Introduction to CSS:

Cascading Style Sheets styles HTML elements to make web pages look good. CSS adds design while HTML provides structures.

- How CSS works:

It uses selectors to target HTML elements.
A rule has a selector + declaration (property : value).

Eg: p { color: red; }

- 3 ways to attach CSS to HTML.

1. Inline CSS.
Written inside an HTML element using the style attribute.
Eg: <h1 style = " color: yellow; ">
Hello </h1>

Internal CSS:
2. Styles inside a <style> tag in the head of the HTML document.
Good for small projects with limited styling.

3. External CSS:
A separate .css file linked with
<link rel = "stylesheet" href = "style.css">
Best for scalable projects, content and presentation are separated.

* CSS selectors: It tells the browser which elements you want to style.

- common selector types:

Element selector: targets by tag name.
Eg: div { ... }

Class selector: Starts with . to style groups of elements.
Eg: .btn { ... }

ID selector: starts with # for a unique element:
Eg: #header { ... }

Child selector: selects direct child elements.
Eg: ul > li { ... }

Descendant selector: selects nested elements at any level.
Eg: div p { ... }

Universal Selector: Selects all elements on the page.

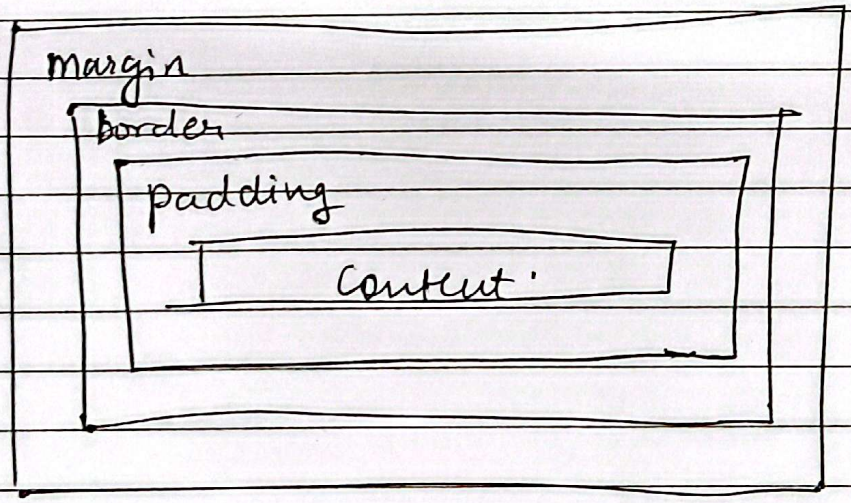
Pseudo selectors:
Eg: :hover, :first-child

* CSS Box model:

Every HTML element is a box:
Content + padding + border + margin.

Parts of the Box:

Content: actual text or image.
Padding: Space between content and border
Border: A line around the padding / content.
Margin: Space outside the border that separates elements from others.



* CSS fonts

Font -> the visual style of text, different fonts make the same text look different.

font-family controls which font is used for an element.

Eg: p {
font-family: "Segoe UI", Tahoma, Sans-serif;
}

Always add fallback fonts
End with generic family: serif, sans-serif, monospace.

- Google fonts

Add via <link> or @import.
Then use it in font-family;

- Font properties:

font-size - Size of text
font-weight - 100-900 (boldness)
font-style - normal/italic
line-height - Space between lines.
letter-spacing - Space between letters.

- text properties

text - align → left / right / center / justify
text - transform → uppercase / lowercase / capitalize.
text - decoration → underline / line-through
text - indent → indent first line.
text - overflow → ellipsis (with overflow hidden).

★ Colors in CSS

- ways to define color.

1. Named (red)
2. Hex (#ff0000)
3. RGB (rgb(255, 0, 0))
4. RGBA (rgba(255, 0, 0, 0.5))
5. HSL (hsl(0, 100%, 50%)).

★★ CSS specificity and cascade.

cascade order (high → low):

	Specificity points
↓ important	
inline	1000
ID	100
class / pseudo-class	10
↓ element	1
universal (*)	

- Best practices:

Avoid ! important
Avoid inline styles
use class selectors
keep specificity low.

* CSS units:

- Absolute
  px

- Relative
  % (relative to parent)
  em (parent font size)
  rem (root font size)
  vw (viewport width)
  vh (viewport height)
  vmin / vmax

- Min & max
  min-height
  max-width
  (used in responsive design)

* Display property:

-> Types:

block - full width
inline - content width
inline-block - inline + width/height
none - removed completely
visibility : hidden - space remains
flex :- flexible layout
grid - grid layout

- Flexbox (basic centering).

```
display: flex;
justify-content: center; (horizontal)
align-items: center; (vertical)
```


* Shadows

- Box shadow (used for cards & UI design).

```
box-shadow: x y blur spread color;
```

- Text shadow

```
text-shadow: x y blur color;
```

* Border v/s outline.

Border	Outline
Affects size	Doesn't affect size
Inside box model	Outside border.

* Styling lists.

