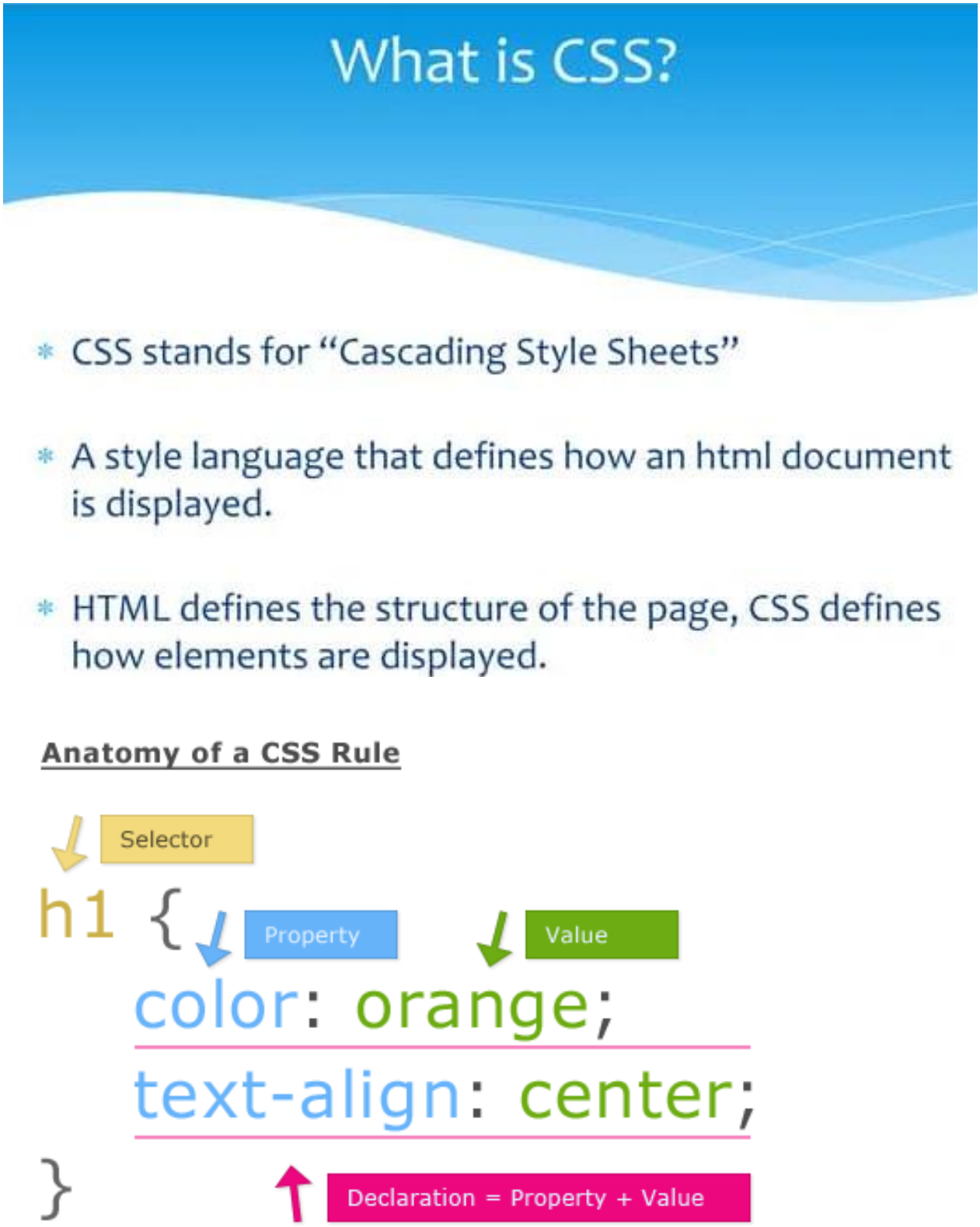
CSS(Cascading Style Sheets)



Types of CSS

1. Inline CSS

- CSS is written **directly inside an HTML element** using the `style` attribute.
- Applies styles only to that specific element.

- ✅ Useful for quick testing.
- ❌ Not recommended for large projects (hard to maintain).

Example:

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1 style="color: blue; text-align: center;">Hello World</h1>
</body>
</html>
```

Hello World

---

2. Internal CSS (Embedded CSS)

- CSS is written inside the `<style>` tag in the `<head>` section of the HTML file.
- Styles apply only to that specific HTML page.
- ✅ Easier than inline for small projects.
- ❌ Not reusable across multiple pages.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    h1 {
      color: red;
      text-align: center;
    }
    p {
      font-size: 18px;
    }
  </style>
</head>
<body>
  <h1>Hello World</h1>
  <p>This is internal CSS.</p>
</body>
</html>
```

Hello World

This is internal CSS.

---

3. External CSS

- CSS is written in a **separate `.css` file** and linked with the HTML using `<link>`.
- Styles can be applied to multiple web pages.
- ✅ Best practice for real projects (clean, reusable, easy to manage).

Example (HTML + CSS file):

index.html

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Hello World</h1>
  <p>This is external CSS.</p>
</body>
</html>
```

style.css

```css
h1 {
  color: green;
}
p {
  font-size: 18px;
  color: red;
}
```

Hello World

This is external CSS.

📌 **Summary:**

- **Inline CSS** → For single elements.
- **Internal CSS** → For one webpage.
- **External CSS** → For multiple webpages (best option).

CSS selectors

CSS selectors are **used to "find" (or select) the HTML elements you want to style**.

1. Universal Selector (*)

- Selects **all elements** on the page.

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
```

2. Element Selector (Type Selector)

- Selects all elements of a specific type.

```
p {

  color: blue;

}
```

3. Class Selector (.classname)

- Selects elements with a specific class attribute.

```
.text-center {

  text-align: center;

}
```

```
<p class="text-center">Hello</p>
```

4. ID Selector (#idname)

- Selects an element with a specific ID.

```
#main-title {

  font-size: 24px;

}
```

```
<h1 id="main-title">Welcome</h1>
```

5. Group Selector (,)

- Applies the same style to multiple selectors.

```
h1, h2, p {
  font-family: Arial, sans-serif;
}
```

6. Descendant Selector (space)

- Selects elements inside another element.

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

→ Targets only <p> inside a .

🎨 CSS Colors

You can set **text color**, **background color**, or even **border color**.

Color → Sets the text color

Ways to define colors:

1. **Named colors** → e.g., red, blue, green

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

2. **Hexadecimal values** → #RRGGBB

```
h1 {
  color: #ff5733; /* orange */
}
```

3. **RGB values** → rgb(red, green, blue)

```
h2 {
  color: rgb(0, 128, 255); /* light blue */
```

```
}
```

🖼 CSS Background

Used to style the **background** of elements.

1. background-color → Sets background color.

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

2. background-image → Sets an image as background.

```
div {
  background-image: url("background.jpg");
}
```

3. background-repeat → Controls repetition of background image.

```
div {
  background-image: url("pattern.png");
  background-repeat: repeat-x; /* repeat only horizontally */
}
```

Values:

- repeat (default)
- no-repeat
- repeat-x (horizontal only)
- repeat-y (vertical only)

4. background-size → Adjusts image size.

```
div {
  background-image: url("nature.jpg");
  background-size: cover;  /* fills area, keeps aspect ratio */
}
```

Values:

- auto (default)
- cover (fills whole area)

- contain (fits without cutting)
- px / % (manual size)

5. background-position → Sets image position.

```
div {
  background-image: url("sun.png");
  background-position: center top;
}
```

6. background-attachment

Defines whether background scrolls with page.

```
div {
  background-image: url("wall.jpg");
  background-attachment: fixed; /* stays fixed while scrolling */
}
```

