30 Days CSS Series

Beginner to Advanced CSS Roadmap

TABLE OF CONTENTS

Day	Topic
Day 1	What is CSS & Types of CSS
Day 2	CSS Syntax & Selectors
Day 3	Basic Selectors
Day 4	Grouping & Specificity
Day 5	CSS Colors
Day 6	CSS Units
Day 7	Typography Basics
Day 8	Box Model
Day 9	Width, Height & Overflow
Day 10	Display Property
Day 11	CSS Positioning
Day 12	Fixed vs Sticky
Day 13	Z-index & Stacking Context
Day 14	Float & Clear
Day 15	Introduction to Flexbox
Day 16	Flex Container Properties
Day 17	Flex Item Properties
Day 18	Flex Alignment & Ordering
Day 19	Introduction to CSS Grid
Day 20	Grid Container
Day 21	Grid Items & Alignment
Day 22	Backgrounds
Day 23	Borders & Shadows
Day 24	Pseudo-Classes
Day 25	Pseudo-Elements
Day 26	CSS Transitions
Day 27	CSS Transforms
Day 28	CSS Animations
Day 29	Media Queries & Responsive Design
Day 30	CSS Variables & Best Practices

Follow 👉 codingwithparvez

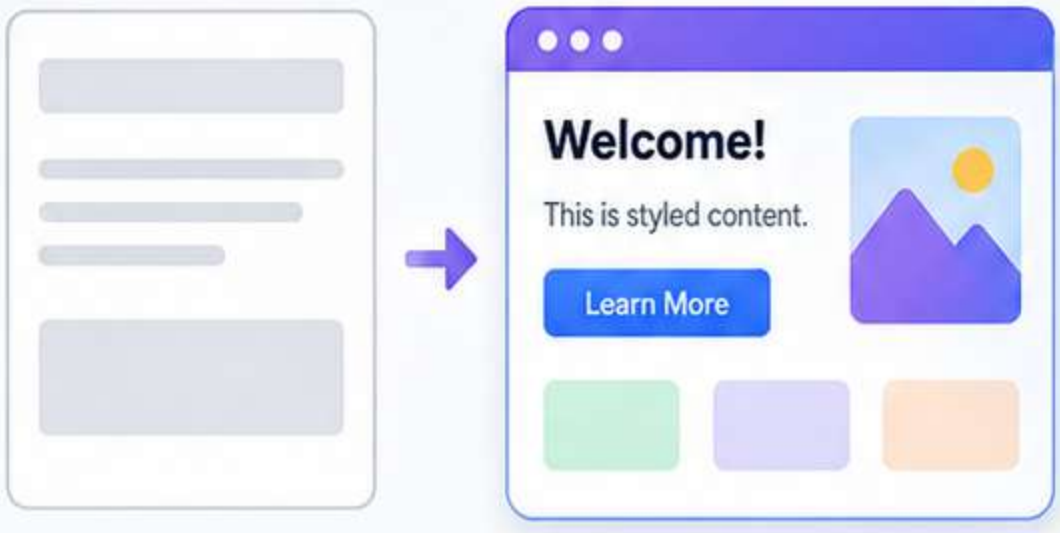
30 Days CSS Series

Day 1: What is CSS?

Styling the web made easy

1 What is CSS?

- CSS = Cascading Style Sheets
- Used to style HTML elements
- Controls colors, layout, spacing, fonts



2 Types of CSS

Inline CSS

- Applied directly to element
- Quick but not reusable

```html
<h1 style="color: blue;">
  Hello
</h1>
```

Internal CSS

- Inside <style> tag
- Used for single page

```html
<style>
  h1 { color: blue; }
</style>
```

External CSS

- Separate .css file
- Best practice

```html
<link rel="stylesheet"
  href="style.css">
```

3 Summary

Inline → quick

Internal → single page

External → scalable & best

Follow 👉 codingwithparvez

30 Days CSS Series

Day 2: CSS Syntax, Selectors & Comments

Understanding how CSS works

1 CSS Syntax

- Selector + Declaration block

Selector → Property → Value

- Property: value;

```css
h1 {
  color: blue;
  font-size: 20px;
}
```

h1 → selector

color → property

blue → value

2 Selectors Overview

Element Selector

```css
p { color: red; }
```

Class Selector

```css
.box { padding: 10px;
```

ID Selector

```css
#header {
  background: black;
}
```

Group Selector

```css
h1, p {
  color: green;
}
```

3 Comments

- Used to explain code
- Not executed by browser

```css
/* This is a comment */
```

Summary

- **Syntax** → structure of CSS
- **Selectors** → target elements
- **Comments** → improve readability

Follow 👉 codingwithparvez

30 Days CSS Series

Day 3: Basic Selectors

Element, Class & ID Selectors

1 What are Selectors?

- Used to target HTML elements
- Apply styles to specific elements

```
<h1 id="title">Hello World</h1>
<p class="text">This is a paragraph.</p>
<div class="box">Content inside box.</div>
<p class="text">Another paragraph.</p>
```

ID Selector `#title`

Class Selector `.text`

Class Selector `.box`

2 Types of Basic Selectors

E Element Selector

- Targets all elements of a type

CSS

```
p {
  color: red;
}
```

HTML

```
<p>First paragraph.</p>
<p>Second paragraph.</p>
<p>Third paragraph.</p>
```

All `<p>` elements are selected

. Class Selector

- Targets elements with class
- Reusable

CSS

```
.box {
  padding: 10px;
}
```

HTML

```
<div class="box">Box 1</div>
<div>Not a box</div>
<div class="box">Box 2</div>
```

Elements with `class="box"` are selected

ID Selector

- Targets unique element
- Used once

CSS

```
#header {
  background: black;
}
```

HTML

```
<div id="header">
  Website Header
</div>
<div>Other content</div>
```

Only the element with `id="header"` is selected

3 Summary

E **Element** → by tag
Targets all elements of the same type.

. **Class** → reusable
Targets elements with the same class.

**ID** → unique
Targets a single, unique element.

Follow 👉 codingwithparvez

30 Days CSS Series

Day 4: Grouping, Universal & Specificity

Smarter ways to target elements

1 Grouping Selector

- Apply same styles to multiple elements
- Reduces repetition

HTML

<h1> </h1>

CSS

```
h1, p, div {
  color: blue;
}
```

One rule for many elements!

2 Universal Selector (*)

- Targets ALL elements
- Useful for reset styles

HTML

<header> Header </header>
<nav> Menu </nav>
<main> Content </main>
 Paragraph
<footer> Footer </footer>

CSS

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

Affects every element on the page

3 Specificity Basics

- Determines which style is applied

Priority Order (High → Low)

CSS Example

```
#id { color: red; }
.class { color: blue; }
p { color: green; }
```

ID Selector (Highest)

Class Selector

Element Selector (Lowest)

Result: ID wins (red applied)
Because ID has the highest specificity.

1 Grouping → cleaner code

2 Universal → apply to all

3 Specificity → controls priority

Follow 👉 codingwithparvez

30 Days CSS Series

Day 5: CSS Colors

Named, HEX, RGB & RGBA

1 What are CSS Colors?

- Used to style text, background, borders
- Multiple formats available

Colors bring life to your website!

2 Types of Colors

Named Colors

Predefined color names

red

Example

```css
color: red;
```

Easy to remember and use

HEX

Hexadecimal values

#ff0000

Example

```css
color: #ff0000;
```

Precise and widely used

RGB

Red, Green, Blue values

R	G	B
255	0	0

Example

```css
color: rgb(255, 0, 0);
```

Flexible and great for dynamic designs

RGBA

RGB + Alpha (opacity)

R	G	B	A
255	0	0	0.5

Example

```css
color: rgba(255, 0, 0, 0.5);
```

Adds transparency control

3 Summary

- **Named** → simple
- **HEX** → precise
- **RGB** → flexible
- **RGBA** → transparency

Follow 👉 codingwithparvez

30 Days CSS Series

Day 6: CSS Units

px, %, em, rem, vw, vh explained

1 What are CSS Units?

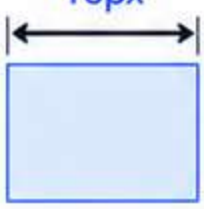
- Define size of elements, text, spacing
- Two types: Absolute & Relative

Box Size Example

2 Common CSS Units

1 px (Pixels)

Fixed size




```css
font-size: 16px;
```

2 % (Percentage)

Relative to parent

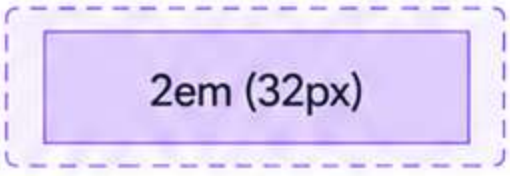
Parent (100%)


```css
width: 50%;
```

3 em

Relative to parent font-size

Parent font-size: 16px




```css
font-size: 2em;
```

4 rem

Relative to root font-size

Root font-size: 16px


```css
font-size: 1.5rem;
```

5 vw (Viewport Width)

Based on screen width


```css
width: 50vw;
```

6 vh (Viewport Height)

Based on screen height


```css
height: 50vh;
```

3 Summary

px	%	em	rem	vw	vh
px → fixed	% → parent-based	em → parent font	rem → root font	vw → viewport width	vh → viewport height

Follow 👉 codingwithparvez

30 Days CSS Series

Day 7: Typography Basics

font-family, font-size, font-weight

1 What is Typography?

- Controls how text looks on a webpage
- Improves readability and design

Same text, different styles

Hello CSS! Hello CSS! *Hello CSS!*

Serif | Sans-serif | Cursive

2 Core Typography Properties

1 font-family

Defines typeface

```
CSS
font-family: Arial, sans-serif;
```

Same word, different fonts

Typography	Arial
Typography	Georgia
Typography	Courier New
Typography	Times New Roman

2 font-size

Controls text size

```
CSS
font-size: 18px;
```

Small vs Large

This is small text	12px
This is medium text	18px
This is large text	32px

3 font-weight

Controls thickness

```
CSS
font-weight: bold;
```

Light vs Bold

Typography	300 (Light)
Typography	400 (Normal)
Typography	600 (Semi-bold)
Typography	700 (Bold)

3 Summary

Aa font-family → style
Choose the right font to set the tone.

A↕ font-size → size
Make text readable and well-balanced.

B font-weight → thickness
Add emphasis and visual hierarchy.

