**Cascading Style Sheets:**

It is used to style HTML elements by defining layout, colors, fonts, animations, and more.

**Common CSS Abbreviations:**

| **Abbreviation** | **Meaning** |
| --- | --- |
| **CSS** | **Cascading Style Sheets** |
| **HTML** | **HyperText Markup Language** |
| **ID** | **Identifier (e.g., #id)** |
| **RGB** | **Red, Green, Blue (color model)** |
| **HEX** | **Hexadecimal color code (e.g., #ff5733)** |
| **HSL** | **Hue, Saturation, Lightness (color model)** |
| **VW** | **Viewport Width** |
| **VH** | **Viewport Height** |
| **EM** | **Relative to parent element's font size** |
| **REM** | **Relative to root element's font size** |
| **PX** | **Pixels (absolute unit)** |
| **%** | **Percentage (relative unit)** |
| **BG** | **Background** |
| **FW** | **Font Weight** |
| **LH** | **Line Height** |
| **WS** | **White Space** |
| **TT** | **Text Transform (uppercase, lowercase)** |
| **D** | **Display (e.g., d: flex;)** |
| **P** | **Padding (e.g., p: 10px;)** |
| **M** | **Margin (e.g., m: 10px;)** |
| **W** | **Width (e.g., w: 100%;)** |
| **H** | **Height (e.g., h: 100vh;)** |
| **OP** | **Opacity** |
| **POS** | **Position (e.g., pos: relative;)** |
| **FZ** | **Font Size** |
| **TA** | **Text Align** |

**Example:**

**/\* CSS using shorthand \*/**

**.bg-red { background-color: red; }**

**.fw-bold { font-weight: bold; }**

**.d-flex { display: flex; justify-content: center; align-items: center; }**

**Understanding CSS Structure:**

**A CSS rule consists of the selector, declaration block, properties, and values.**

**1. Selector:**

The **selector** defines which HTML elements the styles apply to.

#### Example:

p {

color: red;

}

✅ **Selector**: p (applies styles to all <p> elements).

1. **Declaration Block:**

The **declaration block** is enclosed in {} and contains one or more declarations.

#### Example:

p {

color: red;

font-size: 16px;

}

✅ **Declaration Block**: Everything inside { }.

1. **Properties:**

A **property** defines **what** aspect of the element to style (e.g., color, font-size, margin).

#### Example:

p {

color: red;

font-size: 16px;

}

✅ **Properties**: color, font-size.

1. **Values:**

A **value** defines **how** the property should be applied. It comes after : and is followed by ;.

#### Example:

p {

color: red;

font-size: 16px;

}

✅ **Values**: red, 16px.

1. **Complete Breakdown :**

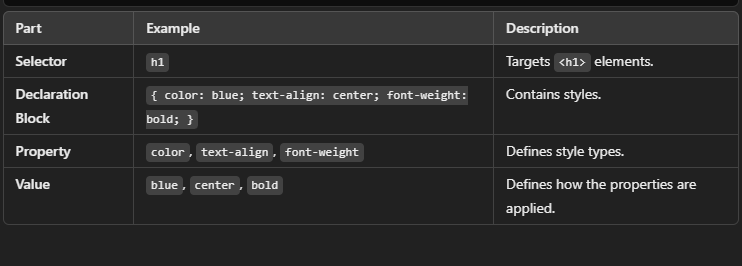
h1 {

color: blue;

text-align: center;

font-weight: bold;

}



**Another Example:**

button {

background-color: green;

border-radius: 5px;

padding: 10px 20px;

}

✅ **Selector**: button  
✅ **Properties**: background-color, border-radius, padding  
✅ **Values**: green, 5px, 10px 20px

**Types of CSS:**

CSS can be applied in three different ways:

**1. Inline CSS (Inside the HTML tag):**

✅ Styles applied **directly** to an element using the style attribute.  
✅ **Used for quick styling** but not recommended for large projects.

<p style="color: red; font-size: 18px;">This is inline CSS</p>

**Pros:**  
 Quick to apply.  
 Overrides other styles.

**Cons:**  
 ❌ Difficult to maintain.  
 ❌ Not reusable.

1. **Internal CSS (Inside <style> tag in the <head> section):**

✅ Styles are written within the HTML file inside a <style> tag.  
✅ **Used for small projects or single-page designs**.