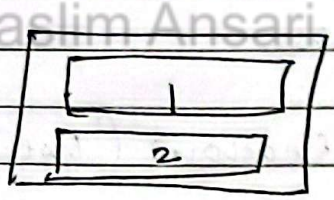
list-style-type → square/ circle/ none.
list-style-position → inside/ outside.

list-style: none; (remove bullets).

* overflow

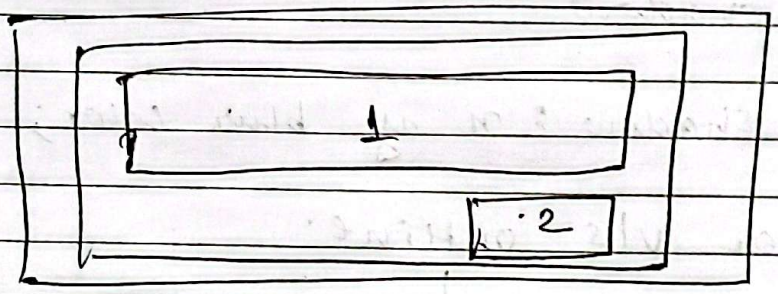
visible → default
hidden → cut content
scroll → always scroll
auto → scroll if needed.

* Position property

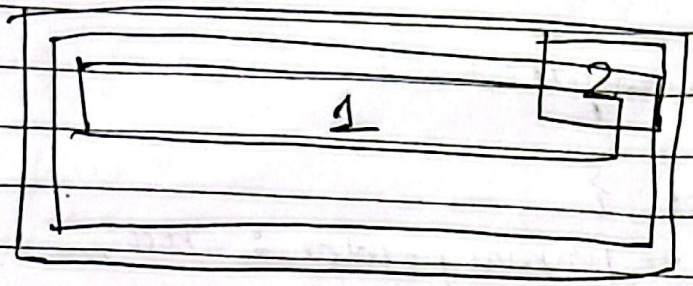


static → default.

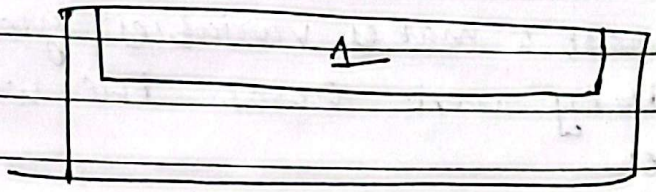
relative → moves visually
Space preserved



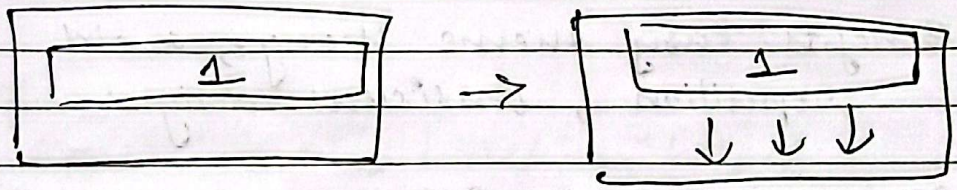
absolute → removed from flow
relative to nearest positioned element.



fixed :→ fixed on screen



Sticky → Scrolls then sticks



& CSS variables.

~~write textdef~~ : CSS variables (custom properties).

Definition : variables to store reusable values, declared inside a selector using --name: value;

Syntax example:

```
:root {
    --primary-color : red;
}
```

use → color : var(--primary-color);

Why? :root : makes variables global so they work across the whole page.

Fallback value: var(--primary-color, blue)
[uses blue if variable not defined]

Benefits: Easy theme change, avoid repetition, consistent design.

* CSS media queries. (Responsive design).

Purpose: Change CSS based on device features (width, height, orientation).

Basic syntax: @media only (max-width: 600px) {

}

use case:
Change layout from multi column to single column.

Resize fonts, hide/show elements.

* CSS float and clear.

float: moves element left/right, text wraps around it.
Common use: image with text, older column layout.
Problems: parent's height collapsing, elements overlapping.

clear: stops wrapping (clear: both;, left, right).

* more on CSS selectors.

Attribute selector:
[type="text"], [href^="https"],
[href$=".pdf"].

Pseudo classes: :nth-child(), :first-child, :last-child, :not().

Pseudo elements: ::before, ::after with content.

Combinators: Descendant: .parent .child.
child: .parent > .child.
Adjacent Sibling: h1 + p.

* Flexbox - Main concepts.