Follow 👉 codingwithparvez

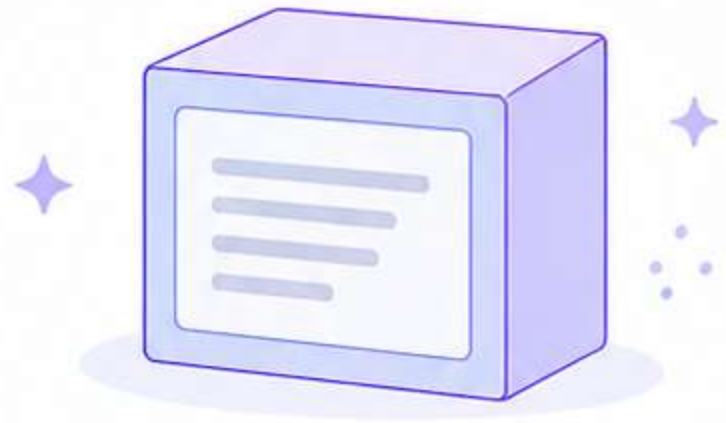
30 Days CSS Series

Day 8: Box Model

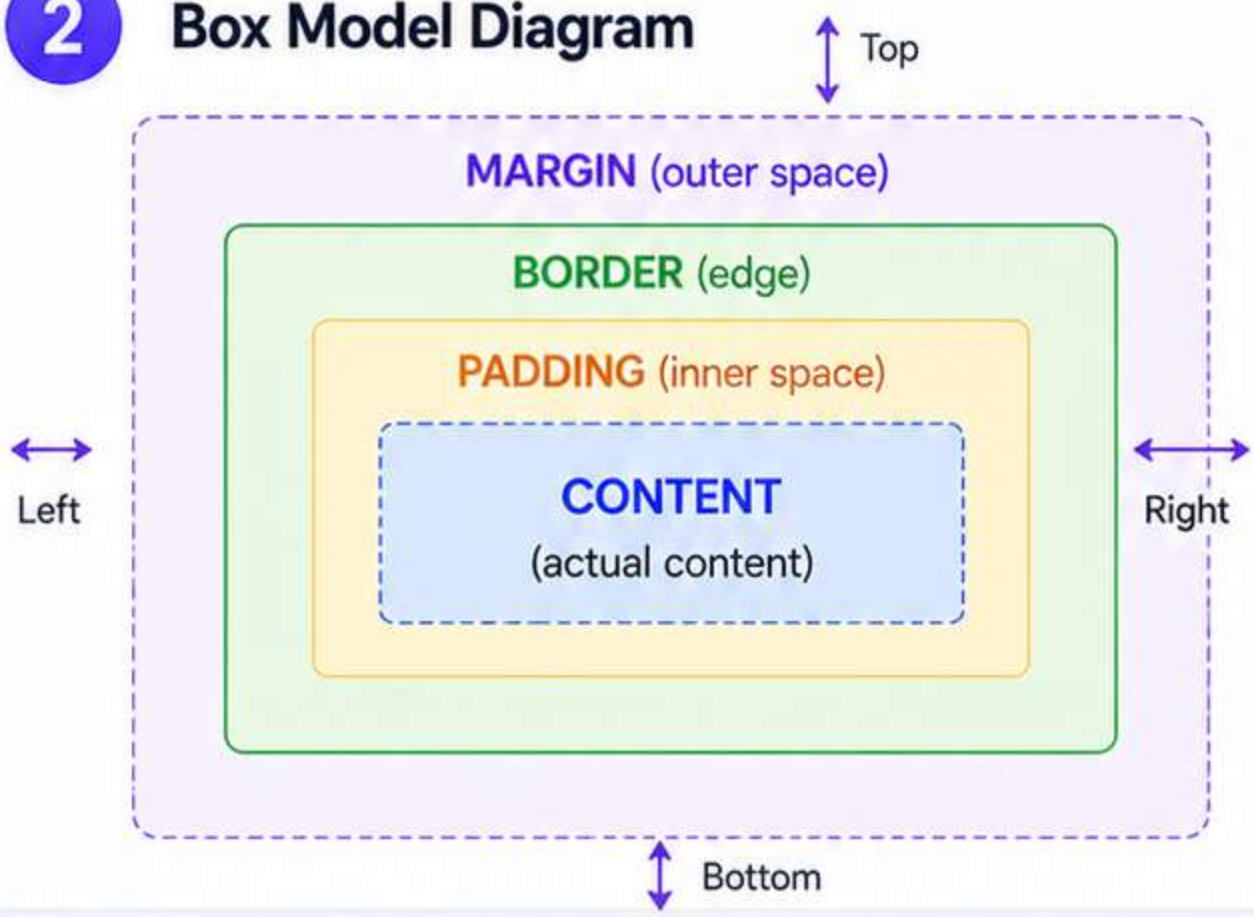
Content, Padding, Border, Margin

1 What is Box Model?

- Every element is a box
- Consists of content, padding, border, margin



2 Box Model Diagram



MARGIN
Space outside the border (transparent)

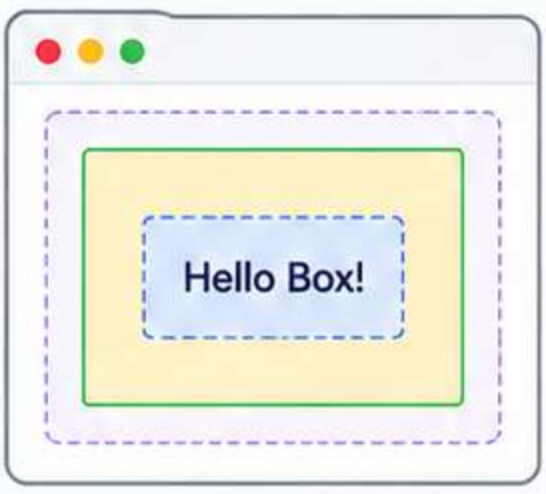
BORDER
The edge of the box

PADDING
Space between content and border

CONTENT
Actual content of the box

3 Code Example

```css
.box {
  margin: 10px;
  border: 2px solid black;
  padding: 15px;
}
```



Summary

- **Content** → actual data
- **Padding** → space inside
- **Border** → edge
- **Margin** → space outside

Follow 👉 codingwithparvez

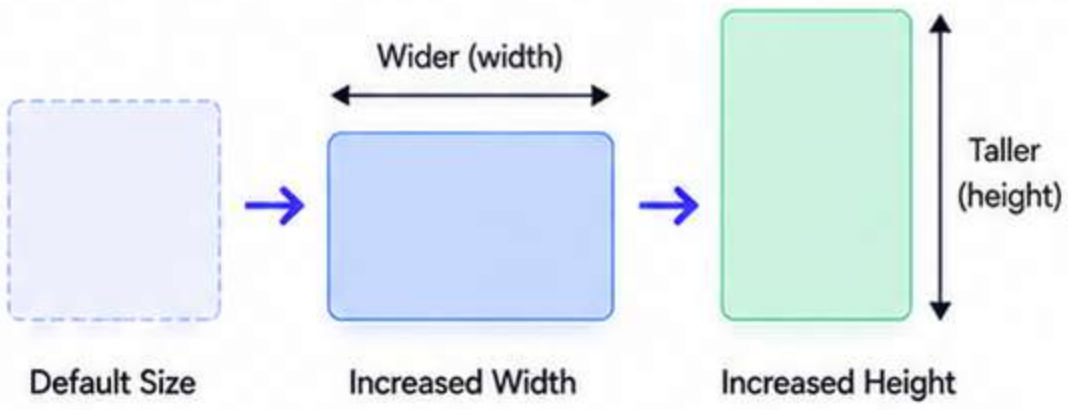
30 Days CSS Series

Day 9: Width, Height & Overflow

Control size and content behavior

1 What do these properties do?

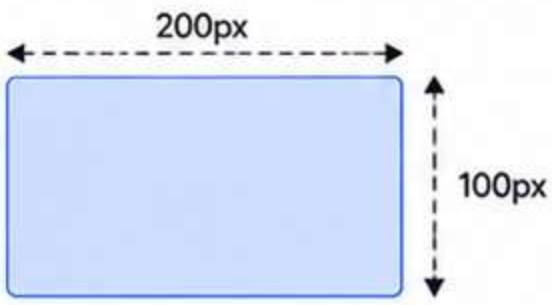
- **width** → controls element width
- **height** → controls element height
- **overflow** → controls extra content



2 Core Properties

Width & Height

Set element size

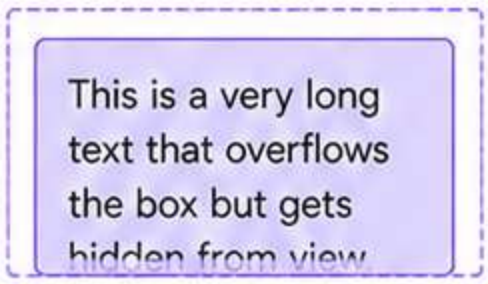


CSS

```css
.box {
  width: 200px;
  height: 100px;
}
```

Overflow: hidden

Hides extra content

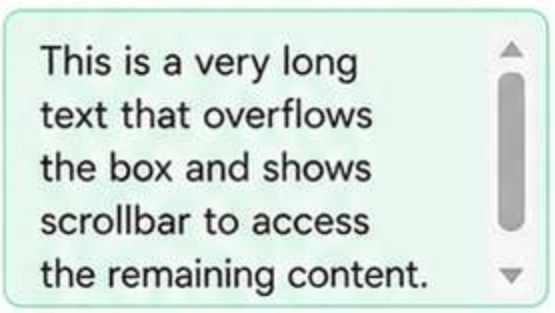


CSS

```css
.box {
  overflow: hidden;
}
```

Overflow: scroll / auto

Adds scroll when needed



CSS

```css
.box {
  overflow: auto;
}
```

3 Summary

- **width/height** → size control
- **overflow** → content control

Follow 👉 codingwithparvez

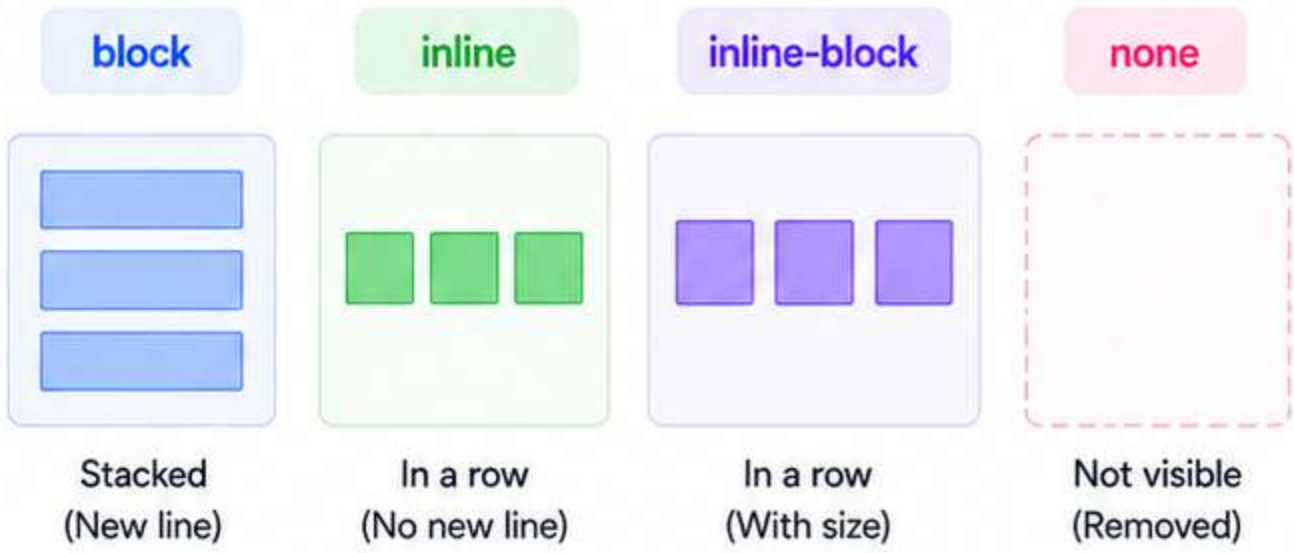
30 Days CSS Series

Day 10: Display Property

✦ block, inline, inline-block, none ✦

1 What is display?

- Controls how elements behave in layout
- Affects size, spacing, and positioning



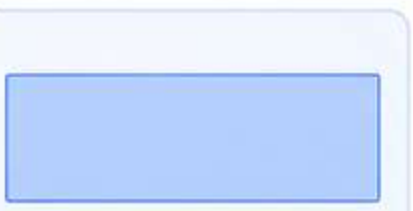
2 Display Types

block

- Takes full width
- Starts on new line

CSS

```
display: block;
```



inline

- Takes only content width
- No new line

CSS

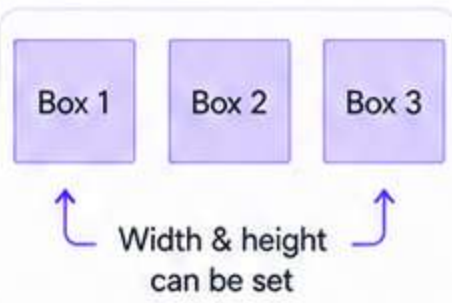
```
display: inline;
```

inline-block

- Inline but supports width/height

CSS

```
display: inline-block;
```



none

- Element removed from layout
- Not visible

CSS

```
display: none;
```

Element is hidden and takes no space

3 Summary

block	inline	inline-block	none
→ full width	→ content width	→ flexible	→ hidden

♥ Follow 👉 codingwithparvez ♥

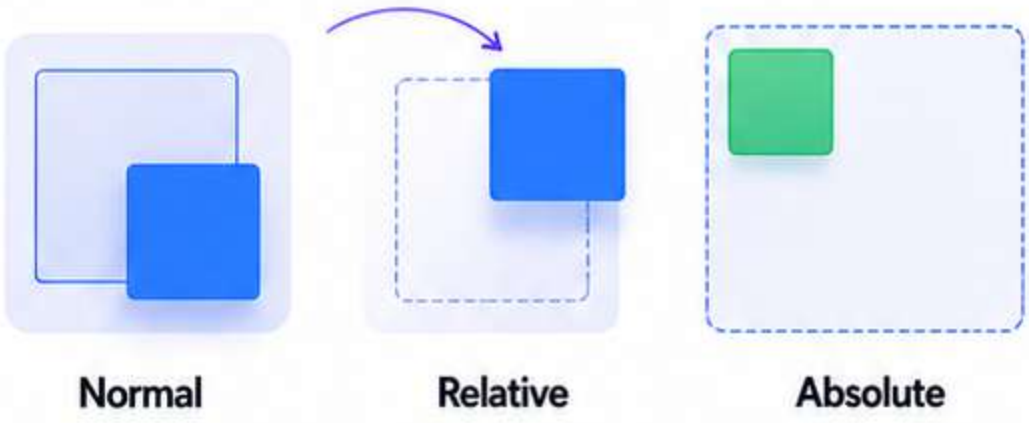
30 Days CSS Series

Day 11: CSS Positioning (static, relative, absolute)

Control where elements appear

1 What is Positioning?

- Controls element placement on the page
- Uses top, left, right, bottom to move



2 Position Types

static

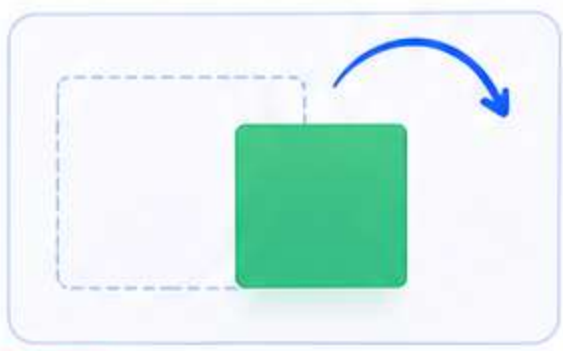
- Default position
- Follows normal document flow

CSS

```
position: static;
```

relative

- Positioned relative to itself
- Can move using top/left



CSS

```
position: relative;
top: 10px;
left: 10px;
```

absolute

- Positioned relative to parent
- Removed from normal flow

CSS

```
position: absolute;
top: 0;
left: 0;
```

3 Summary

static → default flow

relative → move from self

absolute → move inside parent

Tip

Positioning unlocks powerful layouts when combined with Flexbox and Grid!