### ****Example****:

<!DOCTYPE html>

<html lang="en">

<head>

<style>

p {

color: blue;

font-size: 20px;

}

</style>

</head>

<body>

<p>This is internal CSS</p>

</body>

</html>

**Pros**:  
 Keeps styles separate from elements.  
 Easier to maintain than inline CSS.

**Cons**:  
 ❌ Not reusable across multiple pages.

1. **External CSS (Linked CSS file):**

✅ Styles are written in a separate .css file and linked using the <link> tag.  
✅ **Best for large projects** as it allows reusability across multiple pages.

### ****Example:****

**HTML File (**index.html**)**

<!DOCTYPE html>

<html lang="en">

<head>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<p>This is external CSS</p>

</body>

</html>

**CSS File (** styles.css )

p {

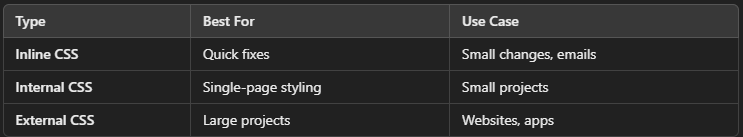
color: green;

font-size: 22px;

}

**Pros**:  
 Best for scalability and reusability.  
 Keeps HTML clean and organized.

**Cons**:  
 ❌ Requires an additional HTTP request to load the file.



**Id,Class And Univesal In Css:**

In CSS, id, class, and universal selectors are used to target HTML elements to apply styles. Here's a breakdown of each:

**1. ID Selector (#):**

* An id selector targets a specific HTML element with a unique id attribute.
* The id should be unique within a page, meaning only one element should have a specific id.
* In CSS, it is prefixed with a # symbol.

### ****Example:****

### ****Html:****

**<div id="header">This is the header</div>**

**Css:**

#header {

color: blue;

}

This CSS rule will apply the style to the element with the id="header".

**2. Class Selector (.):**

* A class selector targets elements that have a specific class attribute.
* Multiple elements can share the same class, so this is often used for styling groups of elements.
* In CSS, it is prefixed with a . symbol.

### ****Example:****

### ****Html:****

<p class="intro">This is an introduction.</p>

<p class="intro">This is another introduction.</p>

### ****Css:****

.intro {

font-size: 16px;

}

This CSS rule will apply the style to all elements with the class intro.

**3. Universal Selector (\*):**

* The universal selector targets all elements on the page.
* It's often used for general styling, like setting a global font or resetting margins/paddings.

### ****Example:****

\* {

margin: 0;

padding: 0;

}

This CSS rule will remove all default margins and paddings from all elements on the page.

### ****Summary:****

* #id: Targets a unique element by its id.
* .class: Targets multiple elements by their class name.
* \*: Targets every element on the page.

**Color In Css:**

In CSS, you can define colors using different methods. Here’s a breakdown:

### ****1. Named Colors:****

CSS provides predefined color names.

color: red;

background-color: blue;

### ****2. Hex Codes:****

A six-digit hexadecimal code (#RRGGBB or shorthand #RGB).

color: #ff5733; /\* Reddish-orange \*/

background-color: #000; /\* Black \*/

### ****3. RGB (Red, Green, Blue):****

Values range from 0-255 per channel.

color: rgb(255, 87, 51);

background-color: rgb(0, 0, 0);

### ****4. RGBA (RGB + Alpha Transparency):****

The a (alpha) value ranges from 0 (fully transparent) to 1 (fully opaque).

color: rgba(255, 87, 51, 0.5);

### ****5. HSL (Hue, Saturation, Lightness):****

* Hue (0-360 degrees on the color wheel)
* Saturation (0% gray, 100% full color)
* Lightness (0% black, 100% white)

color: hsl(11, 100%, 50%); /\* Reddish-orange \*/

### ****HSLA (HSL + Alpha Transparency):****

color: hsla(11, 100%, 50%, 0.5);

### ****CurrentColor:****

Uses the current text color.

border: 2px solid currentColor;

### ****Transparent:****

Makes an element fully transparent.

background-color: transparent;

**Background in CSS:**

The background property in CSS allows you to style the background of elements in various ways. Here are the key aspects:

