Parul® University

Web Designing (03010501PC01)

Unit 3 : CSS Basics & Styling

Parul®
University

What is CSS?

- CSS = Cascading Style Sheets
- Used to style HTML elements (colors, fonts, layouts)
- Separates **content (HTML)** from **design (CSS)**

Example:

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

Parul®
University

Why Use CSS?

- Controls **look & feel** of websites
- Reuse styles across multiple pages
- Makes websites **responsive** and **consistent**

➢ Without CSS = plain text pages

➢ With CSS = modern, colorful, user-friendly websites

Parul®
University

Ways to Apply CSS

1. Inline CSS (inside tag)

```
<p style="color: red;">This is red text</p>
```

2. Internal CSS (inside <style> in HTML)

```
<style>
        p { color: green; }
</style>
```

3. External CSS (linked file)

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

CSS Syntax

General rule:

```
selector {
    property: value;
}
```

Example:

```
h1 {
  color: purple;
  font-size: 24px;
}
```

CSS Comments

- Used to add notes inside CSS
- Not shown on webpage

```
/* This styles all headings */

h1 {
   color: navy;
}
```

Example: Styling a Paragraph

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red;
  font-size: 18px;
  font-family: Arial, sans-serif;
}
</style>
</head>
<body>
  <p>CSS makes text look better!</p>
</body>
</html>
```

Demo Output

. CSS makes text look better!

Parul®
University

Summary

- **Element selector** → targets tags
- **Class selector (.class)** → reusable styles
- **ID selector (#id)** → unique element
- **Hover & Active** → interactive styles

Parul®
University

What is a Selector?

.

- Selector = tells CSS **which HTML element(s)** to style.
- Properties = tell **how to style** it.

Element Selector

Selects all elements of a type.

```
p {
  color: blue;
}
```

Output: All <p> tags become blue.

Class Selector

Used with **dot (.)** symbol.

```
.intro {
   font-size: 20px;
   color: green;
}
```

ID Selector

- Used with **hash (#)** symbol.
- IDs are **unique per page**.

```
css:
#main {
   background-color: yellow;
}

html:
<div id="main">Main Section</div>
```

Multiple Selectors

```
css:
.
h1, h2, p {
   font-family: Arial;
}
```

- Styles applied to all h1,h2 and p.

Pseudo-classes – Hover

- Apply styles when mouse hovers.

css:

```
a:hover {
  color: red;
  text-decoration: underline;
}
```

Pseudo-classes – Active

- Apply styles when element is **clicked/active**.

css:

```
a:active {
  color: red;
}
```

Parul®
University

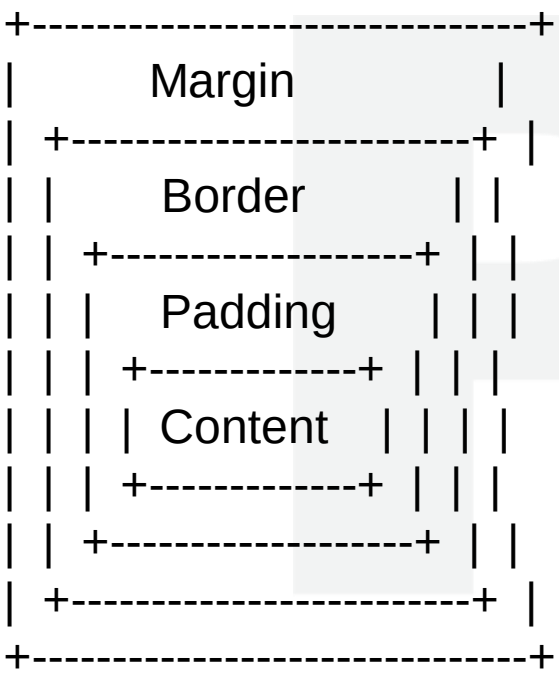
What is the Box Model?

- Every HTML element is a **box** consisting of:

1. **Content** → Text, image, or element inside
2. **Padding** → Space between content and border
3. **Border** → Line surrounding the box
4. **Margin** → Space outside the border

Visual Diagram

- Box Model



Box Model Example

- Apply styles when mouse hovers.