Follow 👉 codingwithparvez

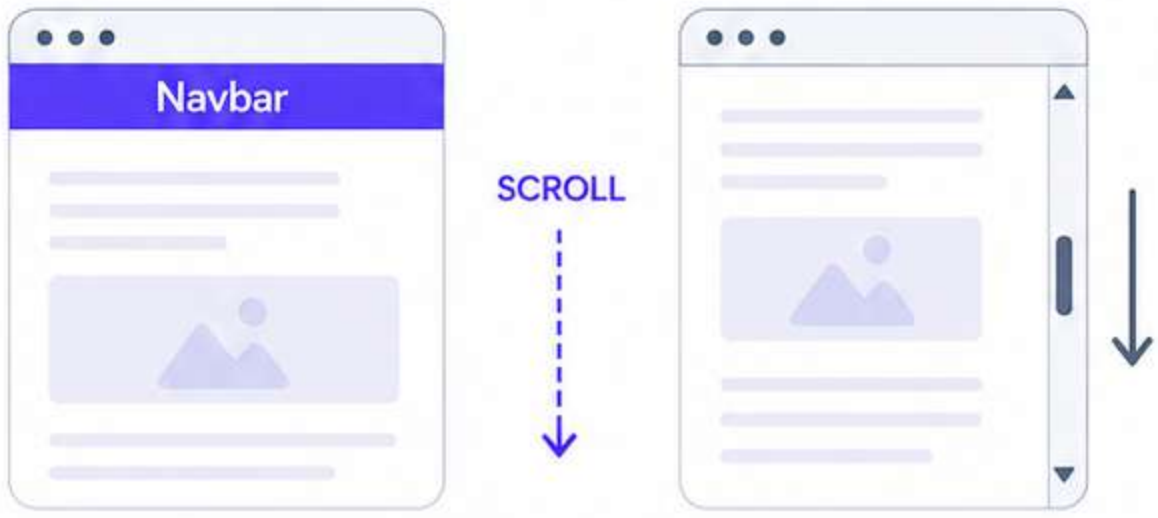
30 Days CSS Series

Day 12: Fixed vs Sticky

Understand scrolling behavior

1 What is positioning on scroll?

- Controls how elements behave during scroll
- Used for navbars, headers



2 Fixed vs Sticky

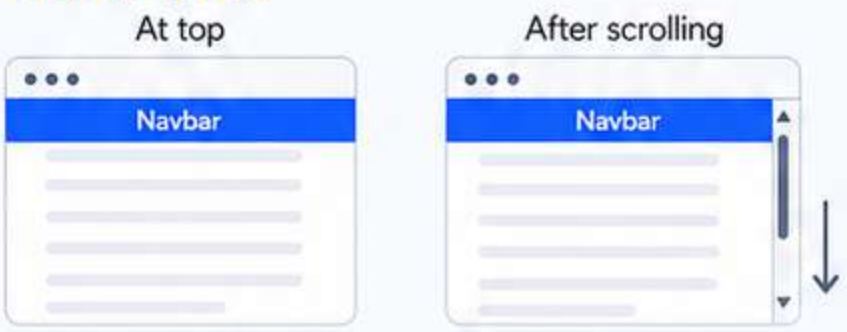
Fixed

- Always fixed to viewport
- Removed from normal flow
- Stays visible at all times

CSS

```css
position: fixed;
top: 0;
```

How it works



Navbar stays fixed at the top while content scrolls

Sticky

- Acts normal until scroll
- Sticks when it reaches top
- Depends on parent container

CSS

```css
position: sticky;
top: 0;
```

How it works

Navbar scrolls with content until it reaches top, then sticks in place

3 Summary

Fixed
→ always fixed
Stays visible regardless of scrolling

Sticky
→ scroll-based behavior
Sticks when it reaches the top

Note
Sticky requires parent without `overflow: hidden`

Follow 👉 codingwithparvez

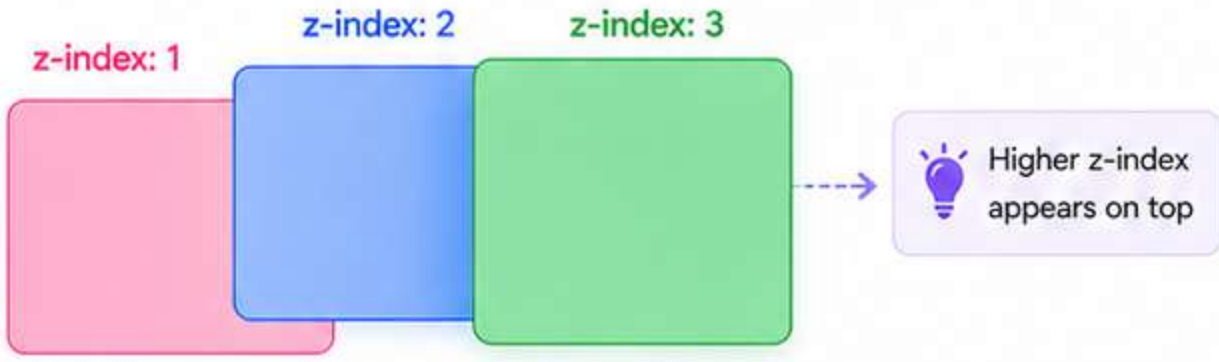
30 Days CSS Series

Day 13: Z-Index & Stacking Context

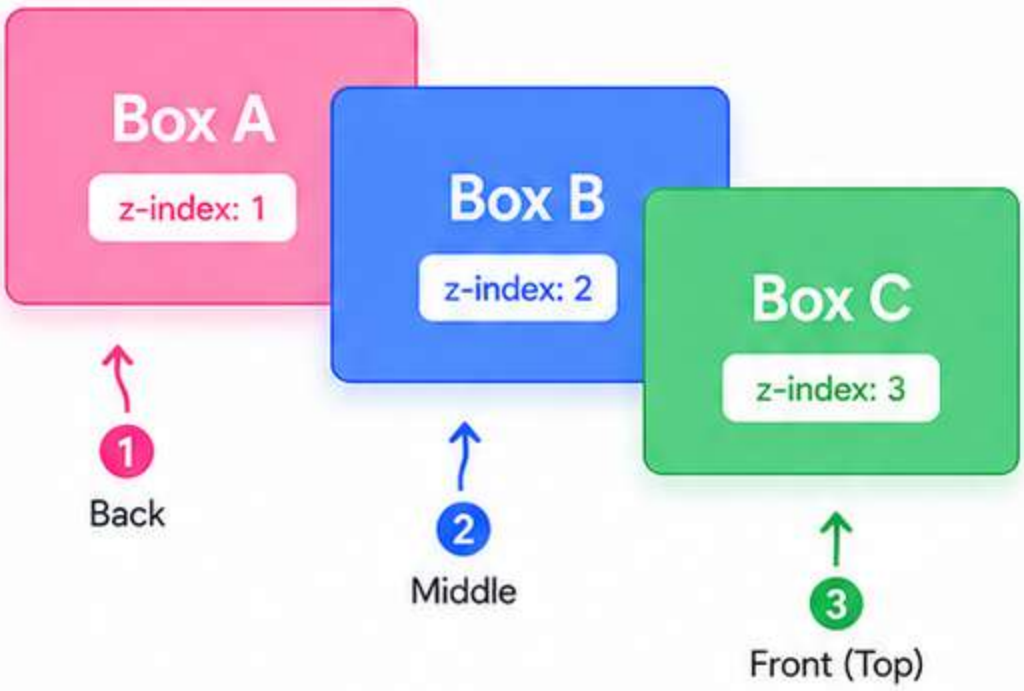
Control which element appears on top

1 What is z-index?

- Controls vertical stacking (front/back)
- Works with positioned elements



2 Layering Example



HTML

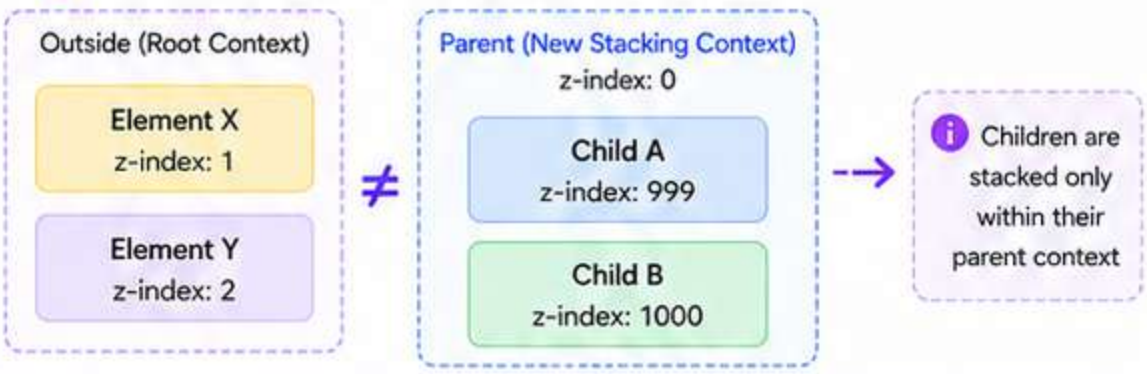
```html
<div class="box1"></div>
<div class="box2"></div>
<div class="box3"></div>
```

CSS

```css
.box1 {
  position: relative;
  z-index: 1;
}
.box2 {
  position: relative;
  z-index: 2;
}
.box3 {
  position: relative;
  z-index: 3;
}
```

3 Stacking Context Basics

- z-index works inside stacking context
- New context created by positioned elements
- Child elements cannot escape parent stacking



Summary

- Higher z-index → on top
- Needs position (relative/absolute/fixed)
- Context affects layering

Follow 👉 codingwithparvez

30 Days CSS Series

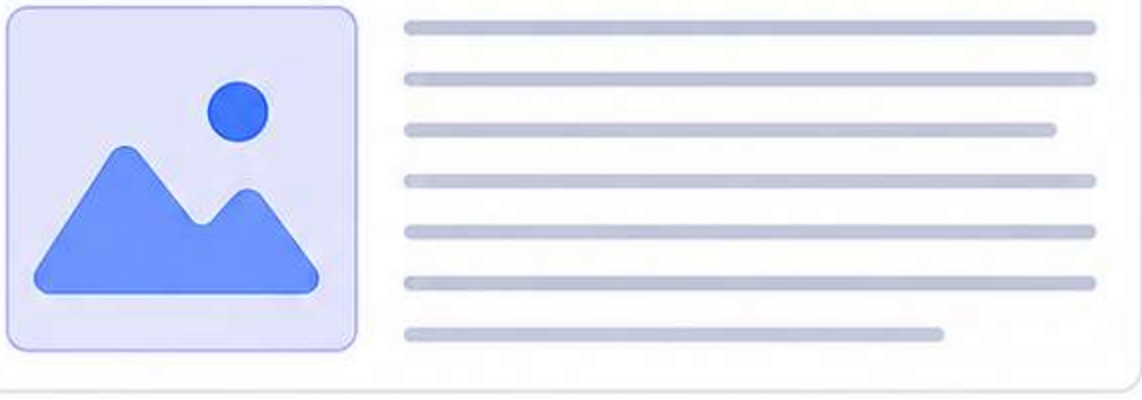
Day 14: Float & Clear

(Legacy Layout)

✦ Old layout techniques in CSS ✦

1 What is Float?

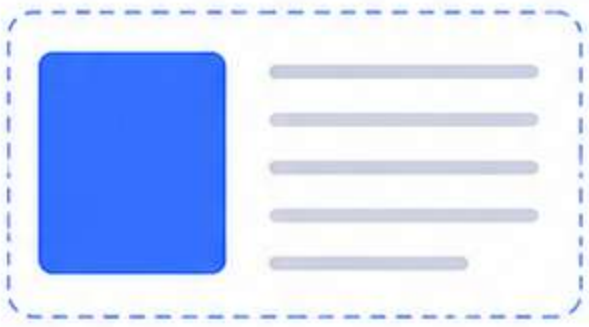
- Used to position elements left or right
- Text wraps around floated elements



2 Float Examples

float: left

Element moves to left

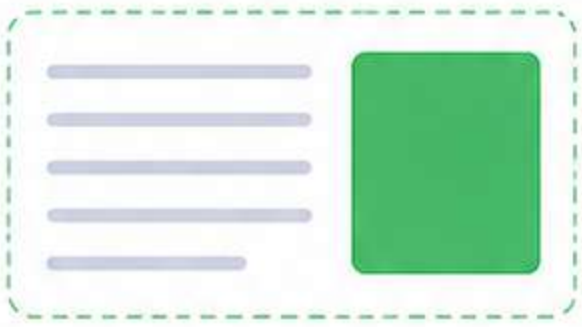


CSS

```
float: left;
```

float: right

Element moves to right

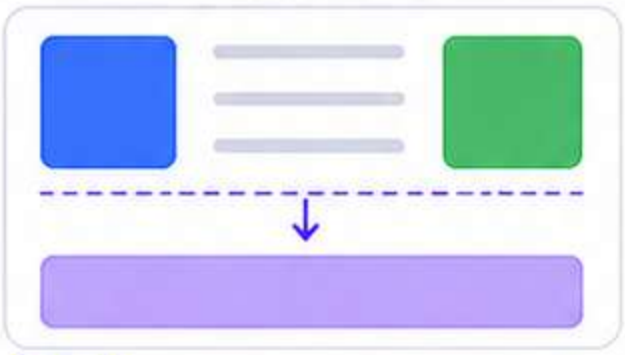


CSS

```
float: right;
```

clear

Stops wrapping
Moves element below floated items



CSS

```
clear: both;
```

3 Summary

- **float** → left/right positioning
- **clear** → fix layout flow
- Legacy method (use **Flexbox/Grid** today)

Note

Float was used for page layouts in the past. Today, use **Flexbox** or **Grid** for modern layouts.