### ****Background Color:****

Sets a solid color as the background.

background-color: lightblue;

## ****2. Background Image:****

Sets an image as the background.

background-image: url('image.jpg');

### ****Background Image Properties:****

* **Size** (background-size)

background-size: cover; /\* Covers the entire element \*/

background-size: contain; /\* Fits inside the element \*/

background-size: 100px 100px; /\* Custom size \*/

* **Position** (background-position):

background-position: center center;

background-position: top right;

background-position: 50% 50%;

* **Repeat** (background-repeat):

background-repeat: no-repeat;

background-repeat: repeat-x; /\* Repeats horizontally \*/

background-repeat: repeat-y; /\* Repeats vertically \*/

* **Attachment** (background-attachment):

background-attachment: fixed; /\* Stays in place while scrolling \*/

background-attachment: scroll; /\* Moves with content \*/

## ****3.Background Gradient:****

## ****Linear Gradient:****

background: linear-gradient(to right, red, blue);

## ****Radial Gradient:****

background: radial-gradient(circle, red, blue);

## ****Background Shorthand:****

Instead of setting each property separately, you can use shorthand:

background: url('image.jpg') no-repeat center center / cover;

**Order:** background-color → background-image → background-repeat → background-attachment → background-position → background-size

**Border In CSS:**

1. **Basic Border:**

border: 2px solid black;

**Syntax:**  
border: <width> <style> <color>;

* **Width:** px, em, rem (e.g., 2px, 0.5em)
* **Style:** solid, dashed, dotted, double, groove, ridge, inset, outset
* **Color:** Named colors, hex, RGB, etc.

1. **Individual Border Sides:**

You can set different styles for each side:

border-top: 3px solid red;

border-right: 2px dashed blue;

border-bottom: 4px double green;

border-left: 5px dotted orange;

1. **Rounded Borders (Border Radius):**

border-radius: 10px; /\* Rounded corners \*/

border-radius: 50%; /\* Perfect circle \*/

1. **Border with Shadows:**

box-shadow: 5px 5px 10px rgba(0, 0, 0, 0.3);

1. **Border Gradient (Advanced):**

CSS doesn’t support gradient borders directly, but you can use tricks:

border: 5px solid transparent;

border-image: linear-gradient(to right, red, blue) 1;

**Margin in CSS:**

The margin property in CSS is used to create space outside an element, separating it from other elements.

**1. Basic Margin:**

margin: 20px;

This applies a **20px** margin to all four sides of the element.

**2. Margin for Specific Sides:**

margin-top: 10px;

margin-right: 15px;

margin-bottom: 20px;

margin-left: 25px;

**3. Shorthand Margin:**

margin: 10px 20px 30px 40px; /\* top right bottom left \*/

margin: 10px 20px; /\* top-bottom left-right \*/

margin: 10px 20px 30px; /\* top left-right bottom \*/

**4. Auto Margin (Centering):**

To **horizontally center** a block element like a div:

margin: 0 auto;

· 0 (top & bottom)

· auto (left & right) = Centering effect

**5. Negative Margin:**

You can use negative values to **overlap** elements.

margin-left: -20px;

**6. Margin Collapse (Special Case):**

When vertical margins of two elements meet, the **larger margin wins** instead of adding both.

div {

margin-bottom: 20px;

}

p {

margin-top: 30px;

}

/\* The actual space between the two elements = 30px, not 50px \*/

**Padding in CSS:**

The padding property in CSS is used to create space inside an element, between the content and its border.

**1. Basic Padding:**

padding: 20px;

This applies 20px padding to all four sides of the element.

**2. Padding for Specific Sides:**

padding-top: 10px;

padding-right: 15px;

padding-bottom: 20px;

padding-left: 25px;

**3. Shorthand Padding:**

padding: 10px 20px 30px 40px; /\* top right bottom left \*/

padding: 10px 20px; /\* top-bottom left-right \*/

padding: 10px 20px 30px; /\* top left-right bottom \*/

**4.Padding with Percentage (%):**

padding: 10%;

* Percentage values are relative to the **element's width**, not height.

