



# CSS Handbook

---



## Introduction

We will start from the basics (how to add CSS to HTML) and move to important concepts like text styles, the box model, margins, padding, and borders.

Each topic includes:

- Notes (important points)
- Syntax (how to write it)
- Examples with full runnable code
- Simple explanations
- Practice exercises

At the end, you will also get mini-projects, a cheat sheet, glossary, and solutions.

By the end of this handbook, you will be able to style simple web pages with confidence.

---



## Table of Contents

### 1. Ways to Add CSS to HTML

- Inline CSS
- Internal CSS
- External CSS

### 2. Text and Font Properties with Units

- color, font-family, font-size, font-weight, font-style, text-align, text-decoration, line-height, letter-spacing, word-spacing, text-transform

### 3. CSS Box Model

- content, padding, border, margin, width, height, box-sizing

#### 4. Margin and Padding

- Single value, 2-value, 3-value, 4-value

#### 5. Borders and Border Radius.

- border-width, border-style, border-color, border-radius
- Solid, double, dashed, dotted, groove, ridge, inset, outset, none, hidden

#### 6. Mini Projects (per border style)

7. Display : flex ; { layout designing }

#### 8. Cheat Sheet

#### 9. Glossary

#### 10. Answers to Exercises

---

## Section A: Ways to Add CSS to HTML

---

### 1. Inline CSS

#### Notes

- Inline CSS means writing CSS **inside the HTML tag** using the `style` attribute.
- It is applied only to that single element.
- Good for **quick testing**, but not for big projects.

#### Syntax

```
<tag style="property:value;">Content</tag>
```

#### Example 1: Change heading color

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Inline CSS Example</title>
</head>
<body>
  <h1 style="color: blue;">Hello World</h1>
</body>
</html>
```

### Explanation:

- `<h1>` → heading element
- `style="color: blue;"` → inline CSS, makes text blue

## ✅ Example 2: Multiple styles on one element

```
<!DOCTYPE html>
<html>
<head>
  <title>Inline CSS Example</title>
</head>
<body>
  <p style="color: green; font-size: 20px; text-align: center;">
    This is a green centered paragraph.
  </p>
</body>
</html>
```

### Explanation:

- `color: green;` → text color
- `font-size: 20px;` → text size
- `text-align: center;` → center aligned



## Practice Questions

1. Create an inline style that makes a heading **red** and **underlined**.

2. Write inline CSS to make a paragraph **blue, italic**, and **font size 18px**.
  3. Use inline CSS to give a `<div>` a **yellow background**.
- 

## 2. Internal CSS

### Notes

- Internal CSS is written inside the `<style>` tag in the `<head>` section.
- It applies to the whole HTML file.
- Better than inline for **styling multiple elements**.

### Syntax

```
<head>
  <style>
    selector {
      property: value;
    }
  </style>
</head>
```

### Example 1: Style all paragraphs

```
<!DOCTYPE html>
<html>
<head>
  <title>Internal CSS Example</title>
  <style>
    p {
      color: purple;
      font-size: 18px;
    }
  </style>
</head>
```

```
<body>
  <p>This is paragraph one.</p>
  <p>This is paragraph two.</p>
</body>
</html>
```

### Explanation:

- `<style>` → internal CSS area
- `p { color: purple; font-size: 18px; }` → applies to all `<p>`

## ✓ Example 2: Different styles for different tags

```
<!DOCTYPE html>
<html>
<head>
  <title>Internal CSS Example</title>
  <style>
    h1 {
      color: navy;
      text-align: center;
    }
    p {
      color: gray;
    }
  </style>
</head>
<body>
  <h1>Welcome</h1>
  <p>This is a styled paragraph.</p>
</body>
</html>
```

### Explanation:

- `h1 { ... }` → affects all `<h1>`

- `p { ... }` → affects all `<p>`

## Practice Questions

1. Write internal CSS to make all `<h2>` headings green.
2. Add internal CSS that gives all `<p>` text a **blue color** and **center alignment**.
3. Use internal CSS to make all `<div>` elements have a **lightgray background**.

## 3. External CSS

### Notes

- External CSS means writing CSS in a **separate file** (example: `style.css` ).
- The CSS file is linked to HTML using `<link>` .
- Best for **big projects** and **reusability**.

### Syntax (HTML file)

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

### Syntax (style.css file)

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

## Example 1: Heading styles with external CSS

index.html

