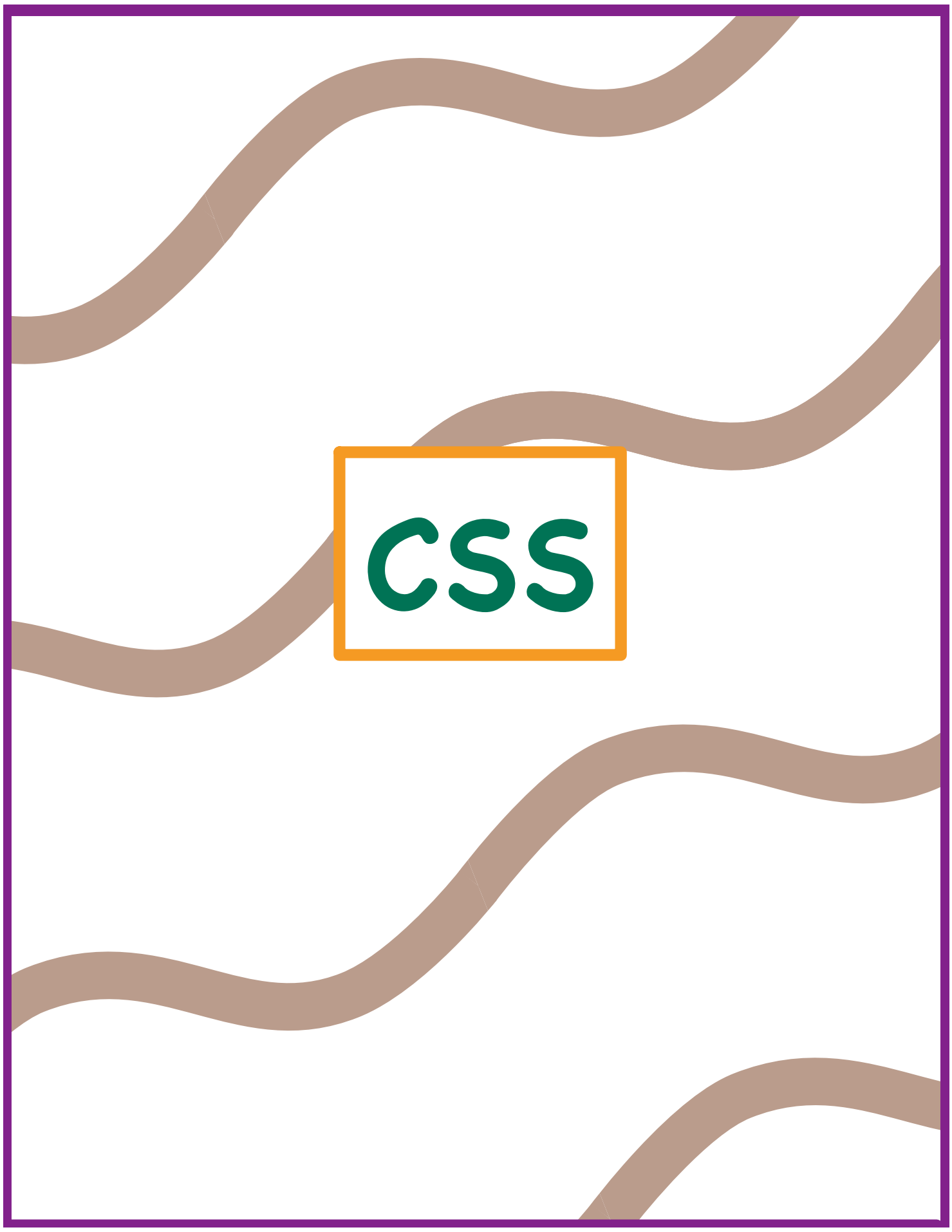

CSS

★ CSS NOTES ★

HTML is just a skeletal layout of a website. We need CSS to design a website, add styles to it and make it look beautiful.

- **What is CSS?**

CSS stands for cascading style sheets.

- **What is DOM?**

DOM stands for Document Object Model. When a page is loaded, the browser creates a DOM of the page which is constructed as a tree of objects.

⇒ HTML "ID" AND "CLASS" ATTRIBUTES:

- When an HTML element is given an ID, it serves as a unique identifier for that element.
- On the other hand, when an HTML element is given a class, it now belongs to that class. More than one element can belong to a single class but every element must have a unique ID (if assigned).

```
<div id = "first" class = "c1 c2 c3">
...
</div>
```

↗ Unique ID

↳ Multiple classes followed by spaces.