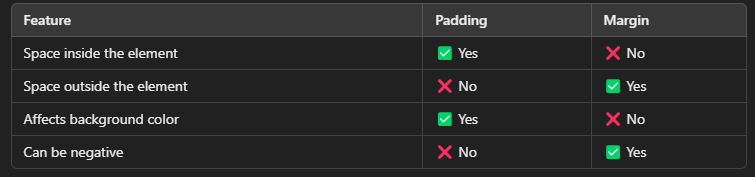
**5. Padding and Box Model:**

By default, padding increases the element’s total size.  
To prevent this, use:

box-sizing: border-box;

This makes sure padding **does not** increase the width/height.

1. **Padding vs Margin (Difference):**



**Height and Width in CSS:**

The height and width properties control the size of an element.

1. **Basic Width & Height:**

width: 200px;

height: 100px;

This sets the element to **200px wide** and **100px tall**.

1. **Percentage (%) Width & Height:**

width: 50%; /\* 50% of the parent element’s width \*/

height: 75%; /\* 75% of the parent element’s height \*/

1. **Auto Value:**

width: auto;

height: auto;

* auto makes the element expand based on content.

**4. Viewport Units (Responsive Design):**

width: 50vw; /\* 50% of viewport width \*/

height: 50vh; /\* 50% of viewport height \*/

* vw = Viewport width
* vh = Viewport height
* vmin = Smaller of vw or vh
* vmax = Larger of vw or vh

**5. Min & Max Width/Height:**

min-width: 150px; /\* Minimum width \*/

max-width: 500px; /\* Maximum width \*/

min-height: 100px; /\* Minimum height \*/

max-height: 300px; /\* Maximum height \*/

This prevents an element from becoming too **small** or **large.**

**6. Box Sizing (Avoid Overflow Issues):**

By default, width and height do not include padding and border.  
To fix this, use:

box-sizing:

border-box;

This makes sure padding and border **stay inside** the defined width/height.

**Text Formatting in CSS:**

CSS provides various properties to style and format text.

1. **Text Color:**

color: red; /\* Named color \*/

color: #ff5733; /\* Hex code \*/

color: rgb(255, 87, 51); /\* RGB value \*/

1. **Font Size:**

font-size: 16px; /\* Pixels \*/

font-size: 1.5em; /\* Relative to parent font-size \*/

font-size: 120%; /\* Percentage of default size \*/

1. **Font Family:**

font-family: Arial, sans-serif;

font-family: "Times New Roman", serif;

font-family: "Poppins", sans-serif;

* Use **multiple font choices** as fallbacks.

1. **Font Weight (Boldness):**

font-weight: normal; /\* Default \*/

font-weight: bold; /\* Bold \*/

font-weight: 300; /\* Lighter \*/

font-weight: 900; /\* Extra Bold \*/

**5. Font Style (Italic, Oblique):**

font-style: normal;

font-style: italic;

font-style: oblique;

**6.Text Alignment:**

text-align: left;

text-align: center;

text-align: right;

text-align: justify; /\* Spreads text evenly \*/

1. **Text Decoration (Underline, Line-through, Overline):**

text-decoration: underline;

text-decoration: line-through;

text-decoration: overline;

text-decoration: none; /\* Remove default underlines \*/

1. **Text Transformation (Uppercase, Lowercase, Capitalize):**

text-transform: uppercase;

text-transform: lowercase;

text-transform: capitalize; /\* First letter uppercase \*/

1. **Letter & Word Spacing:**

letter-spacing: 2px; /\* Space between letters \*/

word-spacing: 5px; /\* Space between words \*/

1. **Line Height (Spacing Between Lines):**

line-height: 1.5; /\* 1.5 times font size \*/

line-height: 20px; /\* Fixed spacing \*/

1. **Text Shadow:**

text-shadow: 2px 2px 5px rgba(0, 0, 0, 0.5);

* **Syntax:** text-shadow: x-offset y-offset blur-radius color;

1. **White Space Handling:**

white-space: nowrap; /\* Prevents text wrapping \*/

white-space: pre; /\* Preserves spaces & line breaks \*/

white-space: normal; /\* Default (wraps text) \*/

**Font Families in CSS:**

The font-family property in CSS defines the typeface (font) for text.