7. Shorthand → background

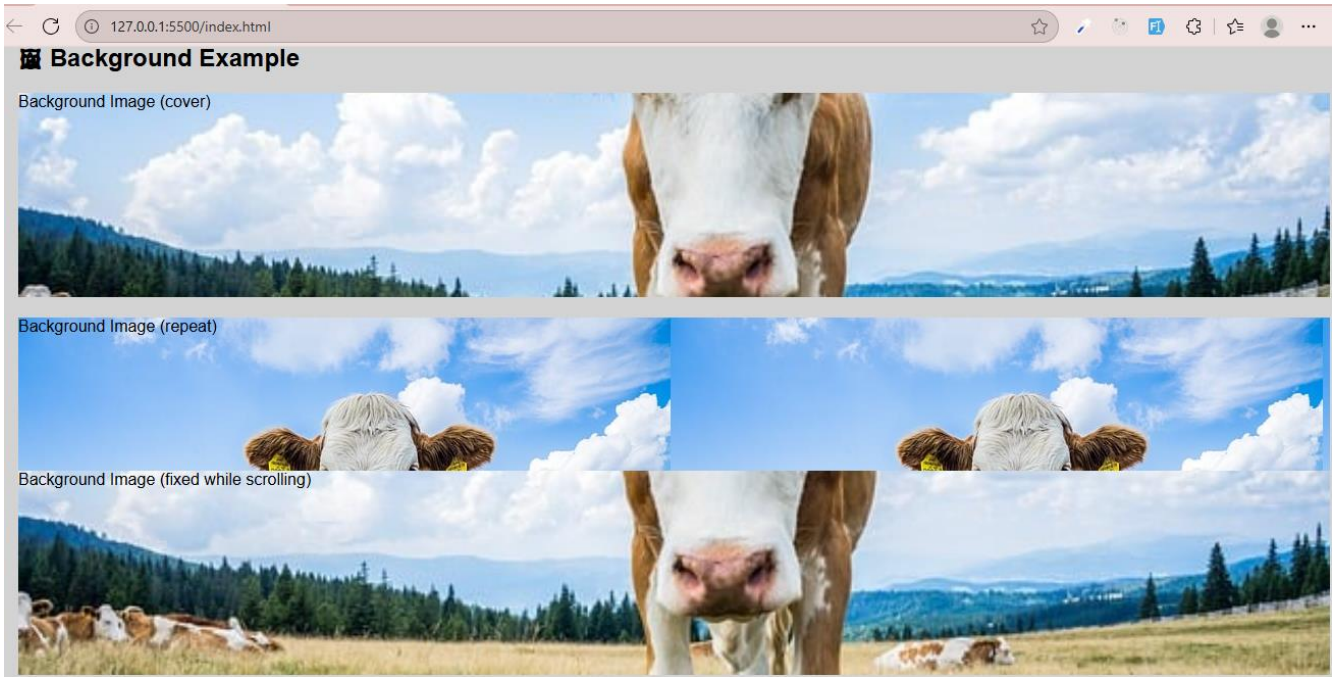
You can combine all properties in one line.

```
div {
  background: url("sky.jpg") no-repeat center/cover fixed;
}
```

Visual Example

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Colors and Background Example</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 0;
      padding: 0;
    }
    .background-section {
      padding: 20px;
      margin-top: 20px;
      background-color: lightgray;
    }
    .bg-image {
```

```
      height: 200px;
      background-image: url("https://cdn.pixabay.com/photo/2025/09/05/18/50/cow-
9817881_640.jpg");
      background-repeat: no-repeat;
      background-size: cover;
      background-position: center;
      margin-bottom: 20px;
    }
    .bg-repeat {
      height: 150px;
      background-image: url("https://cdn.pixabay.com/photo/2025/09/05/18/50/cow-
9817881_640.jpg");
      background-repeat: repeat;
    }
    .bg-fixed {
      height: 200px;
      background-image: url("https://cdn.pixabay.com/photo/2025/09/05/18/50/cow-
9817881_640.jpg");
      background-attachment: fixed;
      background-size: cover;
    }
  </style>
</head>
<body>
  <div class="background-section">
    <h2>🖼 Background Example</h2>
    <div class="bg-image">Background Image (cover)</div>
    <div class="bg-repeat">Background Image (repeat)</div>
    <div class="bg-fixed">Background Image (fixed while scrolling)</div>
  </div>

</body>
</html>
```



✍ CSS Text Styling

1. font-size → Changes text size

```
p {
  font-size: 20px;   /* can be px, em, rem, % */
}
```

2. font-family → Defines the font

```
body {
  font-family: Arial, Verdana, sans-serif;
}
```

👉 Sets the type of font. If Arial isn't available, it will use Verdana, then any sans-serif font.

3. font-weight → Sets thickness

```
h1 {
  font-weight: bold;   /* normal | bold | lighter | 100–900 */
}
```

📌 Example:

- font-weight: normal; → Default weight
- font-weight: bold; → Thick text
- font-weight: 300; → Light text

4. text-align → Aligns text

```
p {
  text-align: center;   /* left | right | center | justify */
}
```

📌 Example:

- left → Default, text starts from left side
- center → Text is centered
- right → Text moves to right side
- justify → Text spreads evenly across the width

5. text-decoration → Adds or removes underline/line 👉 Used for links, headings, or emphasis.

```
a {
  text-decoration: none;       /* remove underline */
}
p {
  text-decoration: underline;  /* adds underline */
}
h2 {
  text-decoration: line-through; /* adds strike-through */
}
```

6. line-height → Controls spacing between lines

```
p {

  line-height: 1.8;   /* can be number, px, em */

}
```

📌 Example:

- line-height: 1; → Tight lines
- line-height: 2; → Spacious lines

7. letter-spacing → Adjusts space between letters 👉 Useful for headings or stylized text.

```
p {

  letter-spacing: 5px;

}
```

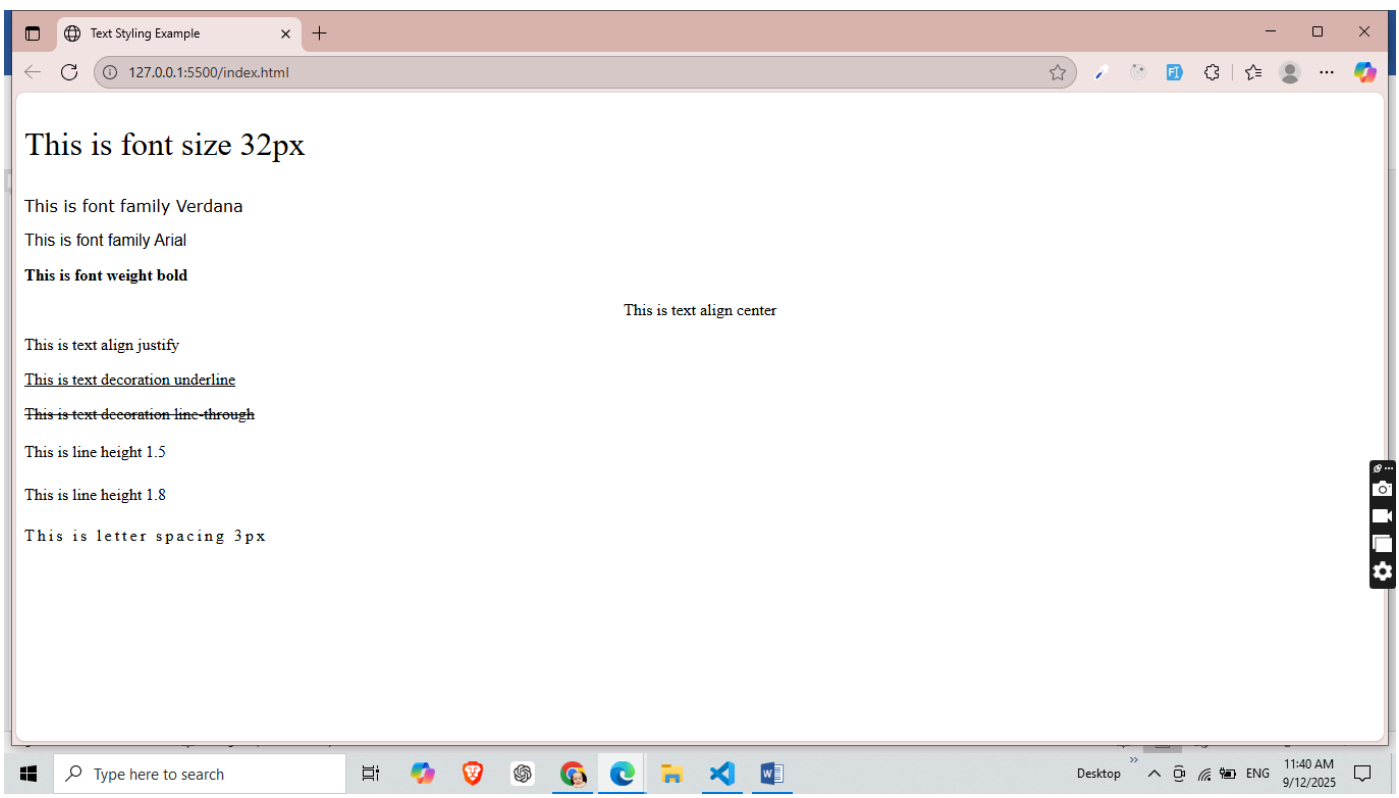
📌 Example:

- letter-spacing: normal; → Default spacing
- letter-spacing: 3px; → Extra spacing between letters

Visual Example

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Text Styling Example</title>
  <style>
    .p1 { font-size: 32px; }
    .p2 { font-family: Verdana, sans-serif; }
    .p3 { font-family: Arial, sans-serif; }
    .p4 { font-weight: bold; }
    .p5 { text-align: center; }
    .p6 { text-align: justify; }
```

```html
    .p7 { text-decoration: underline; }
    .p8 { text-decoration: line-through; }
    .p9 { line-height: 1.5; }
    .p10 { line-height: 1.8; }
    .p11 { letter-spacing: 3px; }
  </style>
</head>
<body>
  <p class="p1">This is font size 32px</p>
  <p class="p2">This is font family Verdana</p>
  <p class="p3">This is font family Arial</p>
  <p class="p4">This is font weight bold</p>
  <p class="p5">This is text align center</p>
  <p class="p6">This is text align justify</p>
  <p class="p7">This is text decoration underline</p>
  <p class="p8">This is text decoration line-through</p>
  <p class="p9">This is line height 1.5</p>
  <p class="p10">This is line height 1.8</p>
  <p class="p11">This is letter spacing 3px</p>

</body>
</html>
```