Follow 👉 codingwithparvez

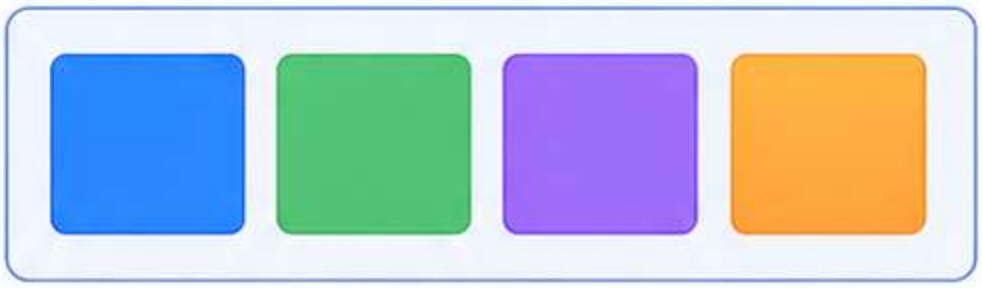
30 Days CSS Series

Day 15: Introduction to Flexbox

One-dimensional layout system

1 What is Flexbox?

- Layout system for aligning items
- Works in row or column
- Makes layouts easier

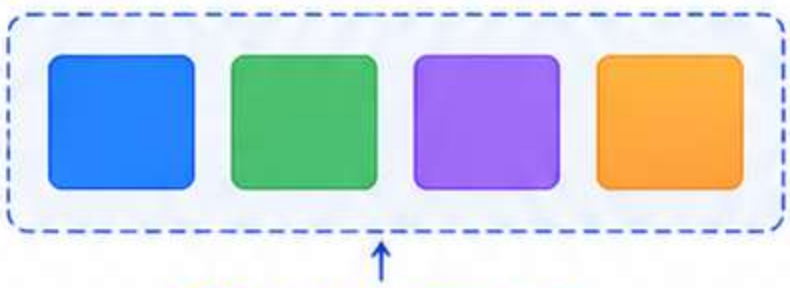


Items aligned in a row

2 Flex Container & Items

Flex Container

Enables flex layout



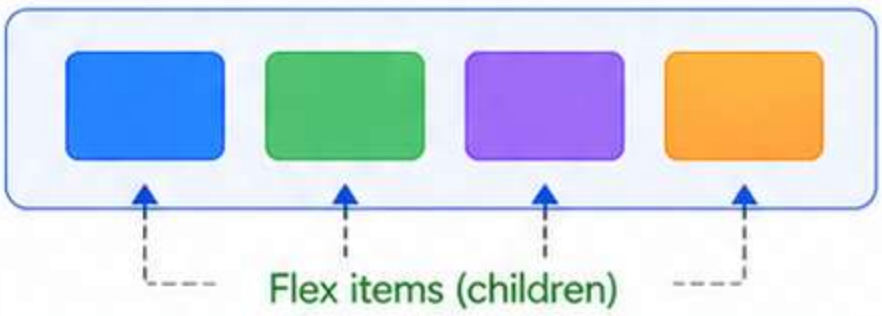
This is the flex container

CSS

```css
.container {
  display: flex;
}
```

Flex Items

Children inside container
Can be aligned & spaced



Alignment Example

```css
justify-content: center;
align-items: center;
```

3 Summary

- `display: flex` → enable flex
- Container → parent
- Items → children

Follow 👉 codingwithparvez

30 Days CSS Series

Day 16: Flex Container Properties

✦ justify-content & align-items ✦

1 What do these properties do?

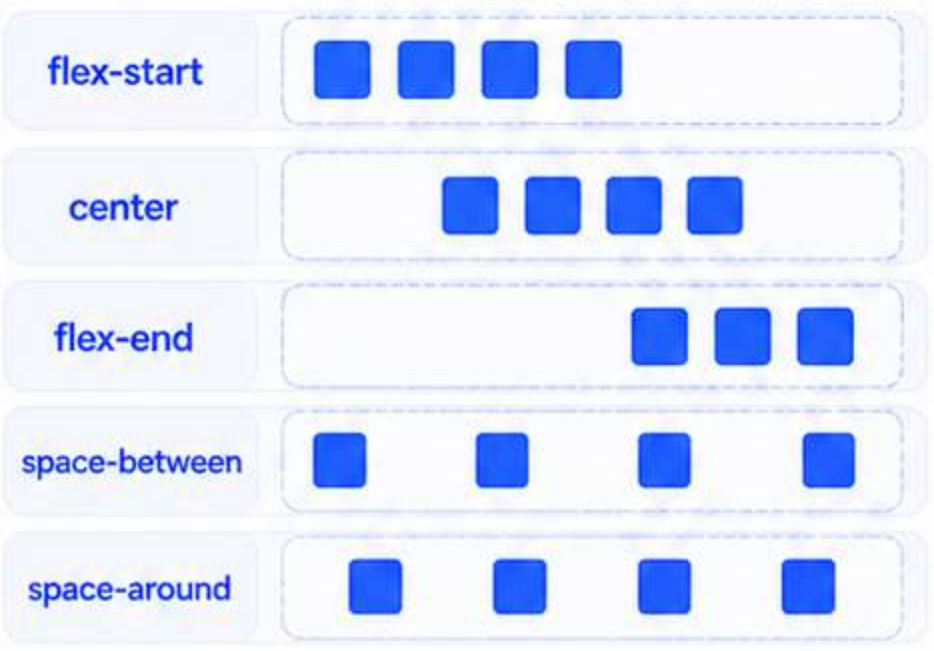
- Control alignment of flex items
- Horizontal & vertical positioning

2 Core Properties

justify-content

Aligns items on main axis (horizontal)

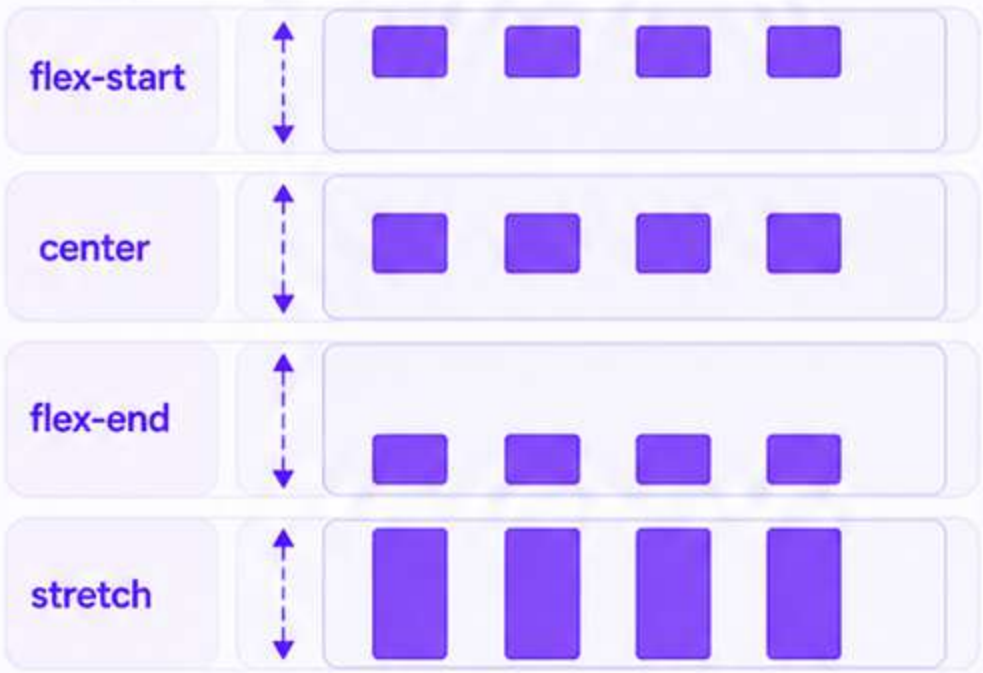
```
justify-content: center;
```



align-items

Aligns items on cross axis (vertical)

```
align-items: center;
```



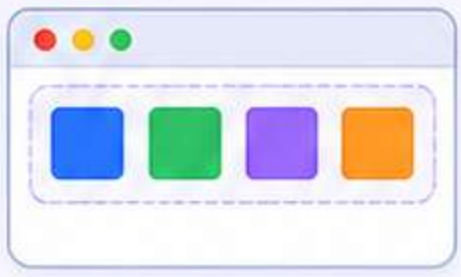
3 Summary

justify-content → horizontal alignment

Controls spacing & alignment along the main axis.

align-items → vertical alignment

Controls alignment along the cross axis.



Follow 👉 codingwithparvez

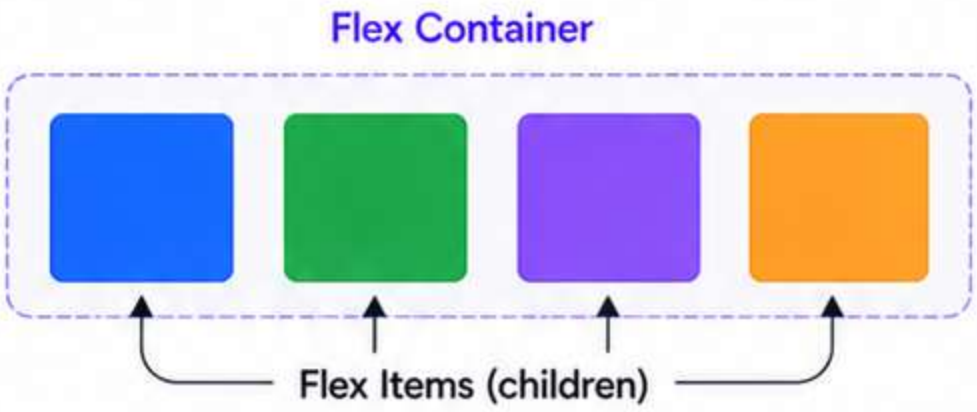
30 Days CSS Series

Day 17: Flex Item Properties

✦ flex-grow, flex-shrink, flex-basis ✦

1 What are Flex Item Properties?

- Control size & behavior of items
- Applied to children inside flex container



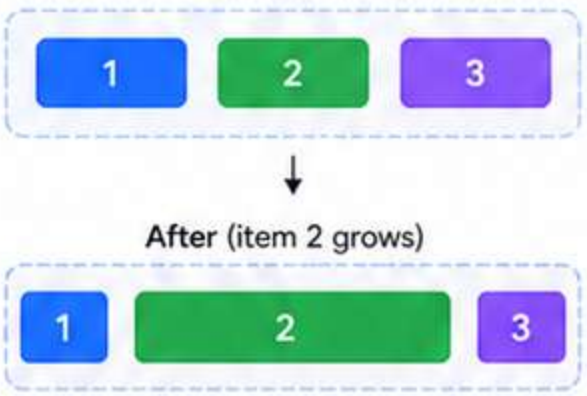
2 Core Properties

flex-grow

Controls how items expand

```
flex-grow: 1;
```

Before (equal sizes)



After (item 2 grows)

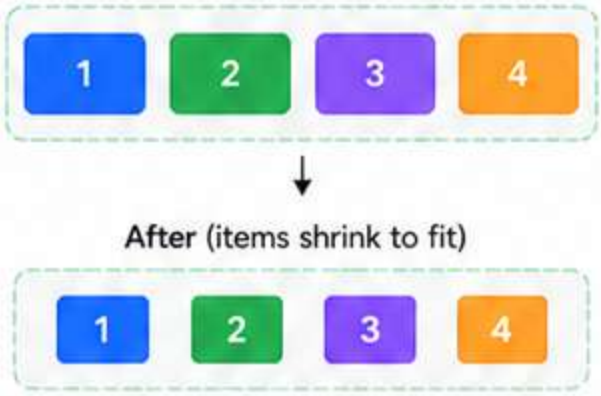
Item with higher flex-grow takes more available space.

flex-shrink

Controls how items shrink

```
flex-shrink: 1;
```

Before (items overflow)



After (items shrink to fit)

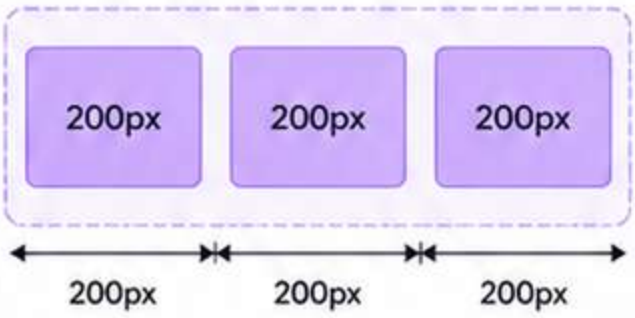
Items shrink to fit the container when space is limited.

flex-basis

Defines initial size

```
flex-basis: 200px;
```

Each item starts at 200px



Sets the initial size before extra space is distributed.