1. **Basic Syntax:**

font-family: "Arial", sans-serif;

* **First font**: Preferred font
* **Second font**: Fallback font
* **Generic family**: Used if no fonts are available

1. **Generic Font Families:**

There are five generic families:

* **serif** → Fonts with small strokes (e.g., Times New Roman)
* **sans-serif** → Clean fonts without strokes (e.g., Arial)
* **monospace** → Each letter has equal width (e.g., Courier New)
* **cursive** → Handwritten-style fonts (e.g., Brush Script)
* **fantasy** → Decorative fonts (e.g., Papyrus)

**Example:**

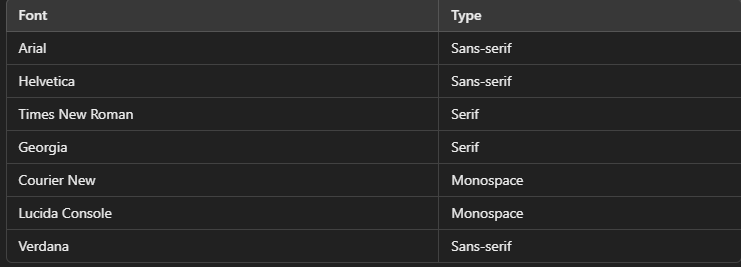
font-family: "Georgia", serif;

font-family: "Verdana", sans-serif;

font-family: "Courier New", monospace;

font-family: "Comic Sans MS", cursive;

1. **Web-Safe Fonts (Commonly Available):**



font-family: "Helvetica", "Arial", sans-serif;

## ****4. Custom Google Fonts:****

You can use **Google Fonts** for more variety.

**Step 1: Import the Font:**

Add this inside your <head>:

<link href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;700&display=swap" rel="stylesheet">

**Step 2: Apply in CSS:**

font-family: "Poppins", sans-serif;

## ****5. Using @font-face (Custom Fonts):****

If you have a custom font file, use:

@font-face {

font-family: "MyCustomFont";

src: url("myfont.woff2") format("woff2"),

url("myfont.woff") format("woff");

}

p {

font-family: "MyCustomFont", sans-serif;

}

## ****6.System Fonts (Faster Load):****

font-family: system-ui, sans-serif;

This uses the default OS font (e.g., **San Francisco** on Mac, **Segoe UI** on Windows).

**Styling Tables in CSS:**

CSS provides various properties to style tables, making them more visually appealing.

1. **Basic Table Styling:**

table {

width: 100%;

border-collapse: collapse; /\* Merges borders \*/

}

th, td {

border: 1px solid black;

padding: 10px;

text-align: left;

}

**Explanation:**

* width: 100% → Makes the table stretch to full width.
* border-collapse: collapse; → Merges adjacent borders.
* padding: 10px; → Adds spacing inside cells.
* text-align: left; → Aligns text to the left.

1. **Adding Table Borders:**

table, th, td {

border: 2px solid black;

}

* This applies a **black border** to the table, header (th), and cells (td).

1. **Alternating Row Colors (Zebra Stripes):**

tr:nth-child(even) {

background-color: #f2f2f2; /\* Light gray \*/

}

* This styles **even rows** differently for better readability.

1. **Hover Effect:**

tr:hover {

background-color: lightblue;

}

* Changes the row’s background when hovered over.

1. **Styling Table Headers:**

th {

background-color: #333;

color: white;

font-weight: bold;

}

* This adds a **dark background** and **white text** to headers.

## ****6. Responsive Table (Scrollable):****

For small screens, wrap the table in a **scrollable div**:

**CSS:**

.table-container {

overflow-x: auto;

}

table {

width: 100%;

min-width: 600px; /\* Prevents table from shrinking \*/

}

**Html:**

<div class="table-container">

<table>

<tr>

<th>Name</th>

<th>Age</th>

</tr>

<tr>

<td>Alice</td>

<td>25</td>

</tr>

</table>

</div>

## ****Priority of Styles in CSS (CSS Specificity & Cascade):****

When multiple CSS rules apply to an element, CSS determines which one takes priority based on **specificity, importance, and source order**.