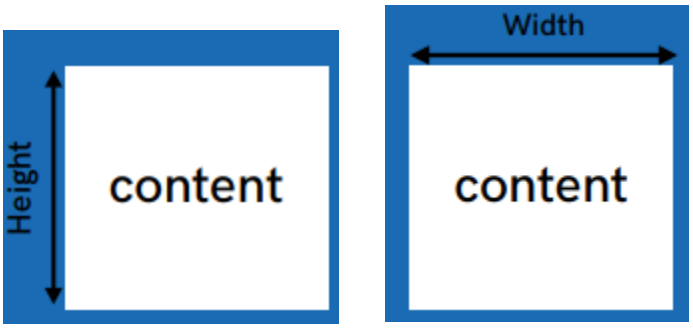


📦 CSS Box Model

Every element in a webpage is like a **box**, and it consists of:

Content → Padding → Border → Margin

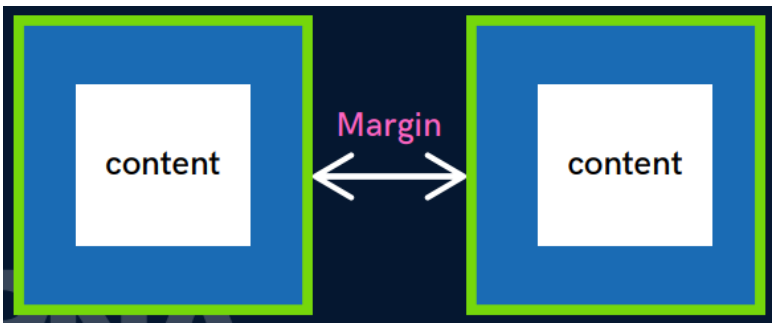
1. width / height → Sets element size




```
div {
  width: 200px;
  height: 200px;
  background-color: lightblue;
}
```

👉 Defines the actual size of the **content area** (not including padding, border, margin).

2. margin → Space outside the element




```
div {

  margin: 20px;

}
```

👉 Creates space between this element and others.

📌 Example: margin: 20px; adds **20px space outside** on all sides.

Individual margin

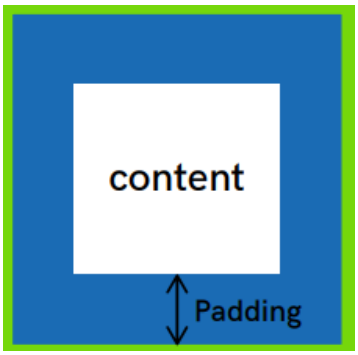
- `margin-top`: Sets the top margin.
- `margin-right`: Sets the right margin.
- `margin-bottom`: Sets the bottom margin.
- `margin-left`: Sets the left margin.

Shorthand property

The `margin` shorthand property allows you to declare values for one to four sides in a single line.

- `margin: 25px;` (All sides are 25px)
- `margin: 25px 50px;` (Top and bottom are 25px, left and right are 50px)
- `margin: 25px 50px 75px;` (Top is 25px, left and right are 50px, bottom is 75px)
- `margin: 25px 50px 75px 100px;` (Top is 25px, right is 50px, bottom is 75px, left is 100px)

3. padding → Space inside the element (between content & border)




```
div {
  padding: 15px;
}
```

👉 Creates space inside the box, pushing the content away from the border.
📌 Example: Adds **15px inner spacing** around the text.

Individual properties

- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

Shorthand property

The `padding` shorthand property allows you to declare values for one to four sides in a single line.

- `padding: 15px;` (All sides are 15px)
- `padding: 10px 20px;` (Top/bottom are 10px, left/right are 20px)
- `padding: 10px 20px 30px;` (Top is 10px, left/right are 20px, bottom is 30px)
- `padding: 10px 20px 30px 40px;` (Top is 10px, right is 20px, bottom is 30px, left is 40px)

4. border → Sets border width, style, and color


```
div {
  border: 3px solid black; /* border-width : 3px; border-style : solid ;border-color : black */
}
```

👉 Outlines the element's box.
📌 Example: 3px solid black → a **solid black border** of 3px width.

- border-style : solid / dotted / dashed

border-radius: Used to round the corners of an element's outer border edge

- border-radius : 10px;
- border-radius : 50%;

5. box-sizing → Controls how size is calculated

```
div {
  width: 200px;
  padding: 20px;
  border: 5px solid red;
  box-sizing: border-box;
}
```

👉 Defines if width includes padding & border.

- content-box (default) → width = **content only** (padding & border added separately).
- border-box → width = **content + padding + border** (everything fits inside 200px)

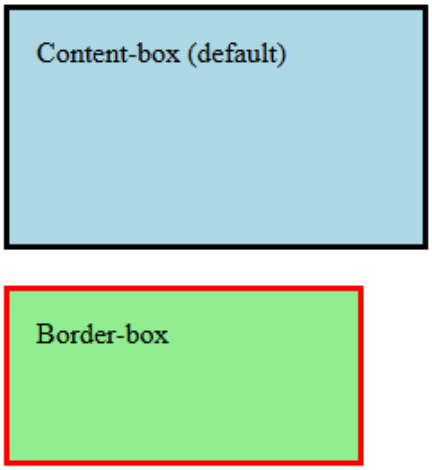
✅ Visual Example of box-sizing

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Box Model Example</title>
  <style>
    .box1 {
      width: 200px;
      height: 100px;
      margin: 20px;
      padding: 15px;
      border: 3px solid black;
      background-color: lightblue;
      box-sizing: content-box; /* default */
    }

    .box2 {
      width: 200px;
      height: 100px;
      margin: 20px;
      padding: 15px;
      border: 3px solid red;
      background-color: lightgreen;
      box-sizing: border-box; /* includes padding & border */
    }
  </style>
</head>
<body>
  <h2>Box Model Demo</h2>
  <div class="box1">Content-box (default)</div>
  <div class="box2">Border-box</div>
</body>
</html>
```

📌 In this example:

- **Box 1 (content-box)** → Actual total width = 200px + padding + border
- **Box 2 (border-box)** → Always stays 200px total, even with padding & border

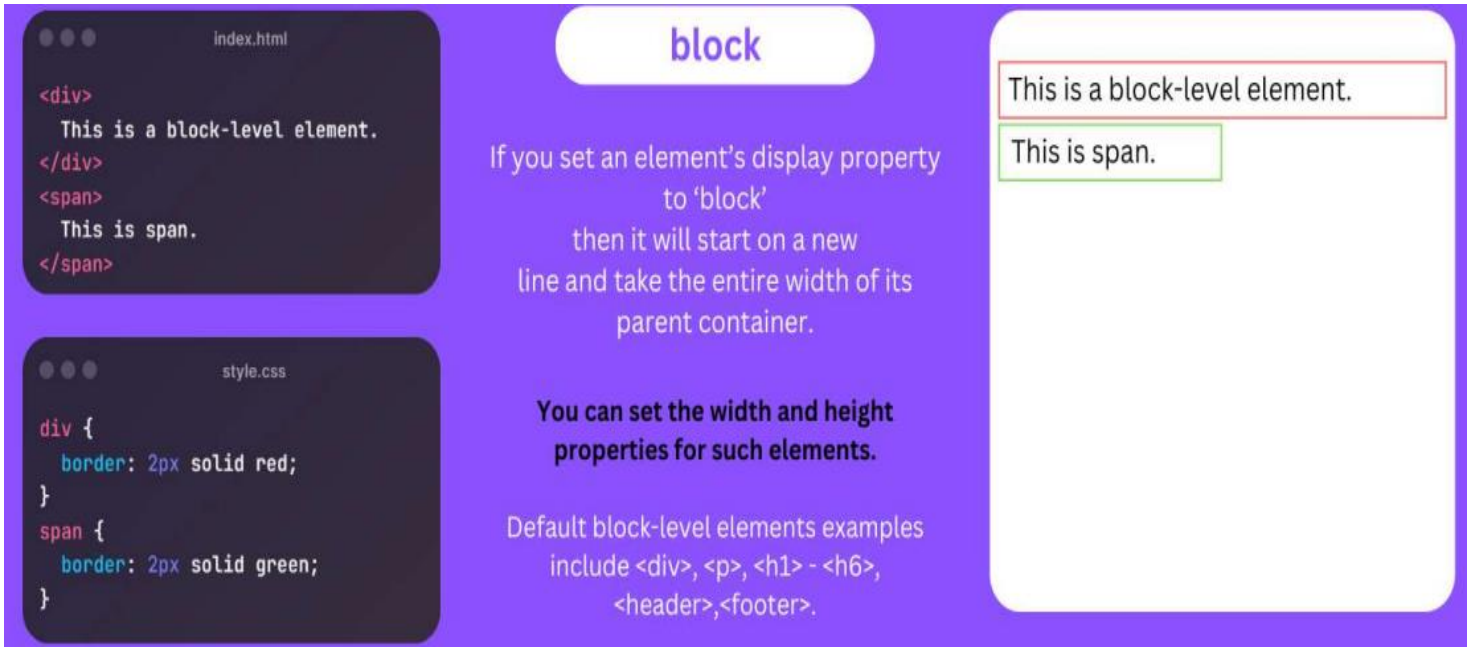
Box Model Demo



🎨 5. Layout Properties in CSS

1. display → Defines how elements are shown

- block → Takes full width (starts on new line). E.g. `<div>`, `<p>`, `<h1>`.
- inline → Takes only needed width (doesn't break line). E.g. `<span>`, `<a>`, `<img>`
- inline-block → Like inline but allows width/height.
- flex → Flexible layout system.
- grid → Grid-based layout.
- none → Hides element.



inline