```
css:

div {
   width: 200px;
   height: 100px;
   padding: 20px;
   border: 5px solid black;
   margin: 15px;
}
```

- Creates a box with **content, padding, border, margin**.

Height and Width

- width → width of content area
- Height → height of content area

```
css:

P {
   width: 300px;
   height: 150px;
}
```

Display Property

- Block → takes full width (div, p)
- Inline → takes only content width (span, p)
- Inline-block → inline but allows width/height
- None → hides element completely

Overflow Property

- Apply styles when mouse hovers.
- Visibility: visible; → default
- Visibility: hidden; → hides element but keeps space

```
css:

h1 {
        visibility: hidden;
}
```

Overflow Property

- Controls content that is too large:
- visible (default)
- Hidden (cut off)
- Scroll (adds scrollbar)
- Auto (scroll only if needed)

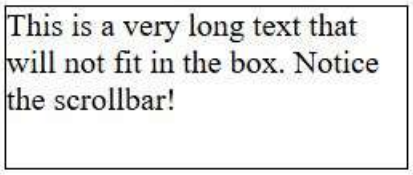
css:

```
div {
        width: 150px;
        height: 100px;
        overflow: scroll;
}
```

Example Demo

```
<div style="width:200px; height:80px; border:1px solid black; overflow:auto;">
  This is a very long text that will not fit in the box.
  Notice the scrollbar!
</div>
```

Output:



Parul®
University

Summary

- Box model = Content + Padding + Border + Margin
- Use width & height for sizing
- Use display, visibility, overflow for layout control.

CSS Colors

- Ways to define colors:

1. Name → color: red;
2. Hex Code → color: #ff0000;
3. RGB → color: rgb(255,0,0);
4. RGBA (with transparency) → color: rgba(255,0,0,0.5);

Example of Colors

- Heading → Blue, Paragraph → Green, Span → Orange

css:

```
h1 { color: blue; }
p { color: #008000; }
span { color: rgb(255, 165, 0); }
```

CSS Fonts – Font Family

```
css:

p {
   font-family: Arial, Helvetica, sans-serif;
}
```

- Always give **fallback fonts**.
- Categories: serif, sans-serif, monospace, cursive, fantasy.

CSS Fonts – Font Size

```
css:

h1 { font-size: 32px; }
p { font-size: 16px; }
```

- Units:
- px (pixels)
- em (relative to parent)
- rem (relative to root)
- % (percentage)

CSS Fonts – Font Style & Weight

```
css:

p {
   font-style: italic;
   font-weight: bold;
}
```

- font-style → normal, italic, oblique
- Font-weight → normal, bold, 100-900

CSS Backgrounds – Colors

```
css:

body {
   background-color: lightblue;
}
```

Output:
Sets background of page.

CSS Backgrounds – Images

```
css:

div {
  background-image: url('bg.jpg');
  background-repeat: no-repeat;
  background-size: cover;
}
```

- repeat / no-repeat / repeat-x / repeat-y
- cover → fill screen
- contain → fit inside

CSS Backgrounds – Position & Attachment

```
css:

body {
  background-image: url('pattern.png');
  background-position: center;
  background-attachment: fixed;
}
```

- Position → left, right, top, bottom, center
- attachment: fixed → background does not scroll

Example Demo

css:

```
<body style="background: url('stars.png') no-repeat center fixed;
            background-size: cover;
            color: black;">
  <h1>Beautiful Background</h1>
  <p>This text appears on top of a starry background.</p>
</body>
```

- Position → left, right, top, bottom, center
- attachment: fixed → background does not scroll

Example Demo : OUTPUT

Parul®
University

Summary

- Colors: names, hex, RGB, RGBA
- Fonts: family, size, style, weight
- Background: color, image, position, repeat, size

Parul® University

What is Flexbox?

- Flexible Box Layout (Flexbox) → arranges items in **rows or columns**.
- Makes layouts **responsive & easy to align**.
- Container = **flex parent** → holds items.
- Items = **flex children** → arranged inside.

Declaring a Flex Container

```
css:

.container {
   display: flex;
}
```

- All direct children of .container become flex items.

Flex Direction

```
css:

.container {
  display: flex;
  flex-direction: row;   /* default */
}
```

- row → items left to right
- row-reverse → right to left
- column → top to bottom
- column-reverse → bottom to top

Justify Content

- Controls **horizontal alignment**.

```
css:

.container {
  display: flex;
  justify-content: space-between;
}
```

- Options:
  - flex-start (default,left)
  - flex-end (right)
  - center
  - space-between (equal gaps between)
  - space-around (equal gaps around items)
  - space-evenly (perfectly even spacing)

Align Items

- Controls **vertical alignment**.

```
css:

.container {
  display: flex;
  align-items: center;
}
```

- Options:

- Stretch (default)
- Flex-start (top)
- Flex-end (bottom)
- Center (middle)
- Baseline (aligns text baselines)

Flex Wrap

- By default, items **shrink into one line**.

```
css:

.container {
   display: flex;
   flex-wrap: wrap;
}
```

- Options:

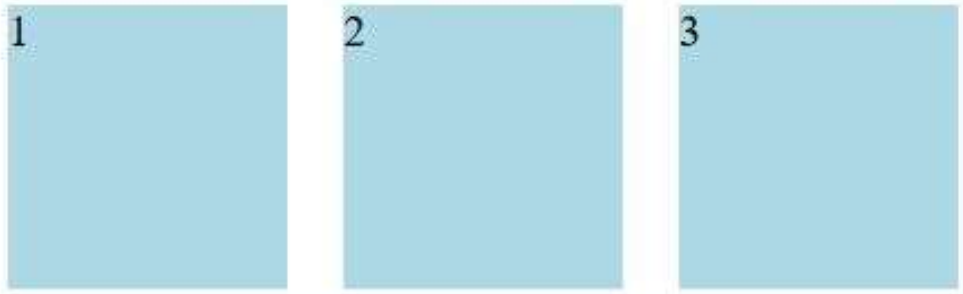
- nowrap (default, all items in one line)
- wrap (items move to new line if needed)
- wrap-reverse (wrap but in reverse order)

Example Flexbox Layout

```
html:
<div class="container">
  <div class="box">1</div>
  <div class="box">2</div>
  <div class="box">3</div>
</div>

css:
.container {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-wrap: wrap;
  gap: 20px;
}
.box {
  width: 100px;
  height: 100px;
  background: lightblue;
}
```

Example Flexbox Layout: OUTPUT



OUTPUT:

- All boxes centered both horizontally and vertically.
- If screen shrinks → boxes wrap into next line.

Parul®
University

Summary

- **display: flex** → enables flexbox
- **flex-direction** → row/column
- **justify-content** → horizontal alignment
- **align-items** → vertical alignment
- **flex-wrap** → controls wrapping

Parul®
University

What is CSS Grid?

- CSS Grid = **powerful 2D layout system**.
- Works with **rows and columns**.
- Useful for page layouts, dashboards, galleries.
- More control than Flexbox for grids.

Declaring a Grid Container

css:

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

- All direct children become **grid items**.

Defining Columns

```
css:

.container {
  display: grid;
  grid-template-columns: 100px 200px auto;
}
```

- More control than Flexbox for grids.

- 1st = 100px
- 2nd = 200px
- 3rd = auto (takes remaining space)

Defining Rows

```
css:

.container {
  display: grid;
  grid-template-rows: 100px 150px auto;
}
```

- More control than Flexbox for grids.

- Row 1 = 100px
- Row 2 = 150px
- Row 3 = auto

Using Fraction Units (fr)

css:

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

- Total space divided into fractions.
- 1st = 1 part, 2nd = 2 parts, 3rd = 1 part.

Adding Gaps

```
css:

.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 20px;
}
```

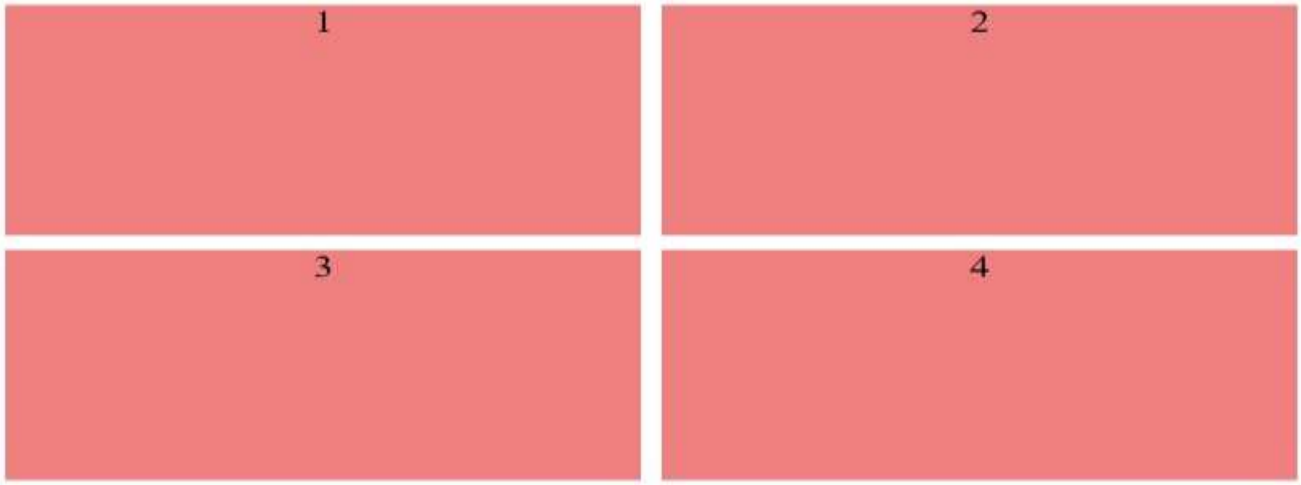
- gap → space between rows & columns.
- Shortcuts:
  - row-gap
  - column-gap

Example Grid Layout

```
html:
<div class="container">
  <div class="box">1</div>
  <div class="box">2</div>
  <div class="box">3</div>
  <div class="box">4</div>
</div>

css:
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 150px 150px;
  gap: 10px;
}
.box {
  background: lightcoral;
  text-align: center;
  font-size: 20px;
}
```

Example Grid Layout: OUTPUT



- Grid has **2 columns & 2 rows**.
- Boxes 1–4 automatically placed.
- Gaps = 10px between them.

Parul®
University

Summary

- **display: grid** → enables grid system
- **grid-template-columns** → sets columns
- **grid-template-rows** → sets rows
- **fr unit** → divides space proportionally
- **gap** → spacing between rows & columns

Parul®
University

What is Positioning?

- Positioning controls **where elements appear**.
- Works with properties:
  - top, bottom, left, right.

- Main types:

  - Static
  - Relative
  - Absolute
  - Fixed
  - sticky

Static Position (Default)

```
css:

.box {
   position: static;
}
```

- Default for all elements.
- Elements follow **normal page flow**.
- Cannot use top, left, etc

Relative Position

```
css:

.box {
  position: relative;
  top: 20px;
  left: 30px;
}
```

- Moves element **relative to its normal position**.
- Space is still reserved in layout.

Absolute Position