1. Container properties.

display: flex;
flex-direction: row, row-reverse, column, column-reverse;
justify-content: (horizontally) flex-start, center, space-between, space-around, space-evenly;
align-items: (vertically) flex-start, center, flex-end, stretch, baseline;
flex-wrap: nowrap, wrap, wrap-reverse;
align-content: controls space between wrapped lines;

2. Item properties:

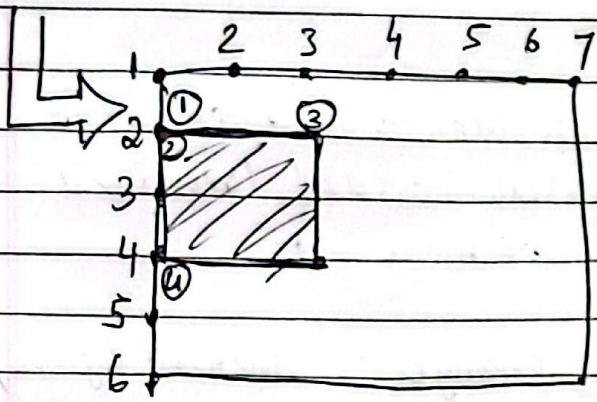
order: change visual order.
flex-grow, flex-shrink, flex-basis.
Shorthand: flex: grow shrink basis.
align-self: override 'align-items' for a single item.

* Css guid masterclass.

Enable guid: display: guid;
Define tracks:
grid-template-columns: repeat(3, 1fr);
grid-template-rows: auto, etc ...

Gaps:
row-gap, column-gap, gap.

Position items: guid-column: 1/3;
guid row: 2/4



Useful functions:
fr unit, minmax(),
auto-fit, auto-fill.

* CSS transforms:

transform functions:
translate(x, y)
scale(x, y)
rotate(deg):
skew(x, y):

Transform origin: transform-origin: center; etc.
2D vs 3D (rotateX, rotateY):
Combine transforms: transform: translateX(10px)
scale(1.2);

CSS transition property.

Concept: Transition = smooth change of CSS property values from one state to another (hover / class toggle).

Basic usage:
HTML: a box inside a container; a class like .translate added/removed (often via JS) to move it.

without transition: change is instant (jumps).
with transition: smooth movement.

- Core transition properties:

transition-property: which property to animate (eg: transform, background-color, or all).

transition-duration: how long change takes. (3s)

transition-timing function: speed curve of the change (linear, ease-in, ease-out, custom → cubic-bezier (---)).

transition-delay: wait time before transition starts (1s).

Eg: transition-property: transform;
transition-duration: 3s;
transition-timing function: ease-in-out;
transition-delay: 1s;

- Shorthand syntax:

transition: property duration timing-function delay;

Eg: transition: all 3s ease-in-out 1s;

- Multiple properties:

Transition all: transition-property: all;
(affects every property that changes).

Specific multiple: transition-property:
background-color, transform;

- Timing functions and dev tools.

Dev tools lets you visually edit
cubic-bezier (---) to customize
easing curves; you copy from
the generated function to the CSS.

CSS animations:

difference vls transition:

transition: from old value to new, triggered by state change.

Animation: defined sequence of keyframes; can loop, reverse, pause, etc and change many properties over time.

- Defining an animation:

1. write @keyframes with a name:
   using from/to or 0% to 100%.

2. Apply it to an element with animation-* properties.

Eg:
```
@keyframes harryAnimation {
    from {
        transform: translateX(0);
    }
    to {
        transform: translateX(100px);
    }
}
```

- Important animation properties:

animation-name: which keyframes to use.
animation-duration: how long one cycle takes (3s).
animation-iteration-count: how many times to repeat (1, 3, infinite).
animation-timing-function: like transitions (ease, linear, etc).
animation-delay: wait before first cycle starts.
animation-direction:
  normal 0% → 100%
  reverse 100% → 0%
  alternate forward then backward.
  alternate-reverse backward then ~~reverse~~ forward.
animation-play-state: running or paused (often toggled via JS).
animation-fill-mode:
  none: element returns to original style.
  forwards: keeps final keyframe styles after animation.
  backwards: apply starting keyframes style during delay.
  both: combination of forwards + backwards.

- Shorthand Syntax:
Order: animation: name duration timing-function delay iteration-count direction fill-mode play-state;

Note ✱ : You can apply multiple animations by comma separating values in animation & and defining multiple @keyframes blocks.

✱ CSS object-fit and object-position

Problem: Images often get stretched / cropped inside fixed size containers.

— object-fit:
used on replaced elements (like <img>, <video>).

▲ Common values:

fill (default): image stretches to fill box (can distort).
cover: image covers entire box, maintains aspect ratio, may crop edges.
contain: entire image visible inside box, keeps ratio; may leave empty space (letterboxing).
none: image not resized; may overflow.
scale-down: smaller of none or contain.

Eg:
```
.img-cover {
    width: 300px;
```

height: 200px;
object-fit: cover;
}

- Object position:

Controls which part of the image stays visible when cropped (with cover/none).

Syntax: object-position: x y;
eg: center center,
top, bottom, left, right or
percentages like 50% 20%

* CSS filters:

Filters apply visual effects to images or elements.

- common filter functions:

blur(px): blur strength.
brightness(%): 100% = normal,
>100% = brighter,
<100% = darker.
contrast(%): adjust contrast.
grayscale(%): 0% color, 100% full B/W.
Sepia(%): brownish old photo effect.
saturate(%): color intensity.
invert(%): invert colors.

hue-rotate (deg) : rotate color-hue.

Eg:

filter: grayscale (100%) blur (2px);

You can also use backdrop-filter on backgrounds.

(blur behind a translucent panel).