```html
<span>
  This is a span inline element.
</span>
<a>
  Hyperlink inline element.
</a>
<section>
  This is not a default inline element.
</section>
```

```css
span {
  border: 2px solid green;
}
a {
  border: 2px solid red;
}
section {
  border: 2px solid blue;
}
```

If you set an element's display property to 'inline' then it does not start on a new line and takes the width required by its content.

You can not set the width and height properties for such elements.

Default inline elements examples include , <a>, and <img>.

This is a span inline element. Hyperlink inline element.

This is not a default inline element.

inline-block

```html
<div>
  This is a default block-level element.
</div>
<span>
  Default inline element
  which is set to inline-block.
</span>
```

```css
div {
  border: 2px solid red;
}
span {
  border: 2px solid green;
  height: 200px;
  display: inline-block;
}
```

If you set an element's display property to 'inline-block' then it contains the features of both block and inline elements.

It takes up only as much width as necessary, but you can set its height and width properties.

This is a default block-level element.

Default inline element which is set to inline-block.

flex

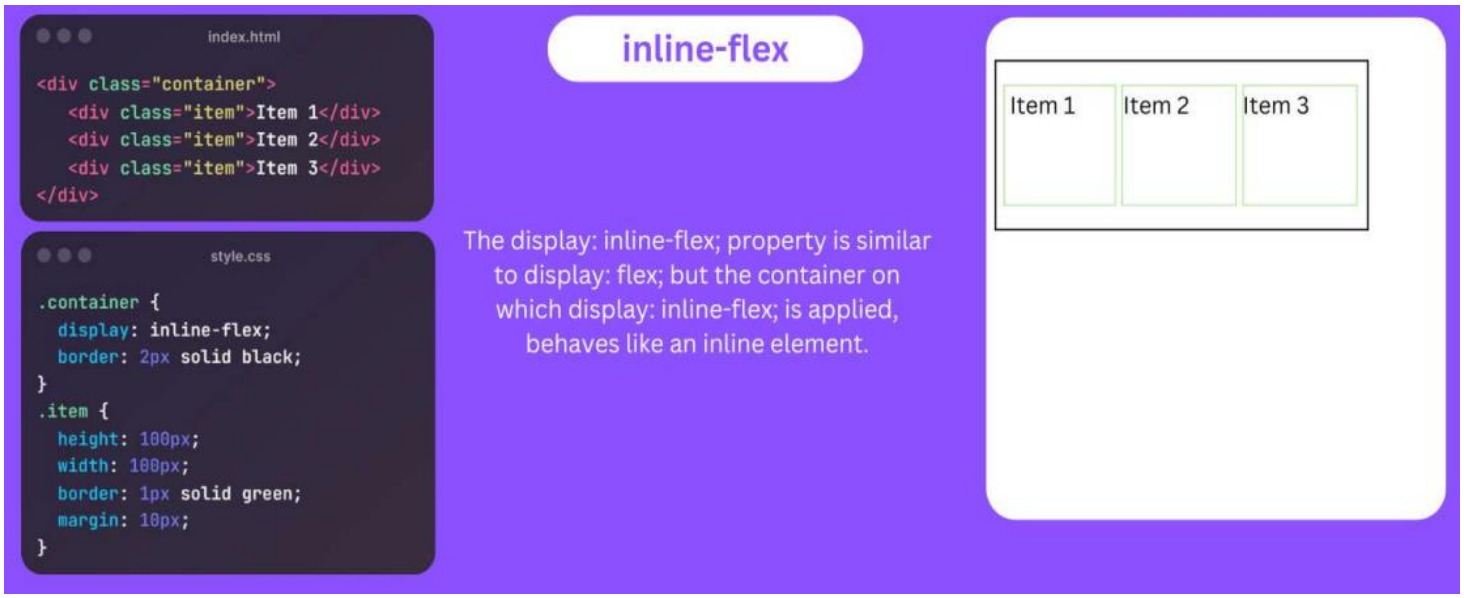
```html
<div class="container">
   <div class="item">Item 1</div>
   <div class="item">Item 2</div>
   <div class="item">Item 3</div>
</div>
```

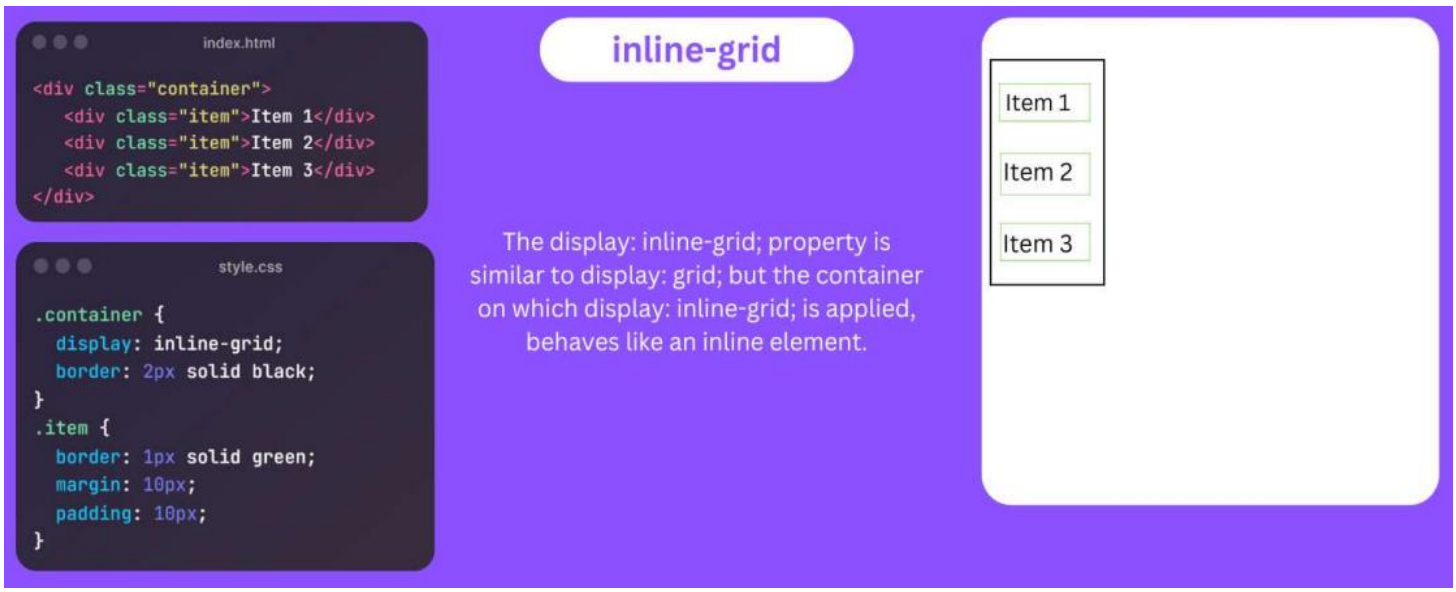
```css
.container {
  display: flex;
  border: 2px solid black;
}
.item {
  height: 100px;
  width: 100px;
  border: 1px solid green;
  margin: 10px;
}
```

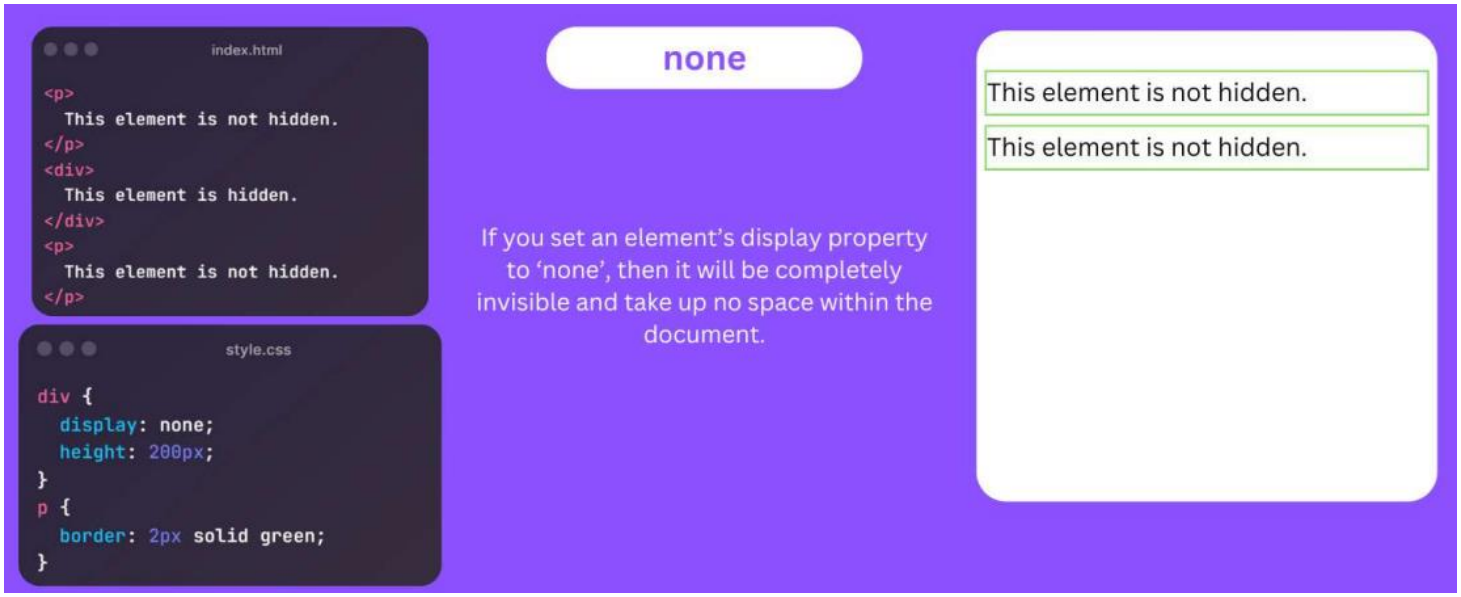
The display: flex; property creates a flex container that can be used for creating flexible layouts.

Item 1 Item 2 Item 3

- **`inline-flex` and `inline-grid`:** These values are similar to `flex` and `grid` respectively, but the container itself behaves like an `inline` element in the document flow, meaning it does not start on a new line.







- Use `display: none` when you want the element **completely removed**.
- Use `visibility: hidden` when you want it **invisible but still occupying space**.

✅ Example Code (All Together)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Display Property Demo</title>
  <style>
    .box {
      padding: 10px;
      margin: 5px;
      background: lightblue;
      border: 2px solid navy;
    }

    /* Different display types */
    .block {
      display: block;
      width: 200px;
    }

    .inline {
      display: inline;
      width: 200px; /* ignored */
      height: 50px; /* ignored */
    }

    .inline-block {
      display: inline-block;
      width: 120px;
      height: 50px;
    }
```

```
    .flex-container {
      display: flex;
      gap: 10px;
    }

    .grid-container {
      display: grid;
      grid-template-columns: 1fr 1fr; /*repeat(2, 1fr)*/

      gap: 10px;
    }

    .child {
      background: lightcoral;
      padding: 10px;
      border: 1px solid darkred;
    }
  </style>
</head>
<body>
  <h2>Block</h2>
  <div class="box block">Block 1</div>
  <div class="box block">Block 2</div>

  <h2>Inline</h2>
  <span class="box inline">Inline 1</span>
  <span class="box inline">Inline 2</span>
  <span class="box inline">Inline 3</span>

  <h2>Inline-block</h2>
  <div class="box inline-block">Inline-block 1</div>
  <div class="box inline-block">Inline-block 2</div>
  <div class="box inline-block">Inline-block 3</div>

  <h2>Flex</h2>
  <div class="flex-container box">
    <div class="child">Flex 1</div>
    <div class="child">Flex 2</div>
    <div class="child">Flex 3</div>
  </div>

  <h2>Grid</h2>
  <div class="grid-container box">
    <div class="child">Grid 1</div>
    <div class="child">Grid 2</div>
    <div class="child">Grid 3</div>
    <div class="child">Grid 4</div>
  </div>
</body>
</html>
```