```
css:

.box {
   position: absolute;
   top: 50px;
   left: 100px;
}
```

- Element positioned **relative to nearest positioned ancestor** (not body).
- Removed from normal flow.

Fixed Position

```
css:

.box {
   position: fixed;
   bottom: 0;
   right: 0;
}
```

- Element stays **fixed on screen**, even when scrolling.
- Common for navigation bars, chat buttons.

Sticky Position

```
css:

.box {
   position: position;
Top: 0;
}
```

- Behaves like **relative** until a scroll point.
- Then “sticks” like **fixed**.
- Good for sticky headers.

z-index

```
css:

.box1 {
  position: absolute;
  z-index: 1;
}
.box2 {
  position: absolute;
  z-index: 2;
}
```

- Controls **stacking order** of elements.
- Higher z-index = appears in front.
- Works only on positioned elements.

Example Demo

```
html:
<div class="box red">Box 1</div>
<div class="box blue">Box 2</div>

css:
.red {
  position: absolute;
  top: 50px;
  left: 50px;
  z-index: 1;
  background: red;
}
.blue {
  position: absolute;
  top: 70px;
  left: 70px;
  z-index: 2;
  background: blue;
}
```

Example Demo : OUTPUT

- Blue box appears above red box.

Summary

- **static** → default flow
- **relative** → move from normal position
- **absolute** → relative to parent
- **fixed** → stuck on viewport
- **sticky** → toggles between relative & fixed
- **z-index** → controls overlapping

Parul®
University

Project Overview

- We will create a **responsive product cards layout** that includes:

- Header (fixed position)
- Responsive card grid (Flexbox + Grid)
- Hover effects
- Sticky section title

HTML Structure

```
html:

<body>
  <header>My Store</header>
  <h2 class="section-title">Featured Products</h2>

  <div class="card-container">
    <div class="card">
      <img src="img1.jpg" alt="Product">
      <h3>Product 1</h3>
      <p>$20</p>
      <button>Buy Now</button>
    </div>
    <div class="card"> ... </div>
    <div class="card"> ... </div>
  </div>
</body>
```

Styling Header (Fixed Position)

```
css:

header {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  background: #333;
  color: white;
  padding: 15px;
  text-align: center;
  z-index: 10;
}
```

- Always visible on scroll.
- z-index: 10 keeps it on top

Sticky Section Title

css:

```
.section-title {
  position: sticky;
  top: 70px;
  background: #f8f8f8;
  padding: 10px;
  font-size: 24px;
}
```

- Stays visible while scrolling product list.

Responsive Card Container (Flexbox)

css:

```
.card-container {
  display: flex;
  flex-wrap: wrap;
  gap: 20px;
  margin-top: 100px;
  justify-content: center;
}
```

- Flexible wrapping layout.
- Gap between cards.
- Responsive behavior.

Card Design (Box Model + Hover)

```
css:

.card {
  background: white;
  padding: 15px;
  border: 1px solid #ddd;
  border-radius: 10px;
  width: 250px;
  box-shadow: 0 4px 6px rgba(0,0,0,0.1);
  transition: transform 0.2s;
}
.card:hover {
  transform: scale(1.05);
}
```

- Uses **box model properties**.
- **Hover effect** zooms in card.

Card Grid with CSS Grid (Alternative)

css:

```
.card-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  gap: 20px;
  margin-top: 100px;
}
```

- Automatically adjusts to screen size.
- Works better for **responsive layouts**.

Responsive Adjustments (Media Query)

css:

```
@media (max-width: 600px) {
  .card {
    width: 100%;
  }
}
```

- On small screens, each card takes **full width**.

Big screen view

My Store

Featured Products

Product

Product 1

$20

Buy Now

Product

Product 2

$20

Buy Now

Product

Product 3

$20

Buy Now

Mobile screen view

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in