- Three ways to add CSS to HTML:

① <style> tag → Adding <style>...</style> to HTML.

② Inline CSS → Adding css using style attribute.

③ External CSS → Adding a stylesheet (*.css) to HTML using <link> tag.

CSS SELECTORS

A CSS selector is used to select HTML elements for styling.

```
body {          → selector
    color : red;          → Declaration (property : value)
    background : pink;    → Declaration
    }
```

- Element Selector:

It is used to select an element based off the tagname.

Eg.)
```
h2 {
    color: blue;
    }
```

- ID Selector:

It is used to select an element with a given id.

Eg.)
```
#first {        → # is used to target by id.
    color: white;
    }
```

■ Class Selector :

It is used to select an element with a given class.

Eg.)
```
.classname {
    background : red ;
}
```
→ (.) is used for class

Note; An inline style will override external and internal styles.

⇒ GROUPING SELECTORS ;

① * → Used as a universal selector to select all the elements.

```
* {
    margin : 0 ;
    padding : 0 ;
}
```

② element, element → Used to group two or more selectors.

```
h1, h2, h3, div {
    color : blue ;
}
```
↓ h1, h2, h3 and div will be blue now.

Space

③ element1 element2 → Selects all elements 2 which are inside element 1.

④ element1 > element2 → Selects all elements 2 whose parent is element 1.

⑤ element1 + element 2 → Select any element 2 that is exactly after element 1.

COLORS AND BACKGROUNDS

- The color Property;

The CSS color property can be used to set the text color inside an element.

```
p {
  color: red ;
}
```
→ Text color changes to red.

Types:

① RGB:

Specify color using Red, green and blue values.

Eg.) rgb(200, 98, 70)

② HEX Code: Specify color using HEX code.

Eg.) #ff7180

③ HSL: Hue, saturation and lightness values.

Eg.) hsl(8, 90%, 63%)

Note; There are also rgba and hsla.

a → alpha, it means opaqueness or transparency.

- The background-color property;

It specifies the background color of a container.

```
.brown {
    background-color: brown;
}
```

- The background-image property;

Used to set an image as the background.

```
body {
    background-image: url("hey.jpg");
}
```

⇒ The image is by default repeated in x and y directions.

- background-repeat property:

① repeat-x → Repeat in horizontal direction
② repeat-y → Repeat in vertical direction
③ no-repeat → Image doesn't repeat.

- The background-size property;

① Cover: fits and no empty space remains.

② Contain: fits and image is fully visible.

③ auto: Display in original size.

④ {{width}} : sets width and height is set automatically.

⑤ {{width}} {{height}} : set width and height.

- The background-position property;

Sets the starting position of a background image.

```
div1 {
    background-position : left top;
    }
```

- The background-attachment property;

Defines a ~~static~~ scrollable / non-scrollable character of a background image.

```
div2 {
    background-attachment : fixed;
    }
```

* The background short hand :

A single property to set multiple background properties.

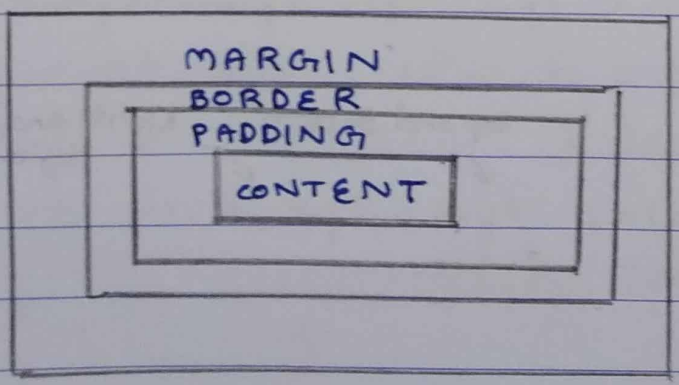
```
div 3 {                    ↗ color      ↗ image
background : red url("image.png")
        no-repeat fixed right top ;
                    ↓ repeat
}
```

⇒ One of the properties can be missing given that the rest of them are in order.

Note; Check the MDN Reference for more properties.

CSS Box Model

The CSS box model represents all HTML elements in the form of boxes.