```
<!DOCTYPE html>
<html>
  <head>
```

```
<title>External CSS Example</title>
<link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Hello CSS</h1>
  <p>This is styled using external CSS.</p>
</body>
</html>
```

### style.css

```
h1 {
  color: red;
}
p {
  color: blue;
}
```

### Explanation:

- `<link rel="stylesheet" href="style.css">` → links CSS file
- `h1 { color: red; }` → makes heading red
- `p { color: blue; }` → makes paragraph blue

## ✓ Example 2: Styling a simple page

### index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>External CSS Example</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Welcome to My Page</h1>
```

```
<p>This paragraph has styles from an external file.</p>
<div>CSS makes web pages beautiful!</div>
</body>
</html>
```

## style.css

```
h1 {
  color: green;
  text-align: center;
}
p {
  font-size: 18px;
}
div {
  background-color: lightyellow;
  padding: 10px;
}
```

## Explanation:

- `h1 { ... }` → styles the heading
- `p { ... }` → increases font size
- `div { ... }` → adds background and padding

## Practice Questions

1. Create a CSS file that makes all `<h1>` headings blue and all `<p>` text gray.
2. Use external CSS to set body background color to **lightblue**.
3. Make all `<div>` boxes have a border with external CSS.

✅ That completes **Section A (Ways to Add CSS to HTML)**.

# Section B: Text and Font Properties

---

## 1. Color

### Notes

- `color` sets the text color.
- You can use **color names** (red, blue), **hex codes** (#ff0000), or **rgb()** values.

### Syntax

```
selector {  
  color: value;  
}
```

### Example 1: Using a color name

```
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    p {  
      color: red;  
    }  
  </style>  
</head>  
<body>  
  <p>This text is red.</p>  
</body>  
</html>
```

### Explanation:

- `p { color: red; }` → makes all paragraphs red.

## ✓ Example 2: Using hex code

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      color: #008000; /* green */
    }
  </style>
</head>
<body>
  <h1>Hello World</h1>
</body>
</html>
```

### Explanation:

- `#008000` → green color in hex.



### Practice

1. Make a paragraph text blue.
2. Use hex code to make a heading orange.
3. Try `rgb(0,0,255)` to make text blue.

## 2. Font-Family



### Notes

- `font-family` sets the font type.
- If the first font is missing, browser uses the next one.



### Syntax

```
selector {  
  font-family: "Font Name", fallback;  
}
```

## ✓ Example 1: Basic font

```
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    p {  
      font-family: Arial, sans-serif;  
    }  
  </style>  
</head>  
<body>  
  <p>This text uses Arial.</p>  
</body>  
</html>
```

## ✓ Example 2: Multiple fonts

```
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    h1 {  
      font-family: "Times New Roman", Georgia, serif;  
    }  
  </style>  
</head>  
<body>  
  <h1>Heading with Times New Roman</h1>
```

```
</body>
</html>
```

## Practice

1. Apply Arial to all `<p>` tags.
2. Use `"Courier New"` for code text.
3. Apply `Verdana` with fallback `sans-serif`.

## 3. Font-Size

### Notes

- `font-size` controls text size.
- Units: px, em, rem, %.

### Syntax

```
selector {
  font-size: 20px;
}
```

### Example 1: Pixels

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      font-size: 20px;
    }
  </style>
</head>
<body>
```

```
<p>This paragraph is 20px.</p>
</body>
</html>
```

## ✓ Example 2: Percentage

```
<!DOCTYPE html>
<html>
<head>
<style>
  p {
    font-size: 150%;
  }
</style>
</head>
<body>
  <p>This text is bigger using %.</p>
</body>
</html>
```



## Practice

1. Set font size of heading to 30px.
2. Make a paragraph 120%.
3. Try using `2em` size for text.

## 4. Font-Weight



### Notes

- `font-weight` controls thickness.
- Values: normal, bold, bolder, lighter, or numbers (100–900).



### Syntax

```
selector {  
  font-weight: bold;  
}
```

## ✓ Example 1: Bold text

```
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    p {  
      font-weight: bold;  
    }  
  </style>  
</head>  
<body>  
  <p>This text is bold.</p>  
</body>  
</html>
```

## ✓ Example 2: Numeric values

```
<!DOCTYPE html>  
<html>  
<head>  
  <style>  
    h1 {  
      font-weight: 900;  
    }  
  </style>  
</head>  
<body>  
  <h1>Very Bold Heading</h1>
```

```
</body>
</html>
```

## Practice

1. Make a heading lighter.
2. Try font-weight: 600.
3. Apply bold to a paragraph.

## 5. Font-Style

### Notes

- `font-style` changes text to italic or normal.