Shorthand (All in One)

Set grow, shrink & basis together

```
flex: 1 1 200px;
```

→ flex-grow
→ flex-shrink
→ flex-basis

Same as:

```
flex-grow: 1;
flex-shrink: 1;
flex-basis: 200px;
```

3 Summary

- **flex-grow** → expand — Takes extra space in the container
- **flex-shrink** → shrink — Shrinks items when space is not enough
- **flex-basis** → initial size — Sets the starting size of the item

Follow 👉 codingwithparvez

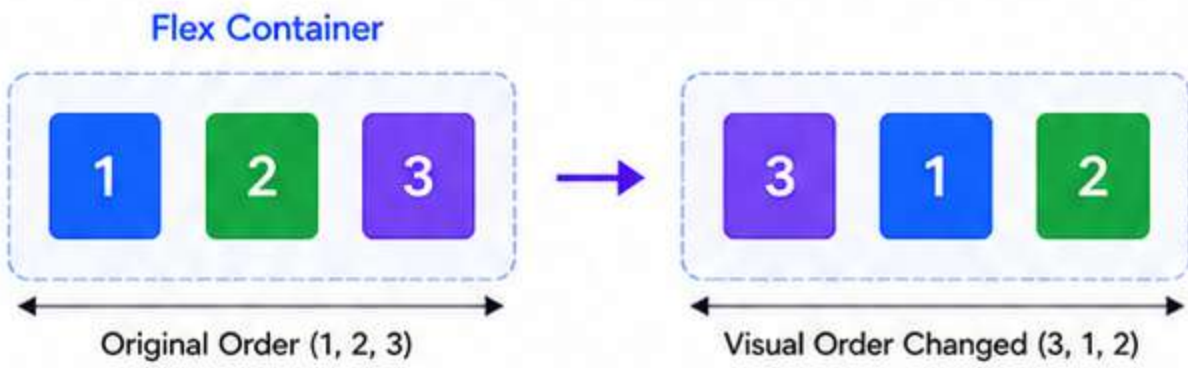
30 Days CSS Series

Day 18: Flexbox Alignment & Ordering

✦ Align items and control order ✦

1 What does this control?

- Align items inside container
- Change visual order of elements

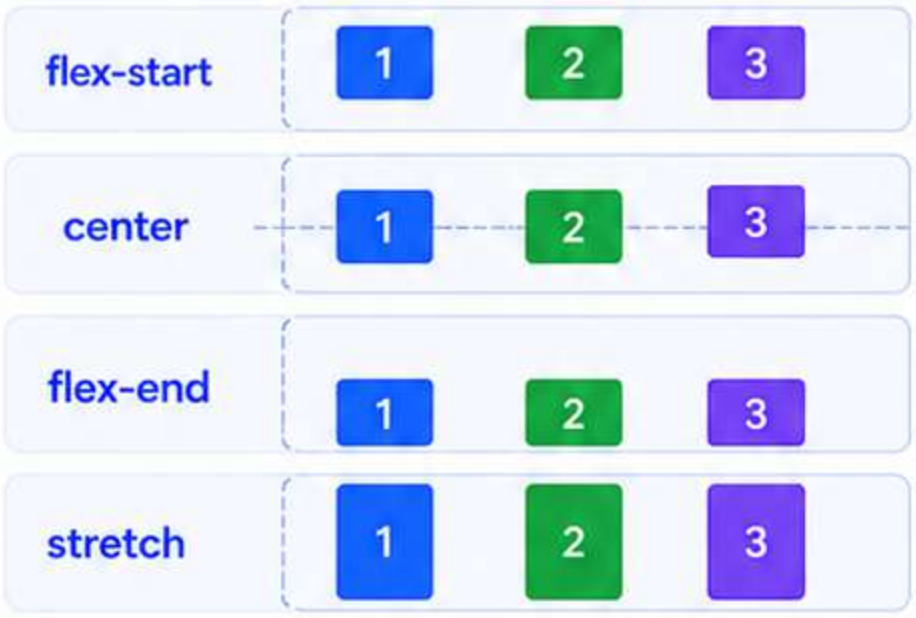


2 Core Properties

Alignment (align-self)

Align individual items

```
align-self: center;
```

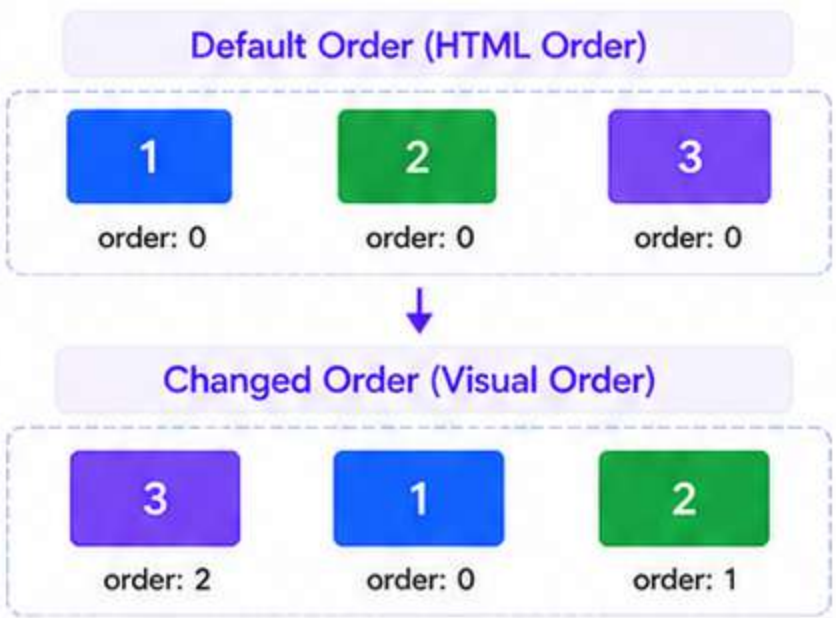


Affects alignment of a single item on the cross axis (vertical).

Ordering (order)

Change item order visually

```
order: 2;
```



Default order = 0
Lower numbers appear first.

3 Summary

align-self → align single item
Controls vertical alignment of an individual item.

order → change position
Changes the visual order of items.

Follow 👉 codingwithparvez

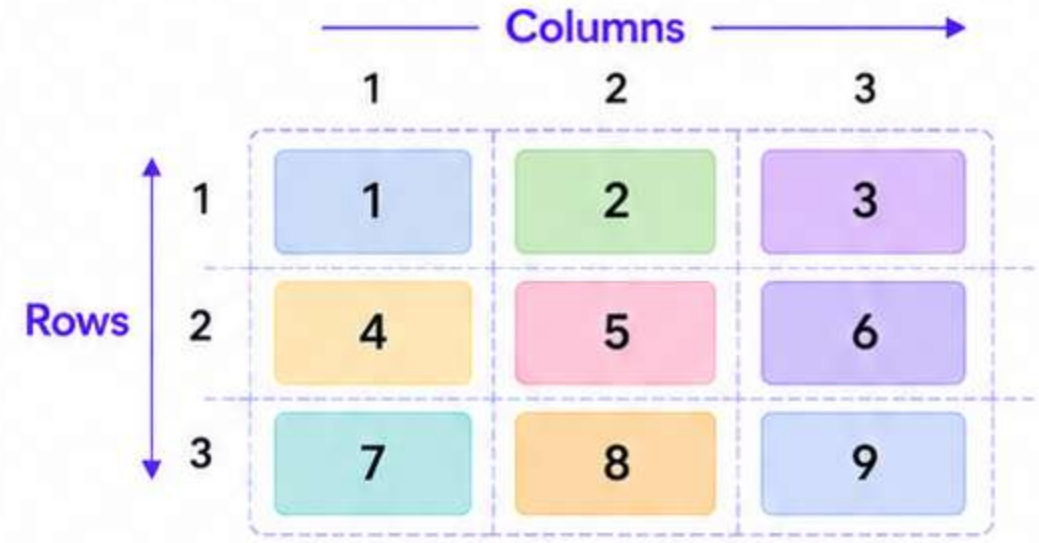
30 Days CSS Series

Day 19: Introduction to CSS Grid

✦ Two-dimensional layout system ✦

1 What is CSS Grid?

- Layout system for rows and columns
- Perfect for complex layouts
- More control than Flexbox

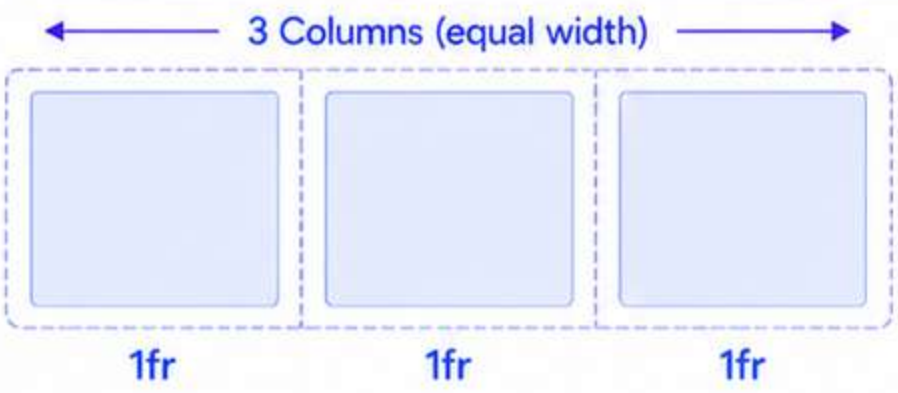


2 Grid Container & Items

Grid Container

Enables grid layout

```css
.container {
  display: grid;
  grid-template-columns: repeat(3, 1fr);
}
```

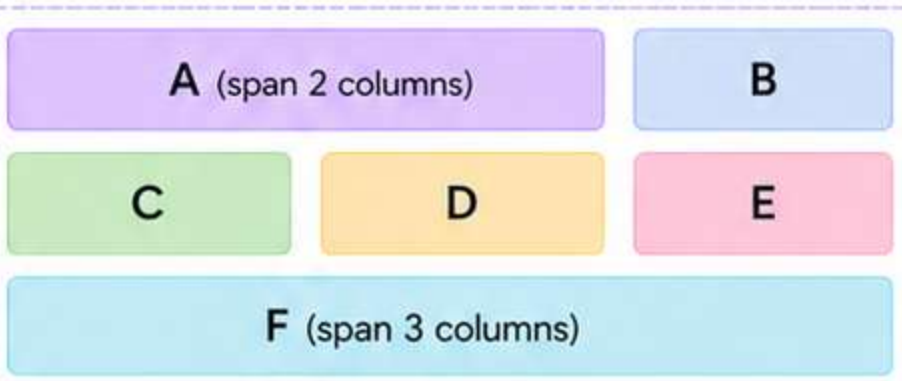


Creates 3 equal columns.
You can also define rows.

Grid Items

Children placed inside grid
Can span rows/columns

```css
/* Item spans across 2 columns */
.item {
  grid-column: span 2;
}
```



Items can span across multiple columns or rows.