- Setting Width and Height;

We can set the width and height in CSS as follows;

```
#box {
    height: 70px;
    width: 70px;
}
```

NOTE;

TOTAL HEIGHT = height + Top/Bottom padding + Top/Bottom border + Top/Bottom margin

- Setting margin and padding;

We can set the margin and padding as follows;

⇒
```
.box {
    margin: 3px;
    padding: 4px;
}
```

⇒ (Top, Right, Bottom, left)
```
.boxMain {
    margin: 7px 0px 2px 11px;
}
```

⇒ (top and Bottom; Right and left.)
```
.boxLast {
    margin: 7px 3px;
}
```

→ We can also set individual margins;

```
margin-top : 70px;
margin-bottom : 3px;
margin-left : 8px ;
margin-right : 9px ;
```

Note; All of the above is applicable for padding too.

- Setting Borders;

We can set the border as follows;

```
.box {
  border-width : 2px;
  border-style : solid;
  border-color : red;
}
```

You can also use;

```
border: 2px solid red;
```

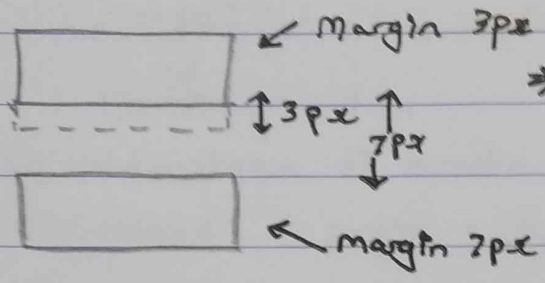
- Border Radius ;

We can set border radius to create rounded borders.

```
.div2 {
    border-radius : 7px;
}
```

- **Margin Collapse:**

When two margins from different elements overlap, the equivalent margin is the greater of the two. This is called margin collapse.



⇒ margin between them is collapsed to the bigger margin.

- **Box Sizing:**

Determines what out of pudding and border is included in elements, width and height.

Content BOX	Border Box
Include only content in width/height	the content width and height includes content + padding + border

```
.div1 {
    box-sizing : border-box;
}
```

FONTS AND DISPLAY

- **The display property:**

The css display property is used to determine whether an element is treated as a block/inline element and the layout used for its children. → flexbox/grid/etc.

→ display: inline

Takes only the space required by the element. No linebreaks before and after. setting width/height/margin/padding not allowed.

→ display: block

Takes full space available in width and leaves a newline before and after the element.

→ display: inline-block

Similar to inline but setting height, width, margin and padding is allowed. Elements can sit next to each other.

→ display: none vs visibility: hidden

With display: none, the element is removed from the document flow. Its space is not blocked.
With visibility: hidden, the element is hidden but its space is reserved.

→ text-align property:
Used to set the horizontal alignment of a text.

```
.div1 {
    text-align : center;
}
```

→ text-decoration property:
Used to decorate the text.
Can be overline, line-through, underline, none

→ text-transform property:
Used to specify uppercase and lowercase letters in a text.

```
p .uppercase {
    text-transform : uppercase;
}
```

- Line-height property:

Used to specify the space between lines.

```
.small {
    line-height: 0.7;
}
```

⇒ FONT: Font plays a very important role in the look and feel of a website.

Font-family:

Font Family specifies the font of a text. Can hold multiple values as a "fallback" system.

```
p {
    font-family: monospace;
}
```

↳ Always do this to ensure the correct font of your choice is rendered.

→ Web Safe Fonts:

These fonts are universally installed across browsers.

→ How to add google fonts:

In order to use custom google fonts, go to google fonts then select a style and finally paste it to the "style.css" of your page.

→ other font properties:

Some of the other font properties are listed below:

font-size → sets the size of the font.
font-style → sets the font style.
font-variant → sets whether text is displayed in small-caps.
font-weight → sets the weight of the font.

→ Generic Families:

Broad class of similar fonts eg. serif, sans-serif, just like when we say fruit, we can buy any fruits.

font-family → Specific
Generic-family → Generic

SIZE, POSITION AND LISTS

There are more units for describing size other than 'px'. There are rem, em, vw, vh, percentage etc.

Q) Whats wrong with pixels?

A) Pixels (Px) are relative to the viewing device. For a device with size 1920×1080, 1px is 1 unit out of 1820/1920.

→ Relative lengths;

These units are relative to other length properties.

1. em: Unit relative to the parent's font size.
2. rem: Unit relative to the root font size. (Html Tag)
3. vw: Unit relative to 1% viewport width.
4. vh: unit relative to 1% viewport height.
5. %: unit relative to the parent element.