### Syntax

```
selector {
  font-style: italic;
}
```

### Example 1: Italic text

```
<!DOCTYPE html>
<html>
<head>
  <style>
    p {
      font-style: italic;
    }
  </style>
</head>
<body>
  <p>This is italic text.</p>
```

```
</body>
</html>
```

## ✓ Example 2: Normal style

```
<!DOCTYPE html>
<html>
<head>
  <style>
    em {
      font-style: normal;
    }
  </style>
</head>
<body>
  <p><em>This looks normal, not italic.</em></p>
</body>
</html>
```



## Practice

1. Make a sentence italic.
2. Change italic element back to normal.
3. Try bold + italic together.

# Section C : CSS Units



## Notes

- Units define the **size** of elements (width, height, padding, margin, font-size, etc.).
- Two main types of units:

1. **Absolute Units** → fixed size (do not change).
2. **Relative Units** → flexible size (change based on parent, screen, or font).


## Absolute Units

Unit	Meaning	Example
px	Pixels (most common, fixed size)	width: 200px;
cm	Centimeters	width: 5cm;
mm	Millimeters	width: 50mm;
in	Inches	width: 2in;
pt	Points (1/72 inch)	font-size: 12pt;
pc	Picas (12 points)	font-size: 2pc;

 In practice → **px** is used the most. Others are rarely used in web design.

## Relative Units

Unit	Meaning	Example
%	Relative to parent size	width: 50%;
em	Relative to parent font size	font-size: 2em; (2× parent)
rem	Relative to root (HTML) font size	font-size: 1.5rem;
vw	Relative to viewport width (1vw = 1% of screen width)	width: 50vw;
vh	Relative to viewport height (1vh = 1% of screen height)	height: 50vh;
vmin	1% of smaller dimension (width or height)	font-size: 5vmin;
vmax	1% of larger dimension	font-size: 5vmax;

 Responsive design uses **%**, **em**, **rem**, **vw**, **vh**.

## Examples

## Example 1: Using px and %

```
<!DOCTYPE html>
<html>
<head>
<style>
.container {
  width: 600px;
  border: 2px solid black;
}
.box {
  width: 50%;
  height: 100px;
  background: lightblue;
}
</style>
</head>
<body>
  <div class="container">
    <div class="box">50% of parent</div>
  </div>
</body>
</html>
```

### Explanation:

- Parent ( `.container` ) → 600px wide.
- Child ( `.box` ) → 50% of 600px = 300px.

## Example 2: em vs rem

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```

html {
  font-size: 16px; /* root font size */
}
.parent {
  font-size: 20px;
}
.em-box {
  font-size: 2em; /* 2 × parent = 40px */
}
.rem-box {
  font-size: 2rem; /* 2 × root (16px) = 32px */
}
</style>
</head>
<body>
  <div class="parent">
    <p class="em-box">This is 2em text</p>
    <p class="rem-box">This is 2rem text</p>
  </div>
</body>
</html>

```

#### Explanation:

- **2em** → based on parent (20px × 2 = 40px).
- **2rem** → based on root (16px × 2 = 32px).

### Example 3: vw and vh

```

<!DOCTYPE html>
<html>
<head>
<style>
  .box {
    width: 50vw; /* 50% of screen width */
  }

```

```
height: 30vh; /* 30% of screen height */
background: lightgreen;
}
</style>
</head>
<body>
  <div class="box">Responsive Box</div>
</body>
</html>
```

 Explanation:

- Box resizes with screen.
- `50vw` = half of browser width.
- `30vh` = 30% of browser height.



## Practice Questions

1. Create a box with **200px width** and **100px height**.
2. Make a child div that is **60% width** of its parent.
3. Create text using **2em** inside a parent with `font-size: 15px`.
4. Make a full-screen div using **100vw** and **100vh**.
5. Try difference between **em** and **rem**.

## Section D : Background Properties in CSS



### Notes

Background properties let us **style the area behind content**.

We can add:

- **Solid color**

- **Images**
- **Gradients**
- Control how they **repeat, position, and size**.

## Common Background Properties

1. `background-color` → sets a solid color.
2. `background-image` → sets an image.
3. `background-repeat` → repeat behavior ( `repeat` , `repeat-x` , `repeat-y` , `no-repeat` ).
4. `background-position` → where the image starts ( `top` , `center` , `bottom` , `left` , `right` , or values like `50px 100px` ).
5. `background-size` → image scaling ( `auto` , `cover` , `contain` , custom size ).
6. `background-attachment` → scroll behavior ( `scroll` , `fixed` ).
7. `background` → shorthand (all in one line).

## Examples

### Example 1: Background Color

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}
</style>
</head>
<body>
  <h2>Hello CSS Background!</h2>
  <p>This page has a light blue background.</p>
```

```
</body>
</html>
```

#### Explanation:

- `background-color: lightblue;` fills whole page with light blue.