Block

Block 1

Block 2

Inline

Inline 1 Inline 2 Inline 3

Inline-block

Inline-block 1 Inline-block 2 Inline-block 3

Flex

Flex 1 Flex 2 Flex 3

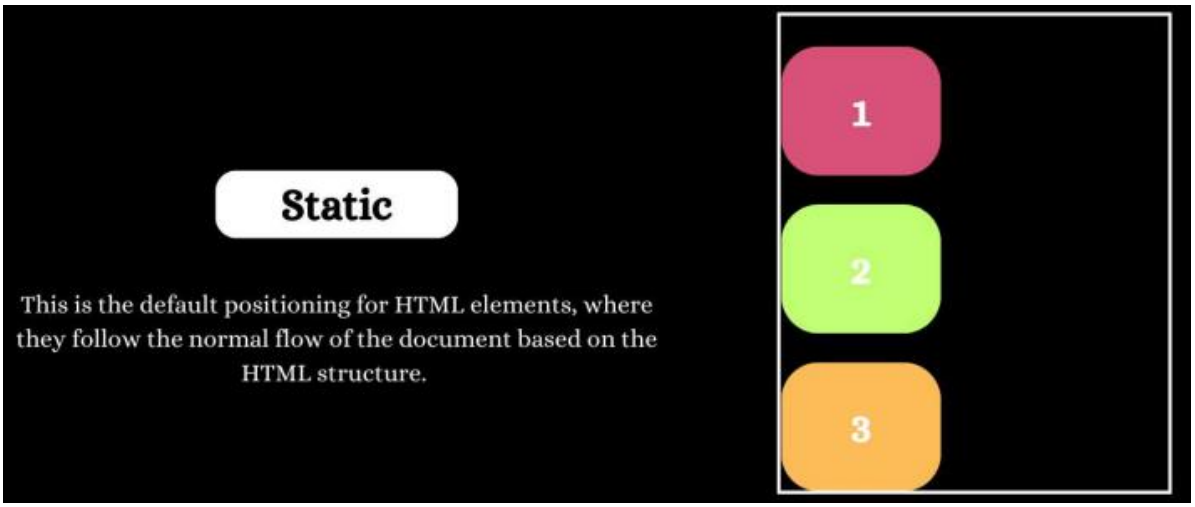
Grid

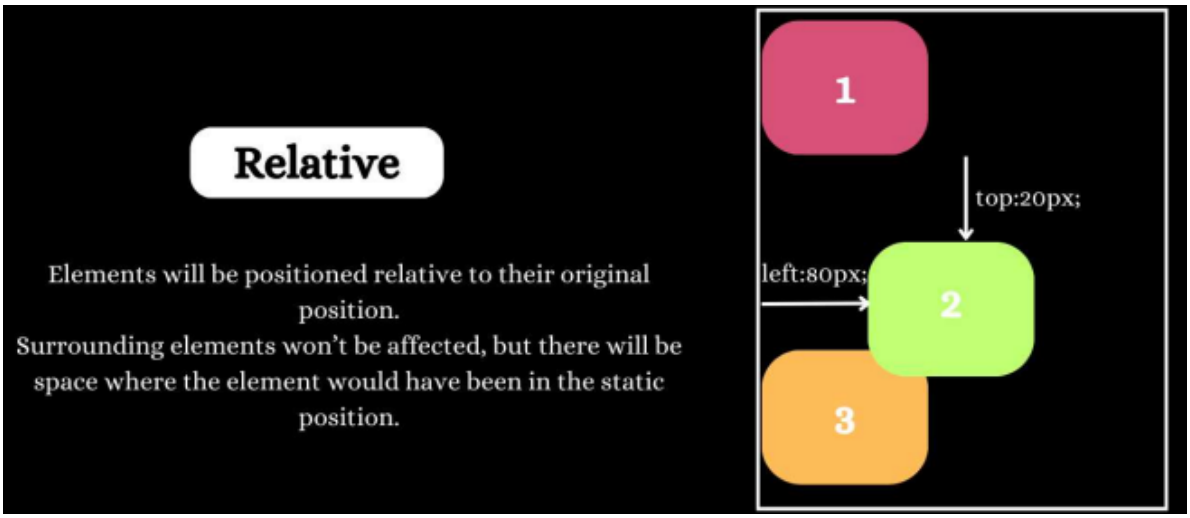
Grid 1 Grid 2

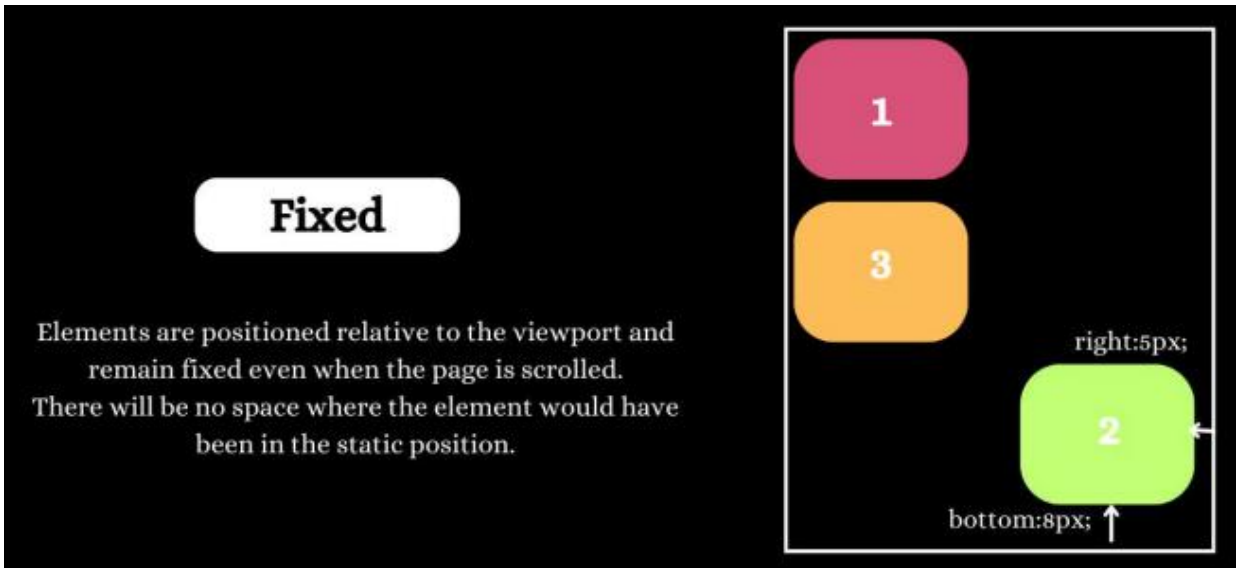
Grid 3 Grid 4

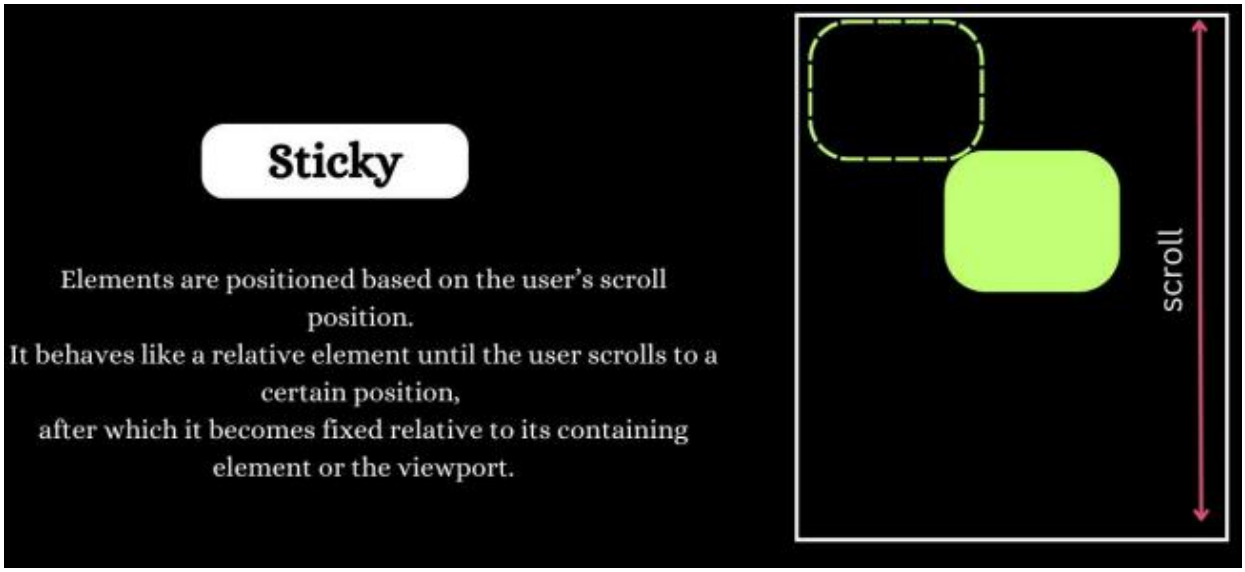
2. position → Controls how elements are placed

- static → Default, follows normal document flow.
- relative → Positioned relative to itself.
- absolute → Positioned relative to its nearest positioned parent.
- fixed → Stays fixed on screen (like navbar).
- sticky → Acts like relative, but sticks to screen when scrolling.









✅ Example Code (All Together)

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Position Property Demo</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      height: 2000px; /* to enable scrolling */
```

```
    }
    .container {
      position: relative; /* parent for absolute */
      margin: 20px;
      padding: 20px;
      border: 2px dashed gray;
      height: 200px;
    }
    .box {
      width: 120px;
      height: 60px;
      background: lightblue;
      border: 2px solid navy;
      margin: 10px;
      padding: 10px;
    }
    /* Static */
    .static {
      position: static; /* default */
    }
    /* Relative */
    .relative {
      position: relative;
      top: 20px;
      left: 30px;
      background: lightgreen;
    }
    /* Absolute */
    .absolute {
      position: absolute;
      top: 20px;
      right: 20px;
      background: lightcoral;
    }
    /* Fixed */
    .fixed {
      position: fixed;
      bottom: 20px;
      right: 20px;
      background: gold;
    }
    /* Sticky */
    .sticky {
      position: sticky;
      top: 0;
      background: violet;
    }
  </style>
</head>
<body>
  <h2>Static (default)</h2>
  <div class="box static">Static Box</div>
```

```html
  <h2>Relative</h2>
  <div class="box relative">Relative Box</div>

  <h2>Absolute (inside container)</h2>
  <div class="container">
    <div class="box absolute">Absolute Box</div>
    <p>I'm the container text. Notice the red box is positioned inside me.</p>
  </div>

  <h2>Fixed</h2>
  <p>Scroll down and notice the yellow "Fixed Box" stays in the bottom-right corner.</p>