3 Summary

- **display: grid** → enable grid layout
- **Rows + Columns** → layout control
- **Items** → placed inside grid

Follow 👉 codingwithparvez

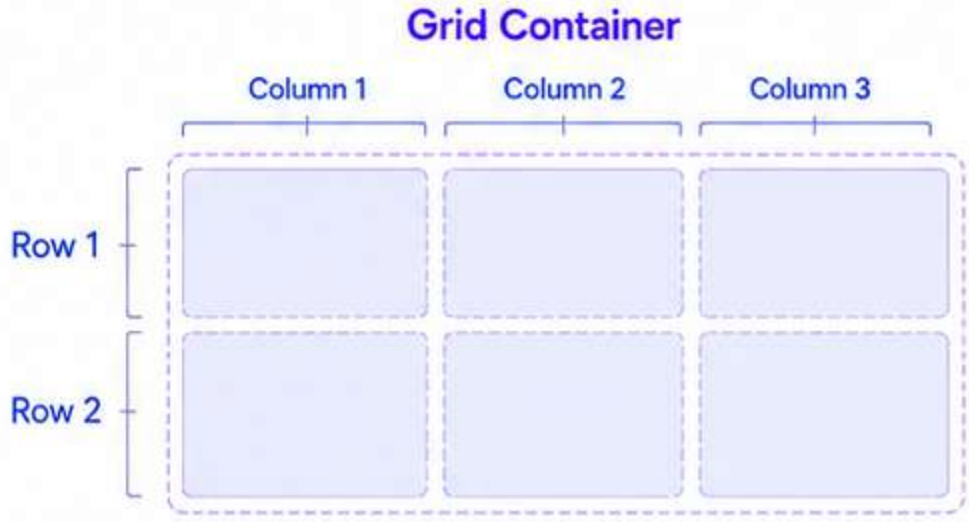
30 Days CSS Series

Day 20: Grid Container

✦ grid-template, rows & columns ✦

1 What is a Grid Container?

- Parent element using `display: grid`
- Defines rows and columns



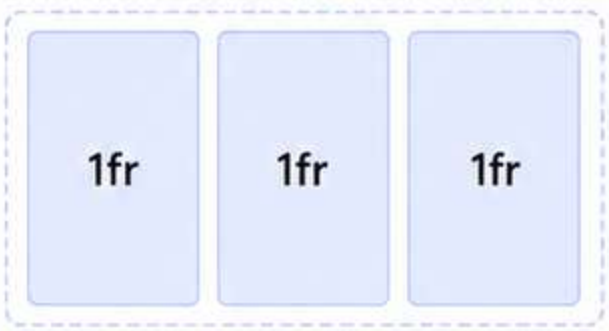
2 Core Grid Properties

grid-template-columns

Defines column structure

```
grid-template-columns:
repeat(3, 1fr);
```

⟵ 3 Columns (equal width) ⟶

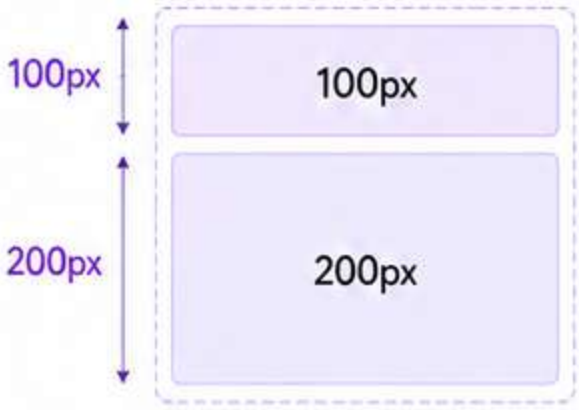


Each column takes equal space with `1fr` unit.

grid-template-rows

Defines row structure

```
grid-template-rows:
100px 200px;
```

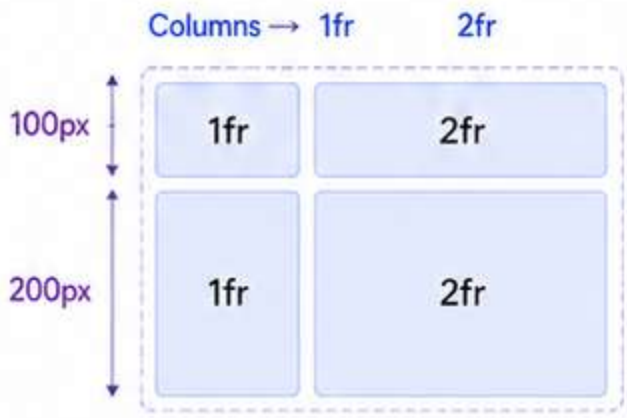


First row is `100px` tall, second row is `200px` tall.

grid-template (shorthand)

Combines rows & columns

```
grid-template:
100px 200px / 1fr 2fr;
```



Rows: `100px, 200px`
Columns: `1fr` and `2fr`

3 Summary

columns → vertical layout
Divide space vertically.

rows → horizontal layout
Divide space horizontally.

template → shorthand
Combine rows & columns.

Follow 👉 codingwithparvez

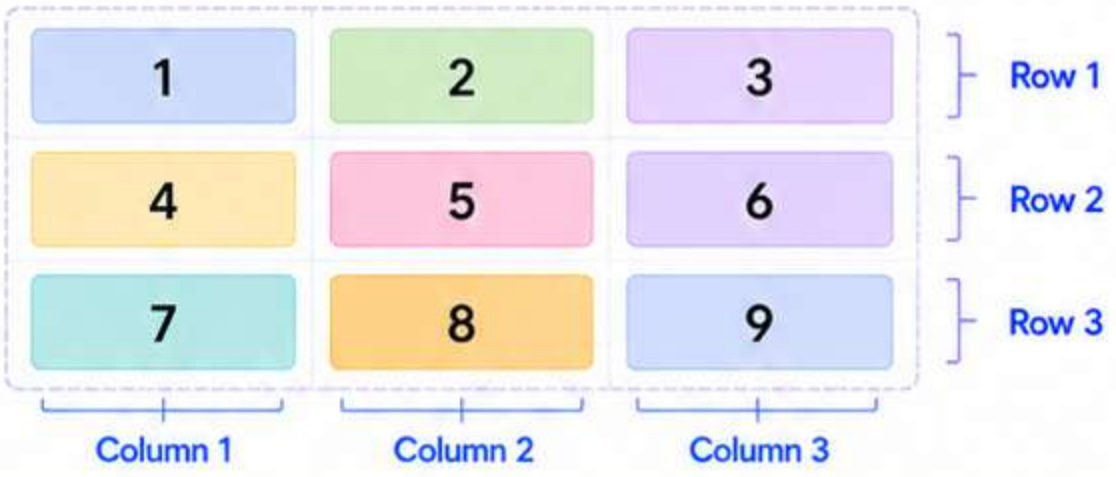
30 Days CSS Series

Day 21:
Grid Items, Gap & Alignment

✦ Control spacing and positioning in grid ✦

1 What does this control?

- Position and spacing of grid items
- Align items inside grid cells

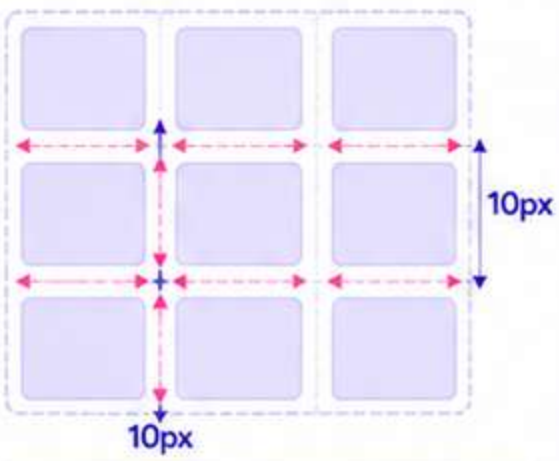


2 Core Properties

gap

Space between rows & columns

```
gap: 10px;
```

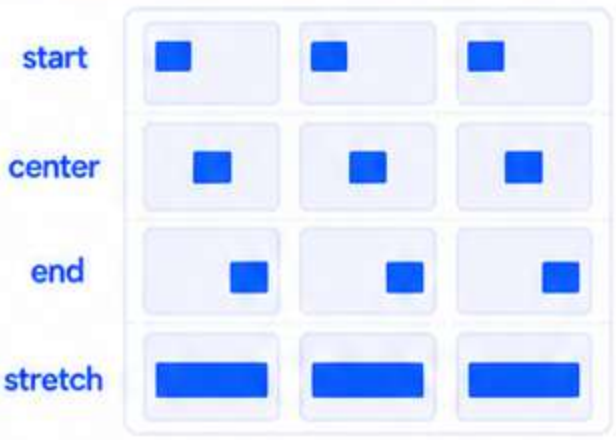


Adds equal space between every row and column.

justify-items

Align items horizontally

```
justify-items: center;
```

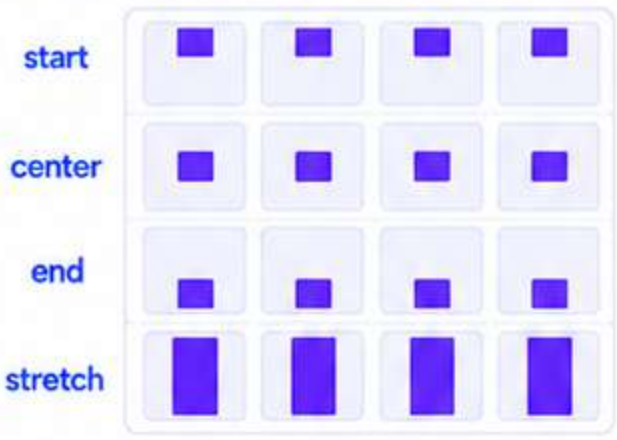


Controls horizontal alignment of items inside their cells.

align-items

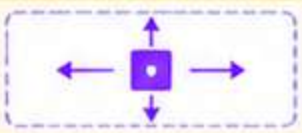
Align items vertically

```
align-items: center;
```



Controls vertical alignment of items inside their cells.

Use `place-items: center;` to center items both horizontally and vertically.



3 Summary

gap → spacing
Adds space between rows and columns.

justify-items → horizontal alignment
Controls how items align horizontally in their cells.

align-items → vertical alignment
Controls how items align vertically in their cells.

Follow 👉 codingwithparvez

30 Days CSS Series

Day 22: Backgrounds

color, image & gradient

1 What are Backgrounds?

- Used to style element background
- Adds color, images, and effects

Color

Image

Gradient

2 Types of Backgrounds

Background Color

Adds solid color

```
background-color: #4CAF50;
```

Simple and effective for clean designs.

Background Image

Adds image as background

```
background-image: url("bg.jpg");
background-size: cover;
```

Great for hero sections and creative layouts.

Gradient

Smooth color transition

```
background: linear-gradient(
to right, #00f, #0ff);
```

Adds depth and makes UI more attractive.

3 Summary

color → solid background

image → visual background

gradient → stylish effect

Choose the right background to enhance your design and user experience.

Follow 👉 codingwithparvez

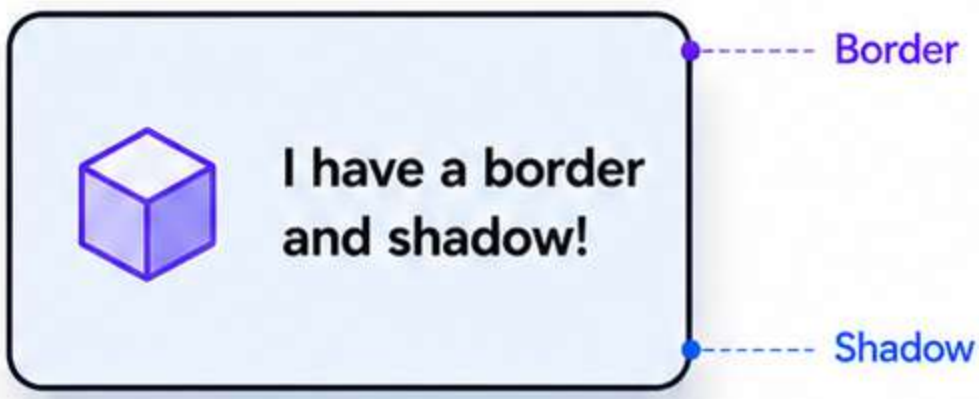
30 Days CSS Series

Day 23:
Borders & Shadows

✦ border, box-shadow, text-shadow ✦

1 What are Borders & Shadows?

- Borders define edges of elements
- Shadows add depth and style



2 Core Properties

border
Defines edge of element

```
border: 2px solid #333;
border-radius: 8px;
```

Rounded Border

box-shadow
Adds shadow to elements

```
box-shadow: 0 4px 10px
rgba(0,0,0,0.2);
```

Elevated Card

text-shadow
Adds shadow to text

```
text-shadow: 2px 2px 4px
rgba(0,0,0,0.3);
```

Shadow Text

3 Summary

border → edges
Defines the edge and shape of elements.

box-shadow → depth
Creates depth and makes elements stand out.

text-shadow → styled text
Adds shadow to text for better readability and style.