## Example 2: Background Image

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url('https://via.placeholder.com/150');
  background-repeat: repeat-x; /* repeats horizontally */
}
</style>
</head>
<body>
  <h2>Background Image Example</h2>
  <p>Image repeats across the top.</p>
</body>
</html>
```

#### Explanation:

- `background-image` → sets an image.
- `repeat-x` → repeats only horizontally.

## Example 3: Background No Repeat with Position

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
body {
  background-image: url('https://via.placeholder.com/200');
  background-repeat: no-repeat;
  background-position: top right;
}
</style>
</head>
<body>
  <h2>Background Position</h2>
  <p>Image is at the top-right corner.</p>
</body>
</html>
```

#### Explanation:

- `no-repeat` → only one image.
- `top right` → aligns image to top-right.

## Example 4: Background Size Cover

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url('https://picsum.photos/800/600');
  background-size: cover;
  background-repeat: no-repeat;
  background-attachment: fixed;
}
</style>
</head>
<body>
  <h1>Full Background Cover</h1>
  <p>Resize window → image always covers screen.</p>
```

```
</body>
</html>
```

 Explanation:

- `cover` → image fills screen (may crop).
- `fixed` → image stays still when scrolling.

## Example 5: Background Gradient

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background: linear-gradient(to right, lightblue, lightgreen);
}
</style>
</head>
<body>
  <h1>Gradient Background</h1>
  <p>From blue to green.</p>
</body>
</html>
```

 Explanation:

- `linear-gradient(to right, color1, color2)` → smooth transition between colors.

 Explanation:

- Gradient background makes card attractive.
- Circle profile picture with white border.

## Practice Questions

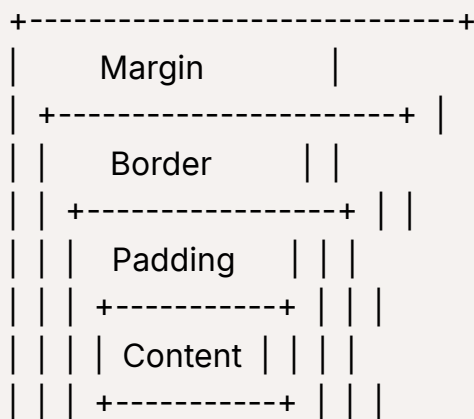
1. Set a **solid background color** for a page.
2. Add a **background image** and make it **no-repeat**.
3. Place an image at the **bottom center** using `background-position`.
4. Create a page with **full-screen background image** (`cover`).
5. Create a box with **gradient background** from red to yellow.

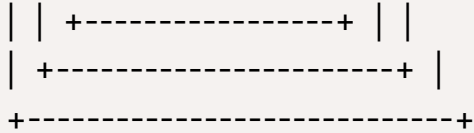
## Section E : CSS Box Model

### 📌 Notes

- Every element in HTML is a **box**.
- The box model has 4 main parts:
  1. **Content** → the actual text or image inside the box.
  2. **Padding** → space between content and border.
  3. **Border** → a line around the padding and content.
  4. **Margin** → space between this box and other elements.
- `width` and `height` set the size of the **content area**.
- `box-sizing` controls how width/height are calculated.