→ min/max - height/width property;

CSS has a min-height, max-height, min-width and max-width. If the content is smaller than the minimum height, minimum height will be applied.

⇒ The position property;

Used to manipulate the location of an element.

① Static; The default position. top/bottom/left/right/ z-index has no effect.

② Relative; The top/bottom/left/right/z-index will now work. Otherwise, the element is in the flow of document like static.

③ Absolute; The element is removed from the flow and is relatively positioned to its first non-static ancestor. top/bottom etc. works.

④ Fixed: Just like absolute except the element is positioned relative to the browser window.

⑤ Sticky: The element is positioned based on user's scroll position.

→ list-style property:

The list style property is a shorthand for type, position and image.

```
ul {
    list-style : square inside url('');
}
```

list-style-position

list-style-type

list-style-image

→ z-index property;

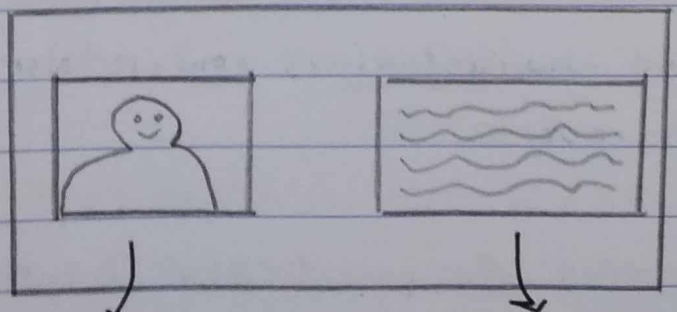
The z-index property specifies the stack order of an element.

It defines which layer which is the case with overlapping elements.

CSS FLEXBOX

- The float property:

It is used to float an element towards the right or left.



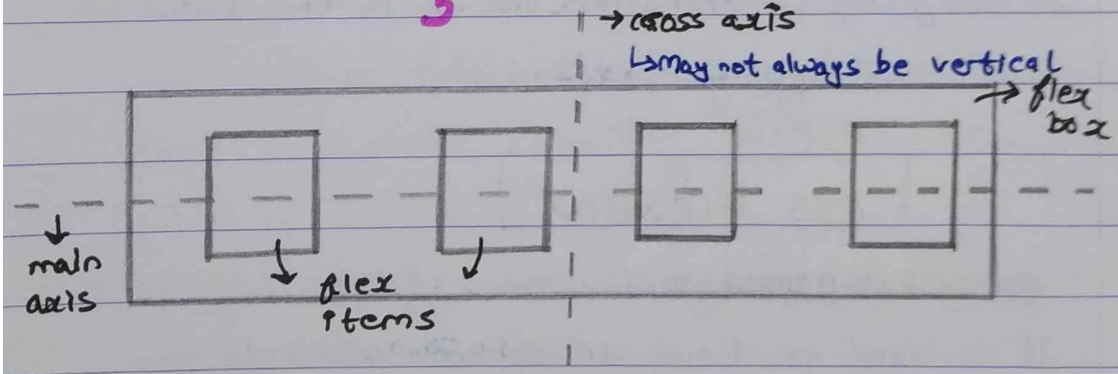
- the clear property:

Used to clear a float. It specifies what elements can float beside a given element.

⇒ THE CSS FLEXBOX :

Aims at providing a better way to layout, align and distribute space among items in a container.

```
container {
    display: flex; ⇒ Initializing a flexbox.
}
```



- flex direction property:

Defines the direction towards which items are laid. Can be row (default), row-reverse, column and column-reverse.

- flex properties for parent (flex container):

① Flex-wrap : Can be wrap, no-wrap, wrap-reverse. Wrap items as needed with this prop.

② Justify-content : Defines alignment along main axis.

③ align-items : Defines alignment across cross axis.

④ align-content : Aligns a flex container's lines when there is extra space in the cross-axis.

- Flex properties for children (Flex items) :

① order : Controls the order in which the items appear in the flex container.

② Align-self : Allows default alignment to be overridden for the individual flex items.

③ flex-grow : Defines the ability of a flex item to grow.

④ flex-shrink : Specifies how much a flex item will shrink relative to the rest of them.

CSS GRID AND MEDIA QUERIES

A CSS grid can be initialized using;

```
container {
    display: grid;
}
```

All direct children automatically become grid items.