Follow 👉 codingwithparvez

30 Days CSS Series

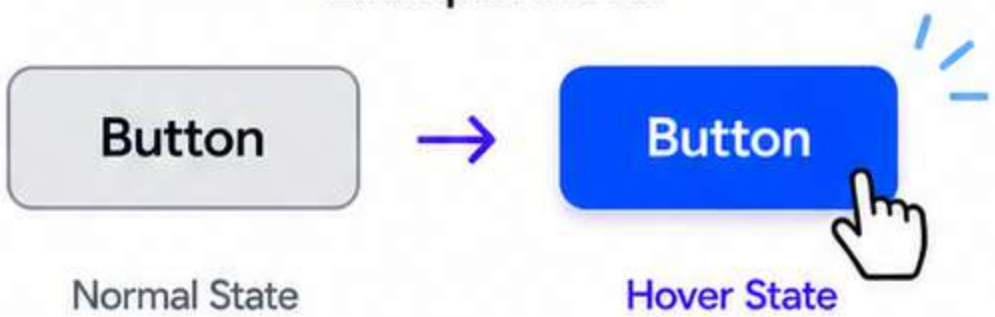
Day 24: Pseudo-Classes

:hover, :focus, :nth-child

1 What are Pseudo-Classes?

- Target elements in specific states
- Add interactivity without JavaScript

Example: :hover



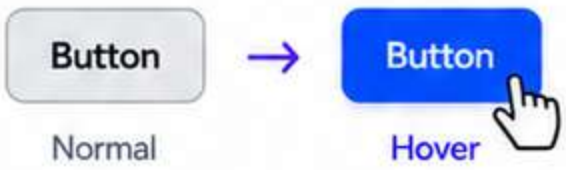
2 Common Pseudo-Classes

:hover

Triggered when mouse is over element

```
button:hover {
  background: blue;
}
```

Example



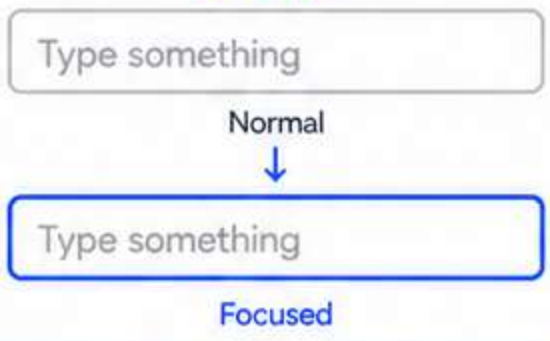
Changes style when the mouse is over the element.

:focus

Triggered when element is focused

```
input:focus {
  border: 2px solid blue;
}
```

Example



Applies style when the element receives focus.

:nth-child()

Targets specific child elements

```
li:nth-child(2) {
  color: red;
}
```

Example

- Item 1
- Item 2
- Item 3
- Item 4

Styles the second list item specifically.

3 Summary

hover
→ mouse interaction
Changes style on hover.

focus
→ active input
Applies style on focus.

nth-child
→ specific elements
Targets elements by position.

Follow 👉 codingwithparvez

30 Days CSS Series

Day 25: Pseudo-Elements

✦ ::before & ::after ✦

1 What are Pseudo-Elements?

- Style specific parts of elements
- Add content using CSS

Example:

Decorative content added before and after the element.

2 Common Pseudo-Elements

::before

Adds content before element

```
h1::before {
  content: " 🔥 ";
}
```

Example

🔥 Welcome to CSS

CSS

::after

Adds content after element

```
h1::after {
  content: " ✔";
}
```

Example

Welcome to CSS ✔

Note: Pseudo-elements require the `content` property to display anything.

3 Summary

::before
Adds content before the element.

::after
Adds content after the element.

Used for
Styling, decorations & UI effects.

Follow 👉 codingwithparvez

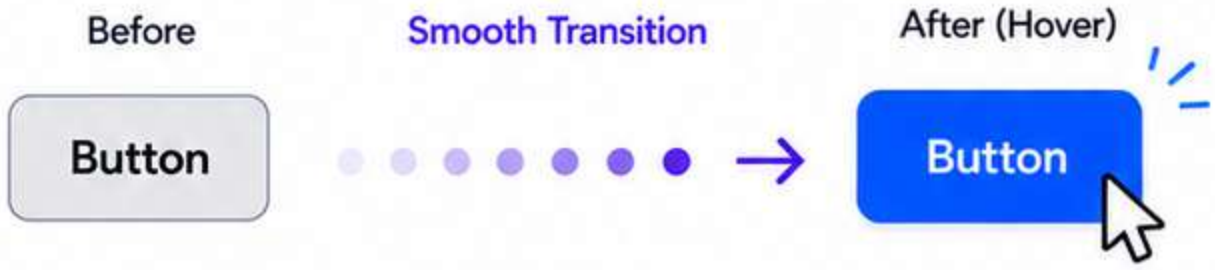
30 Days CSS Series

Day 26: CSS Transitions

✦ ease, duration, delay ✦

1 What are Transitions?

- Smoothly animate property changes
- Triggered by hover or interaction



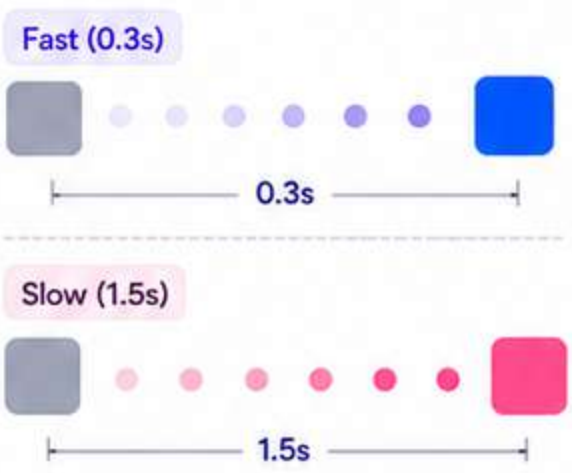
2 Core Properties

transition-duration

Controls speed of animation

```
transition-duration: 0.3s;
```

Example: Speed Comparison



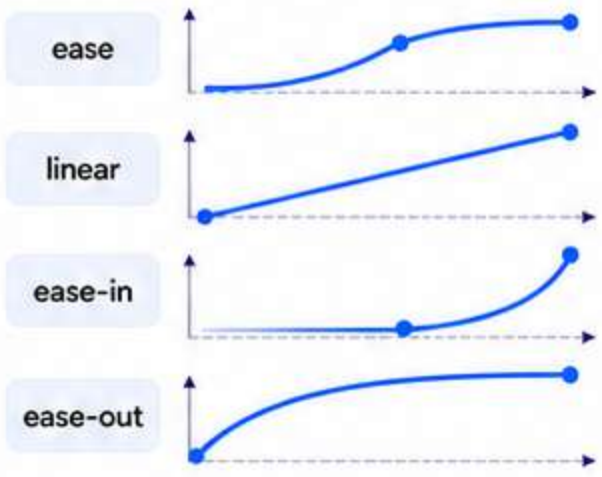
Lower time = faster
Higher time = slower

transition-timing-function (ease)

Controls animation style

```
transition-timing-function: ease;
```

Common Timing Functions



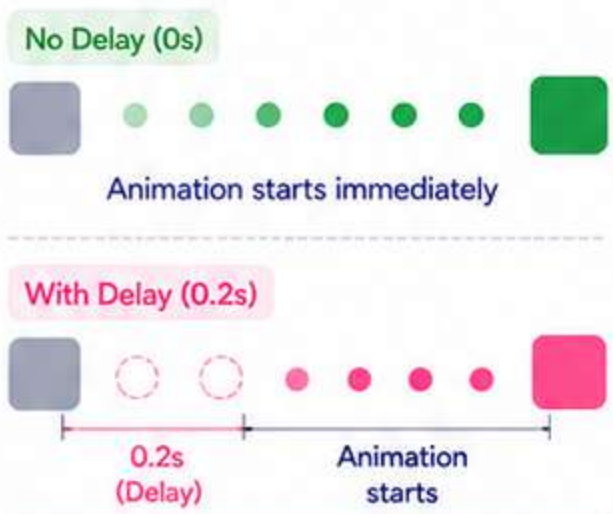
Different styles create different feels.

transition-delay

Delay before animation starts

```
transition-delay: 0.2s;
```

Example: Delay Effect



Adds a pause before animation begins.

Shorthand (All-in-One)

Set all transition properties in one line.

```
transition: all 0.3s ease 0.2s;
```

Property · Duration · Timing Function · Delay

3 Summary

duration → speed
Controls how fast the animation happens.

ease → style
Defines the motion and timing style.

delay → wait time
Sets a pause before the animation starts.

Follow 👉 codingwithparvez

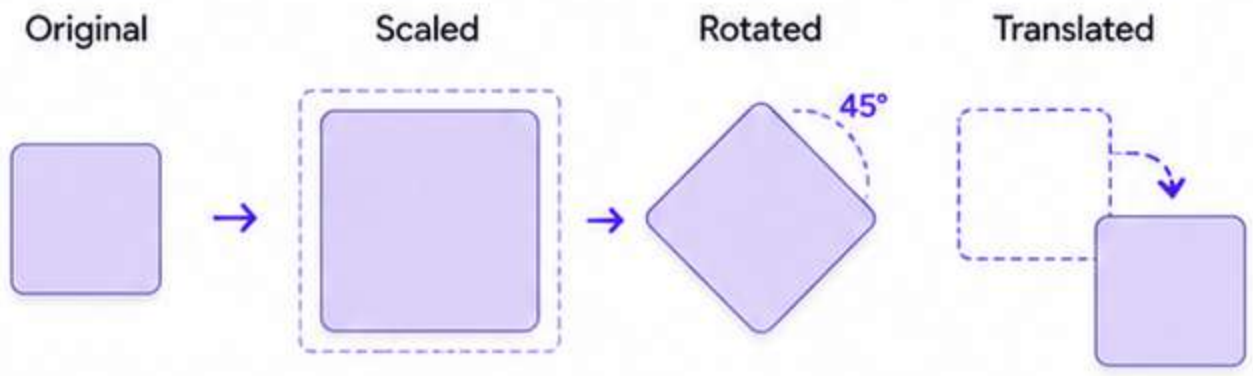
30 Days CSS Series

Day 27: CSS Transforms

✦ scale, rotate, translate ✦

1 What are Transforms?

- Change element shape, size, or position
- Works without affecting layout flow

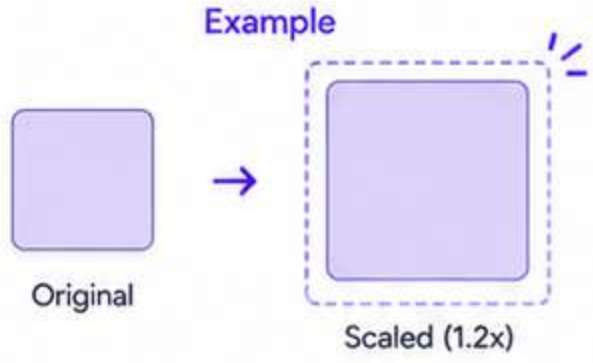


2 Transform Types

scale()
Resize element

```
transform: scale(1.2);
```

Example

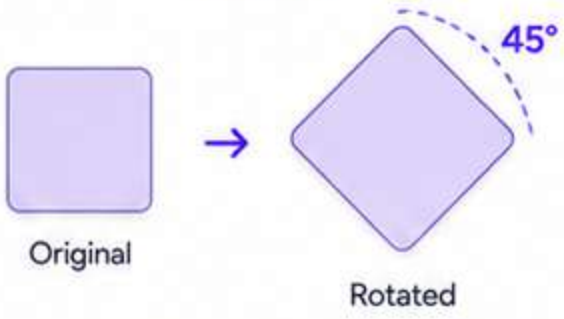


Value > 1 = bigger
Value < 1 = smaller

rotate()
Rotate element

```
transform: rotate(45deg);
```

Example

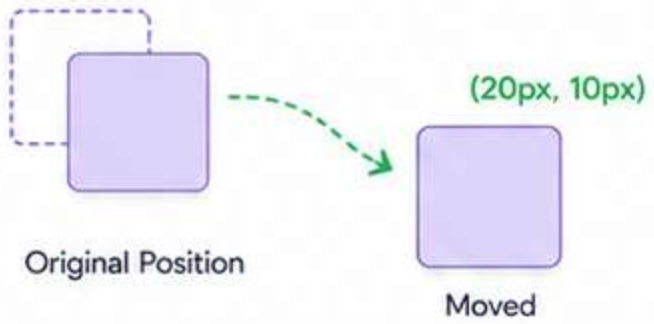


Positive = clockwise
Negative = counter-clockwise

translate()
Move element

```
transform: translate(20px, 10px);
```

Example



translate(X, Y)
+X right, -X left +Y down, -Y up

Combine Transforms
Use multiple transforms together.

```
transform: scale(1.1) rotate(10deg);
```

Resize Rotate

3 Summary

scale() → resize
Makes element bigger or smaller.

rotate() → turn
Rotates element by a given angle.

translate() → move
Moves element in X and/or Y direction.