### 📌 ASCII Diagram of Box Model





## Syntax

```
selector {  
  width: value;  
  height: value;  
  padding: value;  
  border: value;  
  margin: value;  
  box-sizing: content-box | border-box;  
}
```

## Example 1: Basic Box Model

```
<!DOCTYPE html>  
<html>  
  <head>  
    <style>  
      div {  
        width: 200px;  
        height: 100px;  
        padding: 20px;  
        border: 5px solid black;  
        margin: 30px;  
        background-color: lightblue;  
      }  
    </style>  
  </head>  
  <body>  
    <div>Box Model Example</div>
```

```
</body>
</html>
```

### Explanation:

- `width: 200px; height: 100px;` → content area size
- `padding: 20px;` → space inside around text
- `border: 5px solid black;` → border around box
- `margin: 30px;` → space outside the box
- Background helps us see the box clearly

### ✓ Example 2: Content-Box vs Border-Box

```
<!DOCTYPE html>
<html>
<head>
<style>
  .content-box {
    width: 200px;
    padding: 20px;
    border: 5px solid red;
    box-sizing: content-box;
    background: lightyellow;
  }

  .border-box {
    width: 200px;
    padding: 20px;
    border: 5px solid green;
    box-sizing: border-box;
    background: lightpink;
  }
</style>
</head>
```

```

<body>
  <div class="content-box">Content-Box (Total width > 200px)</div>
  <div class="border-box">Border-Box (Total width = 200px)</div>
</body>
</html>

```

### Explanation:

- `.content-box` → width is only for content (padding + border increase total size).
- `.border-box` → width includes padding + border (total size stays fixed at 200px).

### ✓ Example 3: Visual Comparison of Margins

```

<!DOCTYPE html>
<html>
<head>
  <style>
    .box1 {
      width: 150px;
      height: 80px;
      background: lightgreen;
      margin: 20px;
    }
    .box2 {
      width: 150px;
      height: 80px;
      background: lightcoral;
      margin: 50px;
    }
  </style>
</head>
<body>
  <div class="box1">Box 1</div>
  <div class="box2">Box 2</div>

```

```
</body>
</html>
```

### Explanation:

- Both boxes have same size.
- Different margin values push them apart differently.



### Practice Questions

1. Create a box of width 250px, height 150px, padding 15px, border 3px solid black, and margin 40px.
2. Make two boxes: one using `box-sizing: content-box` and another using `box-sizing: border-box`. Compare their widths.
3. Create three boxes with different margin values (10px, 30px, 50px) and observe spacing.
4. Change only the **padding** value and notice how the background area grows.

✓ That completes the **Box Model section**.

## (A) Margin and Padding in Detail

Margins and padding are **two important parts of the CSS Box Model**. They control the **space around elements** but in different ways.



### Notes

- **Margin** → Space **outside** the element's border (between elements).
- **Padding** → Space **inside** the element's border (between content and border).
- Both can be written in **1-value, 2-value, 3-value, or 4-value shorthand formats**.

## 1. Single Value (All Sides Same)

### ◆ Syntax

```
selector {  
  margin: 20px;  
  padding: 10px;  
}
```

### ◆ Example

```
<!DOCTYPE html>  
<html>  
  <head>  
    <style>  
      .box {  
        background-color: lightblue;  
        margin: 20px; /* 20px space outside on all 4 sides */  
        padding: 10px; /* 10px space inside on all 4 sides */  
      }  
    </style>  
  </head>  
  <body>  
    <div class="box">Hello! I am inside a box.</div>  
  </body>  
</html>
```

### ◆ Explanation

- The box has **20px gap from outside elements** (margin).
- The text has **10px breathing space inside the box** (padding).

## 2. Two Values (Top-Bottom | Left-Right)

## ◆ Syntax

```
selector {  
  margin: 30px 50px;  
  padding: 10px 25px;  
}
```

## ◆ Example

```
<!DOCTYPE html>  
<html>  
  <head>  
    <style>  
      .box {  
        background-color: lightgreen;  
        margin: 30px 50px; /* 30px top+bottom, 50px left+right */  
        padding: 10px 25px; /* 10px top+bottom, 25px left+right */  
      }  
    </style>  
  </head>  
  <body>  
    <div class="box">This box uses 2-value margin and padding.</div>  
  </body>  
</html>
```

## ◆ Explanation

- First value → **top & bottom**.
- Second value → **left & right**.

## 📌 3. Four Values (Top | Right | Bottom | Left)

### ◆ Syntax

```
selector {  
  margin: 10px 20px 30px 40px;  
  padding: 5px 15px 25px 35px;  
}
```

## ◆ Example

```
<!DOCTYPE html>  
<html>  
  <head>  
    <style>  
      .box {  
        background-color: lightcoral;  
        margin: 10px 20px 30px 40px; /* top, right, bottom, left */  
        padding: 5px 15px 25px 35px; /* top, right, bottom, left */  
      }  
    </style>  
  </head>  
  <body>  
    <div class="box">Box with 4-value margin and padding</div>  
  </body>  
</html>
```

## ◆ Explanation

- The values go in **clockwise order** → Top → Right → Bottom → Left.