- The grid-column-gap property;

Used to adjust the space between the columns of a CSS grid.

- The grid-row-gap property;

Used to adjust the space between the rows of a CSS grid.

- The grid-gap property;

Shorthand property for grid-row-gap and grid-column-gap.

```
container {
    display: grid;
    grid-gap: 40px 100px;
}
```

40px → Row
100px → Column

NOTE: For a single value of grid-gap, both row and column gaps can be set in one value.

⇒ Following are the properties for grid-container:

① The grid-template-columns property can be used to specify the width of columns.

```
.container {
    display: grid;
    grid-template-columns: 80px 120px auto;
}
```

② The grid-template-rows property can be used to specify the height of each row.

```
.container {
    display: grid;
    grid-template-columns: 80 px 120px auto;
}
```

→ default is stretch

③ The justify-content property is used to align the whole grid inside the container.

④ The align-content property is used to vertically align the whole grid inside the container.

⇒ Following are the properties for grid items:

① The grid-column property defines how many columns will an item span.

putting -1 here means to keep going till the end

```
.grid-item {
    grid-column: 1/3;
}
```

② The grid-raw property defines how many rows will an item span.

③ We can make an item to start on column 1 and span 3 columns like this:

```
.item {
    grid-column: 1/span 3;
}
```

★ CSS media queries:

Used to apply css only when a certain condition is true. Syntax:

```
@media only screen and (max-width: 800px) {
    body {
        background: red;
    }
}
```

- **Responsiveness in grids:**

We can use the "fr" size unit instead of any other. fr stands for fraction.

```
.container {
    grid-template-columns: 1fr 1fr 2fr;
}
```

- **The repeat() function:**

Using repeat (3, 1fr) is the same as 1fr 1fr 1fr, which makes the code a little smaller.

Note: You can use the auto-fill property in repeat() to make it more responsive and min max() to make it further res.

```
.container {
    grid-template-columns:
        repeat(auto-fill, minmax(100px,
                                 1fr));
}
```

TRANSFORMS, TRANSITIONS AND ANIMATIONS

Transforms are used to rotate, move, skew or scale elements. They are used to create a 3D effect.

- The transform property;

Used to apply a 2D or 3D transformation to an element.

- The transform-origin property:

Allows to change the position of transformed elements.

2D Transforms → Can change the x and y axis.

3D Transforms → Can change the z axis as well.

⇒ CSS 2D TRANSFORM METHODS:

1. Translate ()
2. rotate ()
3. scale X ()
4. scale Y ()
5. skew ()
6. matrix ()
7. scale ()

⇒ CSS 3D TRASFORM METHODS:

1. rotate X ()
2. rotate Y ()
3. rotate Z ()

⇒ CSS TRANSITIONS:

Used to change property values smoothly, over a given duration.

- **The transition property:**

The transition property is used to add transition in CSS.

Following are the properties;

① transition-property → the property you want to transition
② transition-duration → Time for which trans. will apply.
③ transition-timing-function → How do you want the property to transition.
④ transition-delay → Specifies the delay for the transition.

* All these properties can be set using a single shorthand property.

```
                                    ↗ ③ timing-function
transition : width 3s case-in 2s;
             ↓      ↓          ↓
          ① property  ② duration  ④ delay
```

→ Transitioning multiple properties:

We can transition multiple properties as follows;

transition: opacity 1s ease-out 1s, transform 2s ease-in;

⇒ CSS Animations :

Used to animate CSS properties with more control.

We can use @keyframes rule to change the animation from a given style to a new style.

```
@keyframes hello {          → can change
    from { width: 20px; }     multiple
    to { width: 31px; }       properties
}
```

⇒ PROPERTIES TO ADD ANIMATIONS :

① animation-name → name of the animation

② animation-duration → How long does the animation run?

③ animation-timing-function → Determines speed curve of the animation.

④ animation-delay → Delay for the start of an animation.

⑤ animation-iteration ~~count~~ cant → Number of times an animation should run.

⑥ animation-direction → Specifies the direction of the animation.

→ The animation shorthand ;

animation: hello 6s linear 1s infinite reverse;

① ② ③ ④ ⑤ ⑥

→ Using % value states with animation:

We can use % values to indicate what should happen when a certain percent of animation is completed.

```
@keyframes harry {
  0% {
      width: 20px;
  }
  50% {
      width: 80 px;
  }
  100% {
      width: 200px;
  }
}
```

→ Can add as many intermediate properties as you need.