  <div class="box fixed">Fixed Box</div>

  <h2>Sticky</h2>
  <p>The purple box below will scroll with the page until it reaches the top, then it will
"stick".</p>
  <div class="box sticky">Sticky Box</div>
  <p style="margin-top:600px;">↓ Keep scrolling to test sticky behavior</p>
</body>
</html>
```

Static (default)

Static Box

Relative

Relative Box

Absolute (inside container)

I'm the container text. Notice the red box is positioned inside me.

Absolute Box

Fixed Box

Fixed

Scroll down and notice the yellow "Fixed Box" stays in the bottom-right corner.

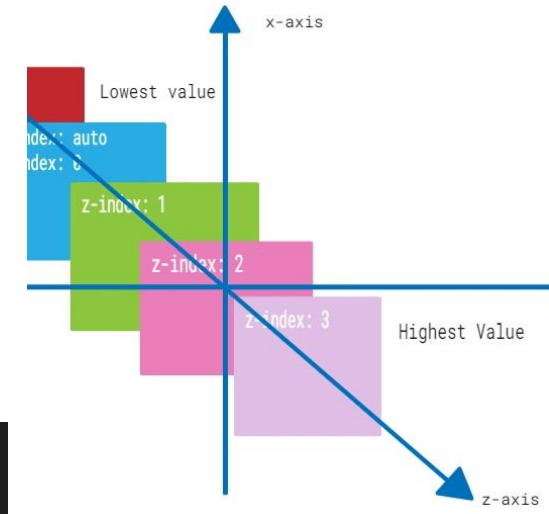
Sticky

The purple box below will scroll with the page until it reaches the top, then it will "stick".

Sticky Box

Fixed Box

z-index → Controls stack order (which element is on top)



- Higher z-index → appears above others.
  - z-index : 0 (default value)
  - z-index : 1/2/…brings elements closer to front
  - z-index : -1/-2…moves element further to back
- z-index only affects elements that have a position value other than static. This includes relative, absolute, fixed, and sticky.

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Document</title>
    <style>
      .box {
        width: 100px;
        height: 100px;
        position: absolute;
        font-weight: bolder;
      }

      .red {
        background: red;
        left: 50px;
        top: 50px;
        z-index: 1; /* This box will appear behind */
      }

      .blue {
        background: blue;
        left: 70px;
        top: 70px;
        z-index: 2; /* This box will appear on top */
      }
    </style>
  </head>
  <body>
    <div class="box red">z-index: 1</div>
    <div class="box blue">z-index: 2</div>
  </body>
</html>
```

3. Float → position elements to the left or right, allowing other content to wrap around them.

```
float: none|left|right|initial|inherit;
```

- **none** → Box sits in normal flow.
- **left** → Box floats left, text wraps on right.
- **right** → Box floats right, text wraps on left.
- **initial** → Same as none (default).
- **inherit** → Takes float value of parent (here parent floats right).

Note: Absolutely positioned elements ignore the float property!