## 💡 Quick Recap

Values	Meaning
margin: 20px;	All 4 sides 20px
margin: 10px 20px;	10px top+bottom, 20px left+right
margin: 10px 20px 30px 40px;	Top=10, Right=20, Bottom=30, Left=40

(Same for padding!)

---

## Practice Questions

1. Create a box with a **margin of 50px** on all sides and **padding of 15px**.
  2. Create a box with **30px top+bottom** and **60px left+right** margin.
  3. Create a box with **different margin on all sides: 10px top, 20px right, 30px bottom, 40px left**.
  4. Same as above but use **padding** instead of margin.
- 
- 

Perfect 👍 Let's continue your **CSS Handbook** with **Borders**.

---

## (B) Borders in CSS

Borders are used to **draw a line around an element's content and padding**.

---

### Notes

- A border has **3 main properties**:
    1. **Width** → thickness of the border.
    2. **Style** → appearance of the border (solid, dashed, dotted, etc.).
    3. **Color** → color of the border.
  - Borders can be applied on:
    - All sides → `border`
    - Specific sides → `border-top` , `border-right` , `border-bottom` , `border-left`
- 

## Border Syntax

```
selector {  
  border: width style color;  
}
```

✓ Example:

```
.box {  
  border: 3px solid red;  
}
```

## Border Styles

Here are the most common border styles:

- **solid** → Continuous line
- **dashed** → Broken line (dashes)
- **dotted** → Series of dots
- **double** → Two parallel lines
- **groove** → 3D grooved border
- **ridge** → 3D ridge border
- **inset** → Looks like the element is pressed inside
- **outset** → Looks like the element is popping out

## Mini Projects for Each Border Style

Let's create **one small project for each border style** (like a card or box).

### 1. Border: Solid

```
<!DOCTYPE html>  
<html>
```

```
<head>
<style>
.card {
  width: 250px;
  padding: 20px;
  border: 3px solid blue;
  background-color: #f0f8ff;
}
</style>
</head>
<body>
  <div class="card">
    <h3>Solid Border Card</h3>
    <p>This card uses a solid border.</p>
  </div>
</body>
</html>
```

## 2. Border: Dashed

```
<!DOCTYPE html>
<html>
<head>
<style>
.card {
  width: 250px;
  padding: 20px;
  border: 3px dashed green;
  background-color: #f8fff0;
}
</style>
</head>
<body>
  <div class="card">
```

```
<h3>Dashed Border Card</h3>
<p>This card uses a dashed border.</p>
</div>
</body>
</html>
```

### 3. Border: Dotted

```
<!DOCTYPE html>
<html>
<head>
<style>
.card {
  width: 250px;
  padding: 20px;
  border: 3px dotted red;
  background-color: #fff0f5;
}
</style>
</head>
<body>
  <div class="card">
    <h3>Dotted Border Card</h3>
    <p>This card uses a dotted border.</p>
  </div>
</body>
</html>
```

### 4. Border: Double

```
<!DOCTYPE html>
<html>
<head>
```

```
<style>
.card {
  width: 250px;
  padding: 20px;
  border: 5px double purple;
  background-color: #f5f0ff;
}
</style>
</head>
<body>
  <div class="card">
    <h3>Double Border Card</h3>
    <p>This card uses a double border.</p>
  </div>
</body>
</html>
```

## 5. Border: Groove

```
<!DOCTYPE html>
<html>
<head>
<style>
.card {
  width: 250px;
  padding: 20px;
  border: 6px groove brown;
  background-color: #fffaf0;
}
</style>
</head>
<body>
  <div class="card">
    <h3>Groove Border Card</h3>
```

```
<p>This card looks grooved (3D effect).</p>
</div>
</body>
</html>
```

## 6. Border: Ridge

```
<!DOCTYPE html>
<html>
<head>
<style>
.card {
  width: 250px;
  padding: 20px;
  border: 6px ridge teal;
  background-color: #f0ffff;
}
</style>
</head>
<body>
  <div class="card">
    <h3>Ridge Border Card</h3>
    <p>This card looks ridged (3D effect).</p>
  </div>
</body>
</html>
```