## ****1. Order of Priority (Highest to Lowest):****

* **Inline Styles** (Highest priority)
* **ID Selectors** (#id)
* **Class, Attribute, and Pseudo-class Selectors** (.class, [attribute], :hover)
* **Element (Tag) Selectors** (div, p, h1)
* **Universal Selector (**\***) and Inherited Styles** (Lowest priority)

## ****2. Specificity Calculation:****

Each CSS rule gets a **specificity score**:

* **Inline styles** (style="color: red;") → 1000
* **ID selector** (#id) → 100
* **Class, Attribute, Pseudo-class** (.class, [type="text"], :hover) → 10
* **Element selector** (div, p, h1) → 1
* **Universal selector (**\***)** → 0

### Example of Specificity Scores:

#header { color: blue; } /\* 100 \*/

.navbar { color: red; } /\* 10 \*/

h1 { color: green; } /\* 1 \*/

If applied to <h1 id="header" class="navbar">Text</h1>, it will be **blue** (#header has the highest specificity).

**3.The !important Rule (Overrides Everything):**

p {

color: blue !important; /\* Forces this rule \*/

}

· !important overrides all other styles, even inline styles.

· However, avoid overusing it because it makes debugging harder.

## ****4. Source Order (Last Rule Wins):****

If two rules have the **same specificity**, the **last one in the CSS file** takes priority.

p {

color: red;

}

p {

color: blue;

}

/\* The text will be BLUE because the second rule comes last. \*/

## ****5. Inheritance in CSS:****

* Some properties inherit from their parent (e.g., color, font-family).
* Others do not inherit by default (e.g., margin, padding).

For non-inherited properties, you can **force inheritance**:

**CSS:**

button {

all: inherit;

}

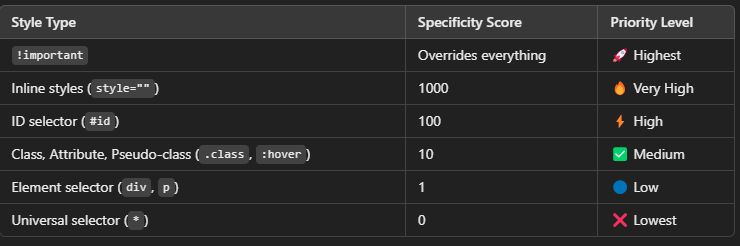
**Or reset styles:**

button {

all: unset;

}

**Priority Recap:**



## ****Rounded Corners in CSS:****

The border-radius property in CSS is used to create **rounded corners** for elements like divs, buttons, images, and cards.

## ****1. Basic Rounded Corners:****

.box {

border: 2px solid #4CAF50;

border-radius: 10px;

padding: 20px;

width: 200px;

height: 100px;

}

border-radius: 10px; → Adds **10px rounding** to **all corners**.

## ****Different Radii for Each Corner:****

.box {

border-radius: 10px 20px 30px 40px;

}

/\* Order: top-left | top-right | bottom-right | bottom-left \*/

Creates **asymmetric rounded corners** with different radii.

## ****Fully Rounded (Circle or Pill Shape):****

### ****Circle (for square elements):****

.circle {

width: 100px;

height: 100px;

background-color: #4CAF50;

border-radius: 50%;

}

border-radius: 50%; → Makes the element a perfect circle (if width = height).

**Pill-Shaped Button:**

.button {

padding: 10px 30px;

background-color: #4CAF50;

color: white;

border-radius: 50px;

border: none;

cursor: pointer;

}

Creates a pill-shaped button with rounded edges.

## ****4. Rounded Corners on Specific Sides:****

.box {

border-radius: 20px 0 0 20px;

}/\* Only top-left and bottom-left corners rounded \*/

Or use **specific properties**:

.box {

border-top-left-radius: 15px;

border-top-right-radius: 5px;

border-bottom-right-radius: 25px;

border-bottom-left-radius: 0;

}

## ****5. Elliptical Rounded Corners:****

.box {

border-radius: 50px 25px;

}/\* Horizontal radius | Vertical radius \*/

Creates elliptical corners by specifying two values.

## ****Rounded Corners with Images:****

img {

width: 300px;

height: auto;

border-radius: 15px;

}

Rounds the corners of images for a smoother look.

## ****Advanced Shape (Blob-Like Corners):****

.blob {

width: 250px;

height: 250px;

background-color: #ff6b81;

border-radius: 50% 40% 30% 60% / 60% 30% 40% 50%;

}

Generates a blob-like, organic shape using complex border-radius values.