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Float Property Demo</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      line-height: 1.5;
    }
    .container {
      border: 2px dashed gray;
      padding: 15px;
      margin: 20px 0;
    }
    .box {
      width: 120px;
      height: 80px;
      background: lightblue;
      border: 2px solid navy;
      margin: 10px;
      text-align: center;
      line-height: 80px;
      font-weight: bold;
    }
    /* Float variations */
    .left {
      float: left;
    }
    .right {
      float: right;
    }
    .none {
      float: none;
    }
    .initial {
      float: initial; /* resets to default (none) */
    }
    .inherit {
      float: inherit; /* takes float value from parent */
```

```
    }
    /* Parent with float */
    .parent-float {
      float: right;
      border: 2px solid red;
      padding: 15px;
      margin: 20px;
      width: 300px;
    }
    .clearfix::after {
      content: "";
      display: block;
      clear: both; /* clears floats to prevent overlap */
    }
  </style>
</head>
<body>
  <h2>Float: none (default)</h2>
  <div class="container clearfix">
    <div class="box none">None</div>
    <p>Text flows normally. The box is not floating, so it stays above this text in the
document flow.</p>
  </div>

  <h2>Float: left</h2>
  <div class="container clearfix">
    <div class="box left">Left</div>
    <p>This text wraps around the blue box, which is floated to the left side of the
container. Notice how text flows on the right.</p>
  </div>

  <h2>Float: right</h2>
  <div class="container clearfix">
    <div class="box right">Right</div>
    <p>This text wraps around the blue box, which is floated to the right side of the
container. Notice how text flows on the left.</p>
  </div>

  <h2>Float: initial</h2>
  <div class="container clearfix">
    <div class="box initial">Initial</div>
    <p>`float: initial` resets the value back to the default, which is `none`. So this
behaves just like float: none.</p>
  </div>

  <h2>Float: inherit</h2>
  <div class="parent-float clearfix">
    <div class="box inherit">Inherit</div>
    <p>This box inherits float from its parent. Since the parent is floated <b>right</b>,
the child also floats right.</p>
  </div>
</body>
</html>
```

Float: none (default)

None

Text flows normally. The box is not floating, so it stays above this text in the document flow.

Float: left

Left

This text wraps around the blue box, which is floated to the left side of the container. Notice how text flows on the right.

Float: right

This text wraps around the blue box, which is floated to the right side of the container. Notice how text flows on the left.

Right

Float: initial

Initial

`float: initial` resets the value back to the default, which is `none`. So this behaves just like float: none.

Float: inherit

This box inherits float from its parent. Since the parent is floated **right**, the child also floats right.

Inherit

---

clear : Used in conjunction with float, this property specifies which sides of an element floating elements are not allowed to appear.

- left: No floating elements allowed on the left.
- right: No floating elements allowed on the right.
- both: No floating elements allowed on either side.

- **No clear (default)** → paragraph flows beside the float.
- **clear: left** → paragraph goes below the left-floated box.
- **clear: right** → paragraph goes below the right-floated box.
- **clear: both** → paragraph goes below both left and right floats.

4. Flexbox → align and distribute space among items, allowing them to grow or shrink to fill available space.

- `display: flex` → Enables flexbox layout.
- `flex-direction` → Row or column arrangement.
- `justify-content` → Aligns items horizontally (`center`, `space-between`).
- `align-items` → Aligns items vertically.
- `flex-wrap` → Allows items to wrap to next line.

display: flex → Enables flexbox layout (arrange items in a row by default)

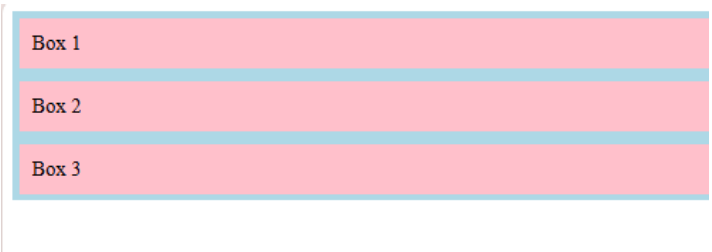
Turns a container into a **flex container**, making its children (items) flexible.

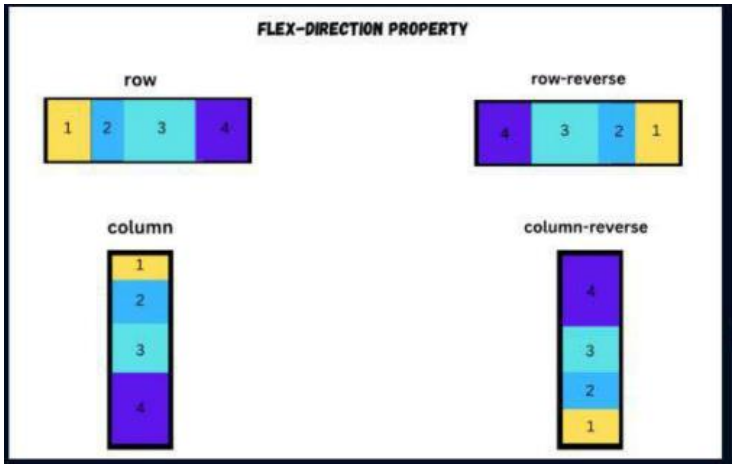
```html
<!DOCTYPE html>

<html lang="en">
<head>
  <title>Flexbox Basics</title>
  <style>
.container {
  display: flex;
  background: lightblue;
}
.item {
  background: pink;
  margin: 5px;
  padding: 10px;
}
  </style>
</head>
<body>
  <div class="container">
  <div class="item">Box 1</div>
  <div class="item">Box 2</div>
  <div class="item">Box 3</div>
</div>
</body>
</html>
```

flex-direction →Controls the **main axis** (the direction of items).

- row (default) → Items placed left to right.
- column → Items placed top to bottom.
- row-reverse → Right to left.
- column-reverse → Bottom to top.

```
.container {
        display: flex;
        flex-direction: column;
        background: lightblue;
      }
```

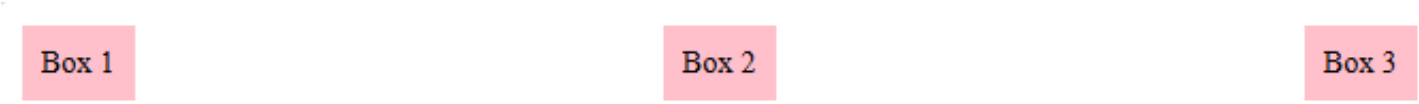


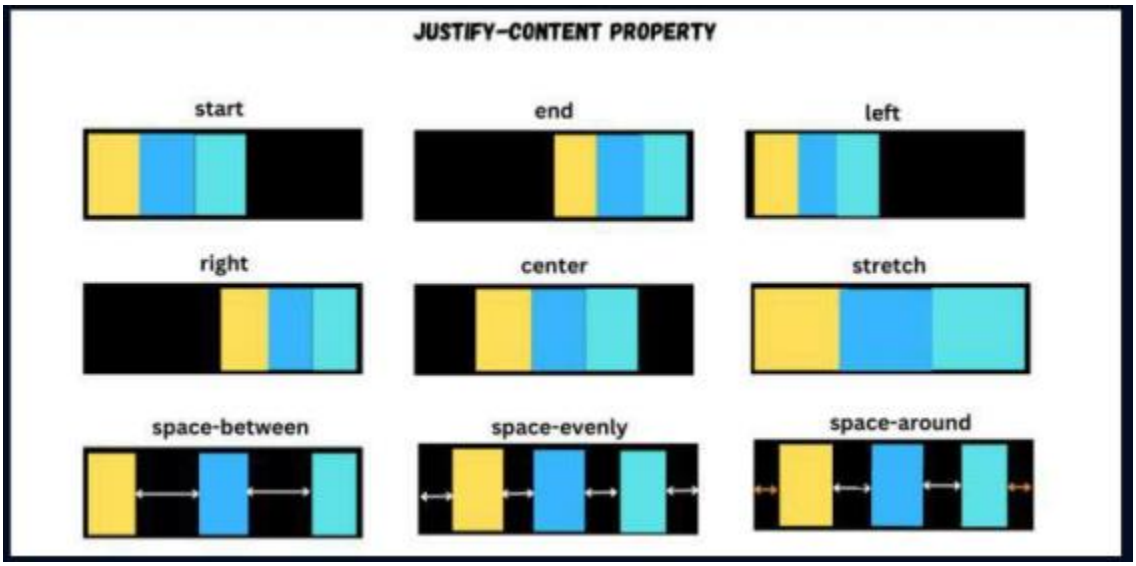


justify-content → horizontal alignment along main axis

- `flex-start` (default) → Items packed at start.
- `center` → Items centered.
- `flex-end` → Items packed at end.
- `space-between` → Equal space **between** items.
- `space-around` → Equal space around items but half before 1st and after last.
- `space-evenly` → Equal space distributed included before 1st and after last.

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

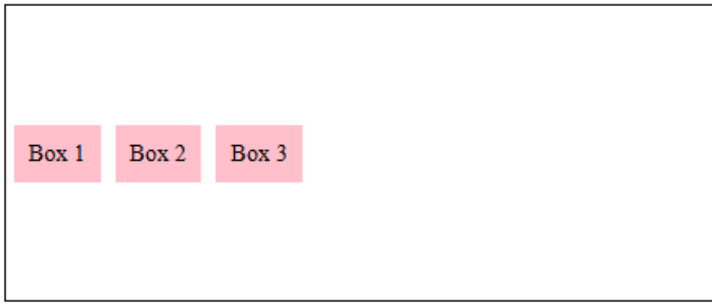


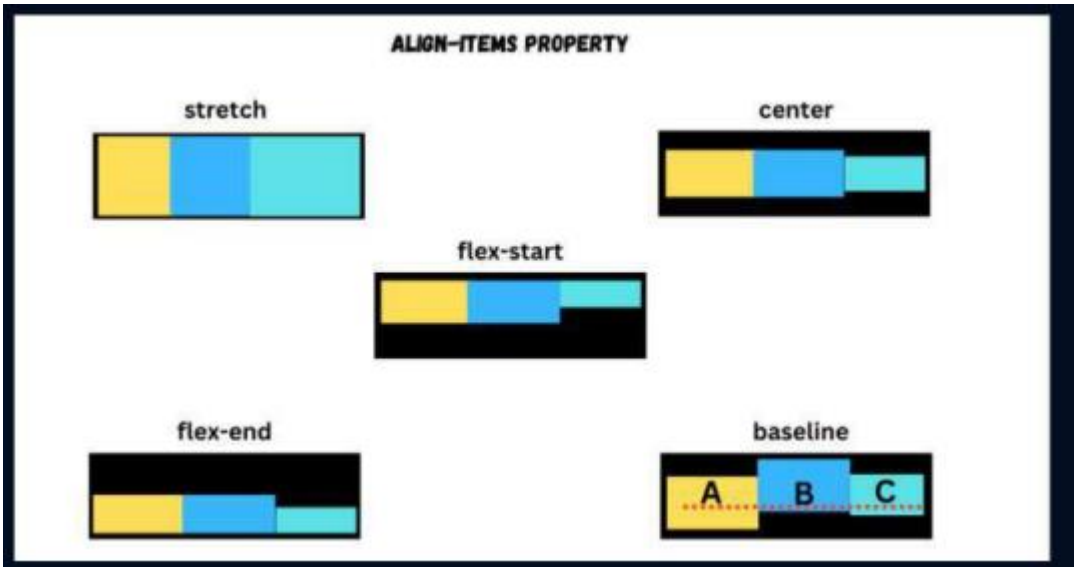


align-items → vertical alignment along cross axis

- `stretch` (default) → Items stretch to fill container height.
- `center` → Items centered vertically.
- `flex-start` → Items at top.
- `flex-end` → Items at bottom.
- `baseline` → Align by text baseline.