## 7. Border: Inset

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```

.card {
  width: 250px;
  padding: 20px;
  border: 6px inset gray;
  background-color: #f5f5f5;
}
</style>
</head>
<body>
  <div class="card">
    <h3>Inset Border Card</h3>
    <p>This card looks like it is pressed inside.</p>
  </div>
</body>
</html>

```

## 8. Border: Outset

```

<!DOCTYPE html>
<html>
<head>
<style>
.card {
  width: 250px;
  padding: 20px;
  border: 6px outset orange;
  background-color: #fffaf0;
}
</style>
</head>
<body>
  <div class="card">
    <h3>Outset Border Card</h3>
    <p>This card looks like it is coming out.</p>
  </div>
</body>
</html>

```

```
</div>
</body>
</html>
```

## Practice Questions

1. Create a **profile card** with a **solid border**.
2. Create a **quote box** with a **dotted border**.
3. Create a **login form** with a **double border**.
4. Create a **warning message box** with a **dashed border**.
5. Try applying different borders on **each side of a box** (`border-top`, `border-right`, etc.).

## (C) Border Radius



### Notes

- `border-radius` makes corners **rounded**.
- Can be used on **all corners** at once or on **individual corners**.
- Common use: buttons, cards, profile images, circles.



### Syntax

```
selector {
  border-radius: value;
}
```

✓ Example:

```
.box {  
  border: 2px solid black;  
  border-radius: 10px; /* rounds corners */  
}
```

## Examples

### Example 1: Rounded Box

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
.box {  
  width: 200px;  
  height: 100px;  
  background: lightblue;  
  border: 2px solid blue;  
  border-radius: 15px;  
}  
</style>  
</head>  
<body>  
  <div class="box">Rounded Box</div>  
</body>  
</html>
```

#### Explanation:

- `border-radius: 15px;` → all four corners rounded.
- Looks smoother compared to sharp edges.

### Example 2: Circle Profile Picture

```

<!DOCTYPE html>
<html>
<head>
<style>
img {
  width: 150px;
  height: 150px;
  border-radius: 50%;
  border: 3px solid purple;
}
</style>
</head>
<body>
  <h3>Circle Profile</h3>
  
</body>
</html>

```

#### Explanation:

- `border-radius: 50%;` → makes image circular.
- Works best on square images (equal width & height).

### Example 3: Different Corner Radius

```

<!DOCTYPE html>
<html>
<head>
<style>
.box {
  width: 220px;
  height: 120px;
  background: lightgreen;
  border: 2px solid green;
  border-radius: 40px 10px 30px 5px;
}

```

```

}
</style>
</head>
<body>
  <div class="box">Different corners</div>
</body>
</html>

```

 Explanation:

- `border-radius: top-left top-right bottom-right bottom-left;`
- Each corner has a **different curve**.

## Mini Project: Rounded Button

```

<!DOCTYPE html>
<html>
<head>
<style>
button {
  background: orange;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 25px;
  font-size: 16px;
  cursor: pointer;
}
button:hover {
  background: darkorange;
}
</style>
</head>
<body>
  <button>Click Me</button>

```

```
</body>
</html>
```

 Explanation:

- `border-radius: 25px;` → pill-shaped button.
- Useful for modern UI buttons.

## Practice Questions

1. Make a **square box** with `border-radius: 20px`.
2. Make a **circle** div using `border-radius: 50%`.
3. Create a **card with only top corners rounded**.
4. Try different values on each corner: `border-radius: 10px 20px 30px 40px;`.

## Section F : CSS Flexbox – Parent Properties



### Notes

- **Flexbox** = Flexible Box Layout.
- Used to **arrange items** inside a container easily.
- Parent element (container) controls layout of its children.
- Very useful for centering, spacing, and responsive design.



### Parent Properties

1. `display: flex;` → turns a container into a flexbox.
2. `flex-direction` → direction of items ( `row` , `row-reverse` , `column` , `column-reverse` ).
3. `justify-content` → alignment on the **main axis** (left-right or top-bottom).