## ****Button with Hover Rounded Animation****

.button {

padding: 15px 30px;

background-color: #6200ea;

color: white;

border: none;

border-radius: 5px;

transition: border-radius 0.3s ease;

}

.button:hover {

border-radius: 50px;

}

Smoothly animates the border-radius on hover.

1. **Rounded Card with Shadow:**

.card {

width: 300px;

padding: 20px;

border-radius: 20px;

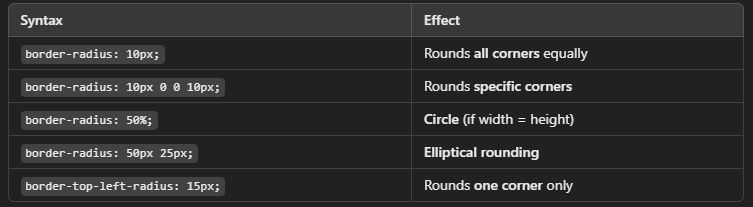
box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);

background-color: white;

}

Combines rounded corners with a soft shadow for a clean, modern UI.

**Summary of border-radius Usage:**



## ****Position In Css:****

The position property in CSS specifies how an element is **positioned** in the document. It defines the **stacking context** and how elements are placed relative to their container, viewport, or other elements.

## ****1. Static (Default):****

* This is the default position for all HTML elements.
* Elements follow the normal document flow with no special positioning.
* top, right, bottom, left properties do not apply here.

div {

position: static;

}

## ****2. Relative:****

* The element is positioned relative to its normal position.
* Use top, right, bottom, and left to offset it.
* Takes up space in the normal document flow.

.relative-box {

position: relative;

top: 20px;

left: 30px;

}

Moves the element 20px down and 30px right from its original spot.

## ****3. Absolute:****

* The element is positioned relative to the nearest positioned ancestor (relative, absolute, or fixed).
* If no such ancestor exists, it positions relative to the <html> (viewport).
* Removed from normal flow (other elements ignore its space).

.parent {

position: relative;

}

.child {

position: absolute;

top: 10px;

right: 15px;

}

The .child will be positioned inside .parent, 10px from the top and 15px from the right.

## ****4. Fixed:****

* The element is positioned relative to the viewport.
* It stays fixed in place when scrolling.
* Commonly used for sticky headers, sidebars, or floating buttons.

.fixed-header {

position: fixed;

top: 0;

left: 0;

width: 100%;

background-color: #333;

color: white;

padding: 20px;

}

The header remains at the top even when the page is scrolled.

## ****5. Sticky:****

* The element toggles between relative and fixed positioning depending on the scroll position.
* It sticks to a specified position when you scroll past it.

.sticky-nav {

position: sticky;

top: 0;

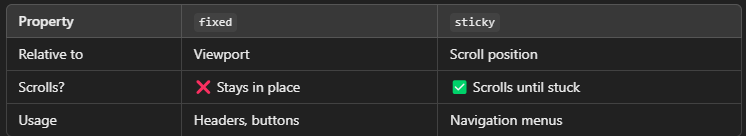
background-color: #ff6b81;

padding: 10px;

}

The navbar will "stick" to the top of the page when you scroll past it.

## ****6.Fixed vs. Sticky (Key Difference):****



## ****7. Z-Index (Layering with Position):****

* The z-index property controls the stacking order of overlapping elements.
* Higher values are placed in front of lower ones.

.box1 {

position: absolute;

z-index: 1;

}

.box2 {

position: absolute;

z-index: 2; /\* Will appear on top of box1 \*/

}

## ****8.Full-Screen Overlays with Absolute & Fixed:****

.overlay {

position: fixed;

top: 0;

left: 0;

width: 100vw;

height: 100vh;

background-color: rgba(0, 0, 0, 0.5);

}

A common pattern for **modals** and **full-screen overlays**.

**9.Combining Position Types for Advanced Layouts:**

.container {

position: relative;

}

.tooltip {

position: absolute;

top: 100%;

left: 0;

margin-top: 10px;

background-color: black;

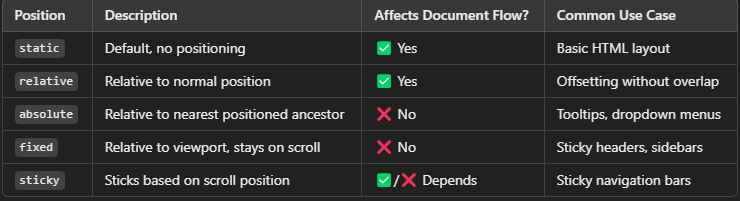
color: white;

padding: 5px;

}

A **tooltip** positioned relative to its container.