Follow 👉 codingwithparvez

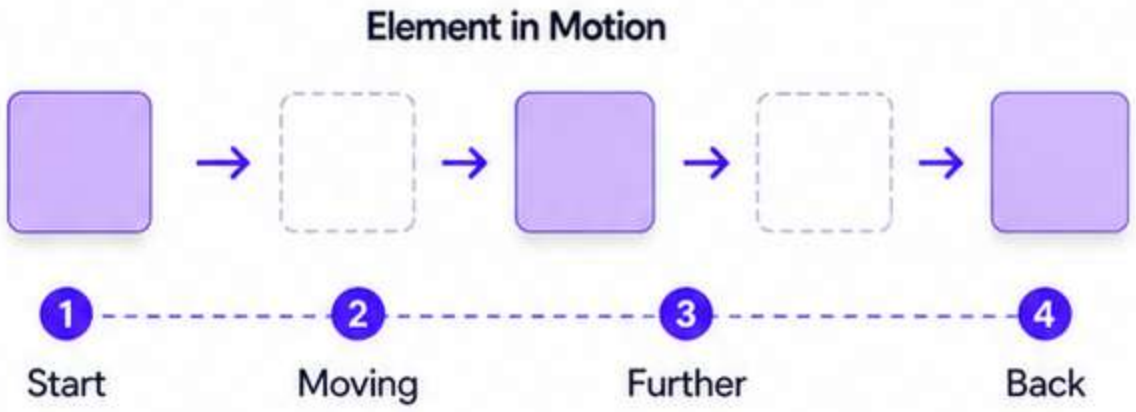
30 Days CSS Series

Day 28: CSS Animations

✦ @keyframes basics ✦

1 What are CSS Animations?

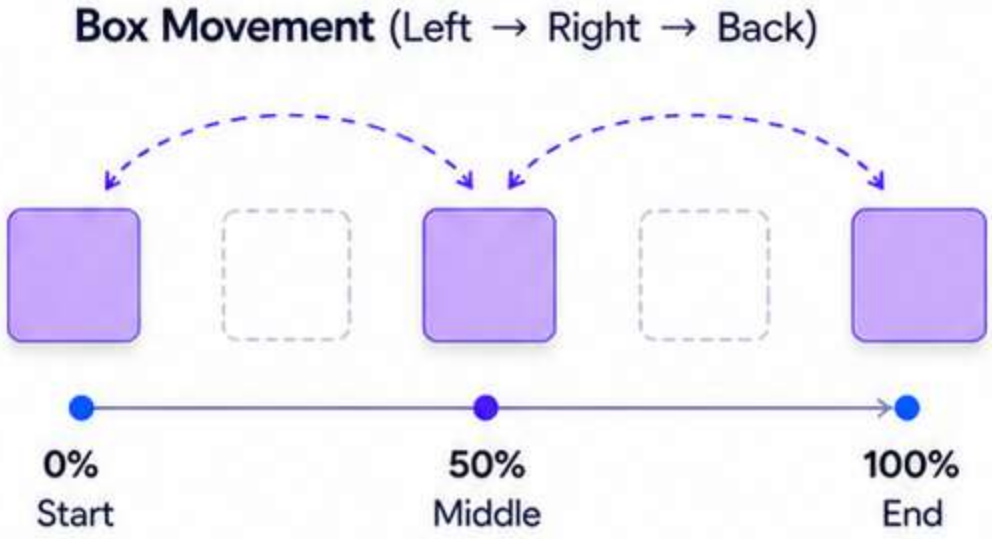
- Create smooth motion effects
- Control element changes over time



2 Keyframes Animation

```css
@keyframes move {
  0%   { transform: translateX(0); }
  50%  { transform: translateX(100px); }
  100% { transform: translateX(0); }
}

.box {
  animation: move 2s ease infinite;
}
```



Animation Properties

- duration 2s
- timing ease
- iteration infinite

Animations make your UI come alive! ✨

How it works

- 0% → Box at original position (left)
- 50% → Box moves 100px to the right
- 100% → Box returns to original position (left)
- Repeats forever (infinite)

3 Summary

@keyframes
Define the animation steps and styles at different points in time.

animation
Apply the animation to an element with duration, timing, and iteration.

30 Days CSS Series

Day 29: Media Queries & Responsive Design

✦ Make your website look good on all devices ✦

1 What is Responsive Design?

- Adapts layout to different screen sizes
- Mobile, tablet, desktop friendly

2 Media Queries

Apply styles based on screen size

```css
@media (max-width: 768px) {
  .container {
    flex-direction: column;
  }
}
```

Layout Change Example

Common Breakpoints

Mobile	Tablet	Desktop
≤ 768px	≤ 1024px	> 1024px

3 Best Practices

Use flexible units
Use %, rem, vw instead of fixed px when possible.

Mobile-first approach
Design for smaller screens first, then enhance for larger ones.

Test on multiple devices
Always test your website on different devices and screen sizes.

30 Days CSS Series

Day 30: CSS Variables

✦ Custom properties & best practices ✦

1 What are CSS Variables?

- Reusable values in CSS
- Defined using `--name`
- Accessed using `var()`

One value, used everywhere

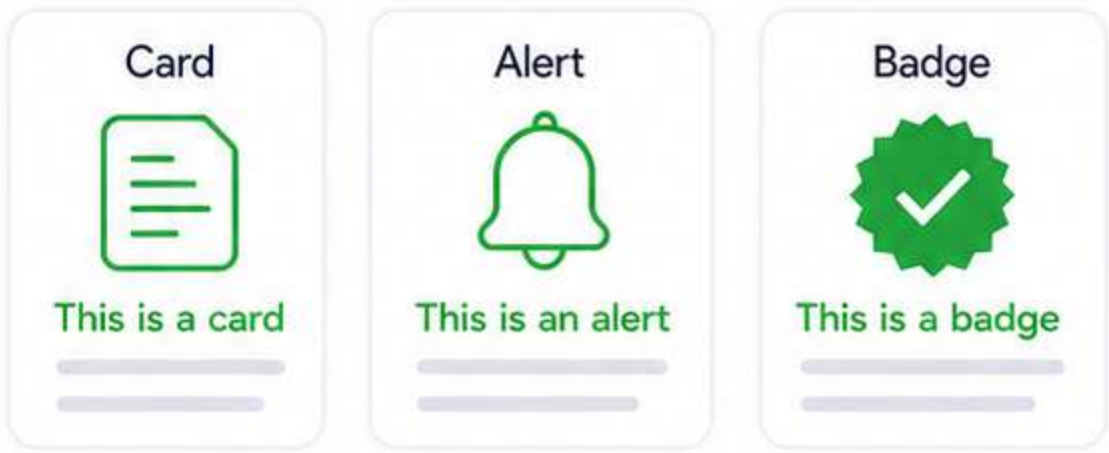
```
--primary-color: #4CAF50;
```

2 How to Use Variables

```css
:root {
  --primary-color: #4CAF50;
  --padding: 10px;
}
.box {
  color: var(--primary-color);
  padding: var(--padding);
}
```

`:root` = global scope

Same variables, consistent results



Change the variable once, update everywhere!

```
--primary-color: #4CAF50;
--padding: 10px;
```

3 Best Practices

Use meaningful variable names
Choose names that describe the purpose clearly.

Keep variables in :root
Store global variables in `:root` for easy access and reuse.

Use for colors, spacing, fonts
Perfect for theming, spacing, typography, and more.

Improves maintainability
Easier updates, better consistency, and cleaner code.

CSS Variables = Cleaner Code, Faster Updates, Better Websites ✨

Follow 👉 codingwithparvez

30 Days CSS Completed

From Beginner to Confident CSS Developer

You learned:

Day 1 → Day 30 ✓

Fundamentals · Layouts · Flexbox · Grid · Animations · Responsive

Skills Covered

Basics
- Selectors
- Colors & Units
- Syntax & Comments

Box Model
- Margin & Padding
- Border & Outline
- Width & Height

Layout
- Display & Position
- Float & Clear
- Z-index & Stack

Flexbox
- Alignment
- Justify & Align Items
- Order & Direction

Grid
- Rows & Columns
- Grid Areas
- Gap & Alignment

Styling
- Backgrounds
- Borders & Radius
- Shadows & Opacity

Effects
- Transitions
- Transforms (Scale, Rotate)
- Animations (@keyframes)

Advanced
- Responsive Design
- Media Queries
- CSS Variables

Before

- Confused beginner
- Struggled with layout
- Inconsistent styles

After 30 Days

- CSS Confident 🚀
- Build clean layouts
- Create responsive sites

Now build real projects!

Keep practicing daily ✨

Follow 👉 codingwithparvez for more dev content

30 Days CSS Completed

From Beginner to Confident CSS Developer

You learned:

Day 1 → Day 30 ✓

Fundamentals · Layouts · Flexbox · Grid · Animations · Responsive

Skills Covered

Basics
- Selectors
- Colors & Units
- Syntax & Comments

Box Model
- Margin & Padding
- Border & Outline
- Width & Height

Layout
- Display & Position
- Float & Clear
- Z-index & Stack

Flexbox
- Alignment
- Justify & Align Items
- Order & Direction

Grid
- Rows & Columns
- Grid Areas
- Gap & Alignment

Styling
- Backgrounds
- Borders & Radius
- Shadows & Opacity

Effects
- Transitions
- Transforms (Scale, Rotate)
- Animations (@keyframes)

Advanced
- Responsive Design
- Media Queries
- CSS Variables

Before

- Confused beginner
- Struggled with layout
- Inconsistent styles

After 30 Days

- CSS Confident 🚀
- Build clean layouts
- Create responsive sites

Now build real projects!

Keep practicing daily ✨

Follow 👉 codingwithparvez for more dev content

12 CSS Project Ideas

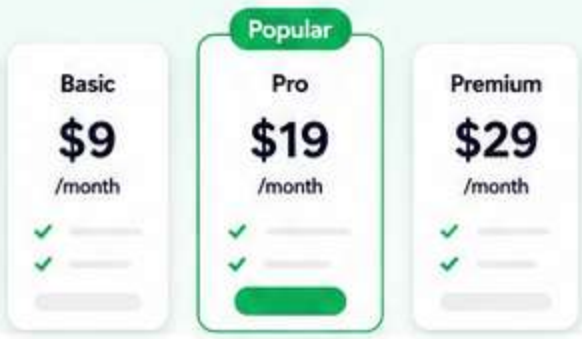
Practice everything learned in the 30 Days CSS Series

1 Responsive Navbar

Practice Flexbox & positioning

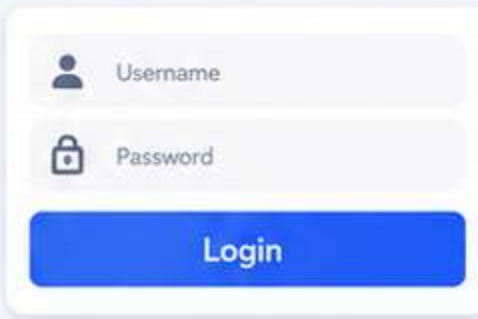
2 Pricing Card UI

Shadows, spacing, hover effects



3 Login Form

Inputs, focus states, alignment

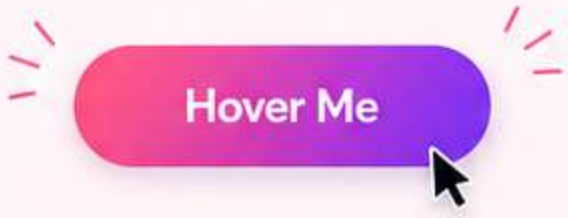


4 CSS Grid Gallery

Grid layout practice

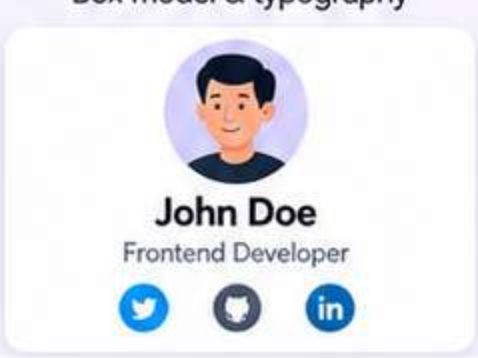
5 Animated Button

Transitions & transforms



6 Profile Card

Box model & typography



7 Sticky Header

Fixed & sticky positioning

8 Loading Spinner

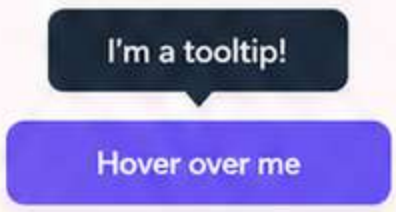
Keyframe animations

9 Responsive Dashboard

Flexbox + Grid combined

10 Tooltip Component

Pseudo-elements & positioning



11 Dark/Light Theme Toggle

CSS variables

12 Landing Page Hero Section

Backgrounds, gradients, layout

Build projects → Improve faster

Follow 👉 codingwithparvez