4. `align-items` → alignment on the **cross axis** (top–bottom or left–right).
  5. `gap` → space between items.
- 

## Examples

---

### Example 1: display: flex

```
<!DOCTYPE html>
<html>
<head>
<style>
.container {
  display: flex;
  border: 2px solid black;
}
.box {
  background: lightblue;
  padding: 20px;
  margin: 5px;
}
</style>
</head>
<body>
  <div class="container">
    <div class="box">Box 1</div>
    <div class="box">Box 2</div>
    <div class="box">Box 3</div>
  </div>
</body>
</html>
```

#### Explanation:

- `display: flex;` → children ( `.box` ) arranged in a row by default.
-

## Example 2: flex-direction

```
<!DOCTYPE html>
<html>
<head>
<style>
.container {
  display: flex;
  flex-direction: column;
  border: 2px solid black;
}
.box {
  background: lightgreen;
  padding: 20px;
  margin: 5px;
}
</style>
</head>
<body>
  <div class="container">
    <div class="box">Item A</div>
    <div class="box">Item B</div>
    <div class="box">Item C</div>
  </div>
</body>
</html>
```

 Explanation:

- `flex-direction: column;` → items stacked vertically.

## Example 3: justify-content

```
<!DOCTYPE html>
<html>
<head>
```

```

<style>
.container {
  display: flex;
  justify-content: space-around;
  border: 2px solid black;
}
.box {
  background: orange;
  padding: 20px;
}
</style>
</head>
<body>
  <div class="container">
    <div class="box">One</div>
    <div class="box">Two</div>
    <div class="box">Three</div>
  </div>
</body>
</html>

```

#### Explanation:

- `justify-content: space-around;` → equal space around items.
- Other values: `flex-start`, `flex-end`, `center`, `space-between`, `space-evenly`.

## Example 4: align-items

```

<!DOCTYPE html>
<html>
<head>
<style>
.container {
  display: flex;
  align-items: center;

```

```

height: 200px;
border: 2px solid black;
}
.box {
background: pink;
padding: 20px;
}
</style>
</head>
<body>
  <div class="container">
    <div class="box">Aligned Center</div>
  </div>
</body>
</html>

```

#### Explanation:

- `align-items: center;` → centers item vertically in container.
- Other values: `flex-start`, `flex-end`, `stretch`, `baseline`.

## Example 5: gap

```

<!DOCTYPE html>
<html>
<head>
<style>
.container {
display: flex;
gap: 20px;
border: 2px solid black;
}
.box {
background: lightcoral;
padding: 20px;

```

```
}  
</style>  
</head>  
<body>  
  <div class="container">  
    <div class="box">Gap 1</div>  
    <div class="box">Gap 2</div>  
    <div class="box">Gap 3</div>  
  </div>  
</body>  
</html>
```


 Explanation:

- `gap: 20px;` → adds equal spacing between items.
- Easier than using `margin`.



## Practice Questions

1. Create a flex container with 4 boxes in a **row**.
2. Change direction to **column** using `flex-direction`.
3. Use `justify-content: center;` to center items horizontally.
4. Make items **vertically centered** using `align-items`.
5. Add `gap: 15px;` between items.

 With just these 5 properties, you can **build simple layouts, center items, and control spacing** easily.

## # Projects:

## Task: Centered Card Layout with Flexbox

👉 Goal: Make 3 cards in a row, evenly spaced, centered both vertically and horizontally, with gaps.

 **Code (Copy-Paste and Run)**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Flexbox Task</title>
  <style>
    body {
      margin: 0;
      height: 100vh;
      display: flex;          /* 1. Enable flexbox */
      flex-direction: row;    /* 2. Arrange in row */
      justify-content: center; /* 3. Center horizontally */
      align-items: center;     /* 4. Center vertically */
      gap: 20px;              /* 5. Space between items */
      background: #f3f3f3;
    }

    .card {
      background: white;
      padding: 20px;
      border: 2px solid #333;
      width: 150px;
      text-align: center;
      font-family: Arial, sans-serif;
      border-radius: 10px;
      box-shadow: 2px 2px 6px rgba(0,0,0,0.1);
```

```
}  
</style>  
</head>  
<body>  
  <div class="card">Card 1</div>  
  <div class="card">Card 2</div>  
  <div class="card">Card 3</div>  
</body>  
</html>
```



## Explanation

- `display: flex;` → turns `body` into a flex container.
- `flex-direction: row;` → items sit side by side.
- `justify-content: center;` → moves cards to the middle horizontally.
- `align-items: center;` → moves cards to the middle vertically (page center).
- `gap: 20px;` → creates space between cards.

Each `.card` is styled with padding, border, radius, and shadow so it looks like a real card.



## Practice Questions

1. Change `flex-direction: row;` to `column;`. What happens?
2. Change `justify-content: center;` to `space-between;`. How do the cards move?
3. Increase the `gap` to `50px`. What do you see?
4. Add one more `.card`. How does the layout adjust?