1. **CSS Position Summary:**



## ****Overflow In Css:****

The overflow property in CSS controls **what happens when content overflows** the boundaries of its container. It determines whether content should be clipped, visible, or scrollable.

**1. Values of overflow:**

### ✅ **visible (Default):**

* Content is not clipped and overflows outside the element’s box.

.box {

width: 200px;

height: 100px;

overflow: visible;

border: 2px solid #4CAF50;

}

Content spills out if it doesn't fit.

### **Hidden:**

* Hides extra content that overflows the container.
* No scrolling is provided to access the hidden content.

.box {

width: 200px;

height: 100px;

overflow: hidden;

border: 2px solid #4CAF50;

}

Useful for hiding overflow without scrollbars.

### **Scroll:**

* Adds scrollbars (horizontal/vertical) even if content fits.
* Ensures content is always scrollable.

.box {

width: 200px;

height: 100px;

overflow: scroll;

border: 2px solid #4CAF50;

}

Best for dynamic content areas where scrollbars should appear only if needed.

**2. Overflow on X and Y Axes:**

Control **horizontal (**overflow-x**)** and **vertical (**overflow-y**)** overflow separately.

.box {

width: 200px;

height: 100px;

overflow-x: scroll; /\* Horizontal scrollbar always \*/

overflow-y: hidden; /\* Hide vertical overflow \*/

}

Great for horizontal scrolling menus or image sliders.

## ****3. Scrollbars with CSS (::-webkit-scrollbar):****

Customize scrollbars in WebKit-based browsers (Chrome, Edge, Safari):

.box::-webkit-scrollbar {

width: 8px;

}

.box::-webkit-scrollbar-thumb {

background-color: #4CAF50;

border-radius: 10px;

}

.box::-webkit-scrollbar-track {

background-color: #f1f1f1;

}

Personalize scrollbar appearance for a better UI experience.

## ****4. Overflow with Text:****

## Truncate overflowing text content with an ellipsis:

.text-ellipsis {

width: 200px;

white-space: nowrap;

overflow: hidden;

text-overflow: ellipsis;

}

Text that doesn’t fit will end with .…

## ****5.Overflow for Responsive Images/Videos:****

Ensure media stays within its container:

.responsive-container {

width: 100%;

height: 300px;

overflow: hidden;

}

.responsive-container img {

width: 100%;

height: auto;

}

Prevents media from stretching or breaking the layout.

## ****6. Overflow for Responsive Images/Videos:****

Ensure media stays within its container:

.responsive-container {

width: 100%;

height: 300px;

overflow: hidden;

}

.responsive-container img {

width: 100%;

height: auto;

}

Prevents media from stretching or breaking the layout.

**7.Advanced Scroll Effects (Smooth Scrolling):**

.scroll-box {

width: 300px;

height: 150px;

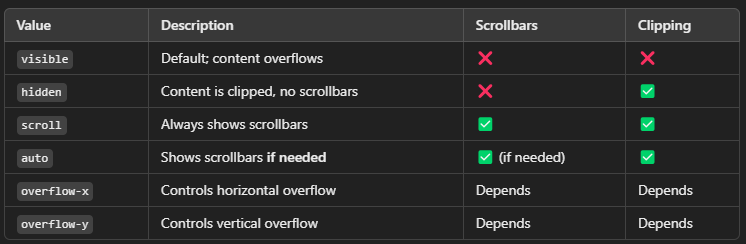
overflow: auto;

scroll-behavior: smooth;

}

Provides a smooth scrolling effect when navigating the element.

**8. Overflow Summary Table:**



## ****8. Real-World Use Cases****

* Scrollable content areas (chat windows, code blocks).
* Horizontal menus or carousels.
* Text truncation with ellipsis.
* Media containers to prevent layout breaks.
* Full-page scrolling effects (with overflow: hidden to lock scrolling