```
.container {
      display: flex;
      align-items: center;
     border: 1px solid black;
    height: 200px;
    }
```

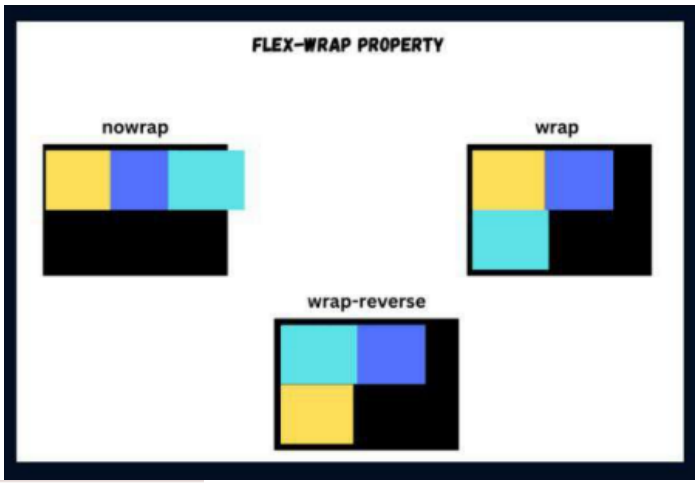


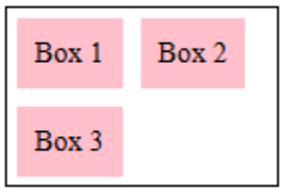


flex-wrap → Allows items to wrap into next line

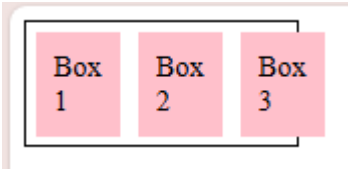
- `nowrap` (default) → All items stay in one line.
- `wrap` → Items wrap onto next line if needed.
- `wrap-reverse` → Wraps but in reverse order.

```
.container {
        display: flex;
        flex-wrap: wrap;
        border: 1px solid black;
        width: 150px;
      }
```

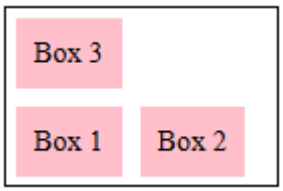




wrap



nowrap(default)

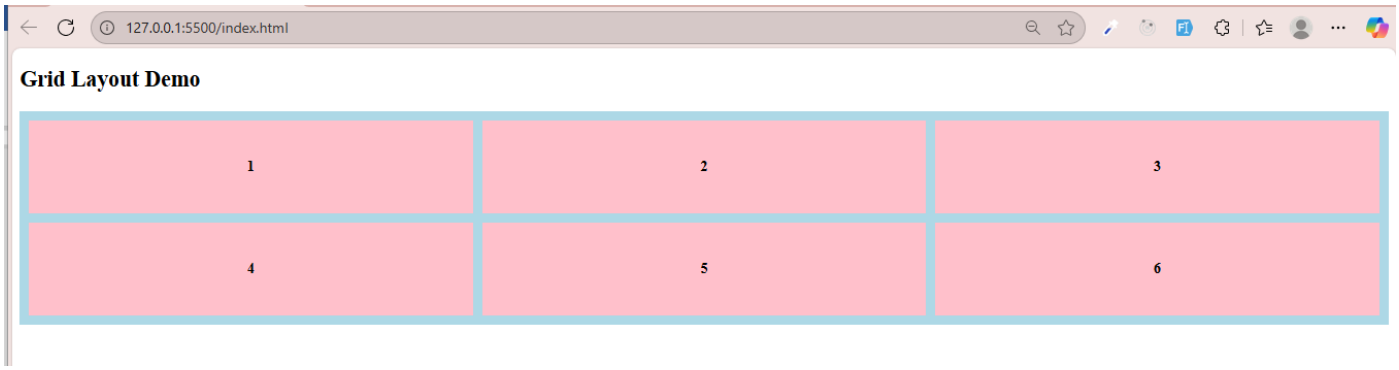


wrap-reverse

5. Grid (Layout Basics)

- **`display: grid`:** Turns a container into a grid layout. Children automatically become grid items.
- **`grid-template-columns`** : Defines how many columns and their widths.
  - `grid-template-columns: 100px 100px 100px;` → 3 equal columns.
  - `grid-template-columns: 1fr 2fr;` → First column takes 1 part, second takes 2 parts.
- **`grid-template-rows`:** Defines rows height.
  - `grid-template-rows: 100px 200px;` → First row 100px, second 200px.
- **`gap`** : Sets spacing between grid items.
  - `gap: 10px;` → 10px gap between rows & columns.
  - `row-gap` and `column-gap` can also be set separately.

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>CSS Grid Example</title>
  <style>
    .container {
      display: grid;
      grid-template-columns: repeat(3, 1fr); /* 3 equal columns */
      grid-template-rows: 100px 100px;       /* 2 equal rows */
      gap: 10px;                             /* space between items */
      background: lightblue;
      padding: 10px;
    }
    .item {
      background: pink;
      display: flex;
      justify-content: center;
      align-items: center;
      font-weight: bold;
    }
  </style>
</head>
<body>
  <h2>Grid Layout Demo</h2>
  <div class="container">
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
    <div class="item">4</div>
    <div class="item">5</div>
    <div class="item">6</div>
  </div>
</body>
</html>
```



6. Other Useful Properties

1. cursor → Changes mouse cursor style

Used for links, buttons, or custom interactions.

```
button {
  cursor: pointer; /* hand icon when hovering */
}

p {
  cursor: default; /* normal arrow */
}
```

Common values:

- `default` → normal arrow
- `pointer` → hand (used for clickable items)
- `text` → I-beam for text selection
- `move` → four arrows
- `not-allowed` → 🚫 icon

2. opacity → Sets transparency (0 to 1)

```
img {
  opacity: 0.5; /* 50% transparent */
}
```

- `opacity: 1;` → Fully visible.
- `opacity: 0.5;` → 50% transparent.
- `opacity: 0;` → Fully invisible (still clickable).

3. overflow → Controls content overflow

- `visible` (default) → overflow is shown
- `hidden` → extra content is clipped
- `scroll` → always adds scrollbars
- `auto` → scrollbars only if needed

```
.box {
  width: 200px;
  height: 100px;
  overflow: auto;
  border: 1px solid black;
}
```

Example code (All together)

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Other Useful CSS Properties</title>
  <style>
    .cursor-example {
      padding: 10px;
      background: lightblue;
      cursor: pointer;
    }
    .opacity-example {
      width: 150px;
      height: 100px;
      background: pink;
      opacity: 0.6;
      margin-top: 10px;
    }
    .overflow-example {
      width: 150px;
      height: 70px;
      background: lightgreen;
      overflow: auto;
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <h2>Other Useful CSS Properties</h2>
  <div class="cursor-example">Hover me (cursor changes)</div>
  <div class="opacity-example">Semi-transparent box</div>
  <div class="overflow-example">
    This is an example of overflow. If the text is too long, you can scroll inside this
box.
  </div>
</body>
</html>
```