🔥 CSS

🎨 What is CSS?

CSS (Cascading Style Sheets) is used to **style and design HTML elements**.

It controls how your website looks, like colors, size, layout, spacing, and animations.

🧾 Simple Definition

CSS is used to add style to HTML elements.

💡 Example without CSS

```
<p>Hello World</p>
```

Output:

Plain text, no color, no style.

💡 Example with CSS

```
<pstyle="color: red; font-size: 20px;">Hello World</p>
```

Output:

Red colored text with bigger size.

CSS Basic Syntax

```
h1 {font-size:30px;
}
```

Selector

Selector is the HTML element that you want to style.

Example:

```
h1
```

It selects the `<h1>` element.

Property

Property is the style you want to change.

Example:

```
font-size
```

Other examples:

```
colorbackground-colormarginpadding
```

Value

Value is the specific setting you give to the property.

Example:

```
30px
```

Other examples:

```
red
blue20px
```

```
bold
```

Declaration

Declaration is the combination of property and value.

Example:

```
font-size:30px;
```

Property → font-size

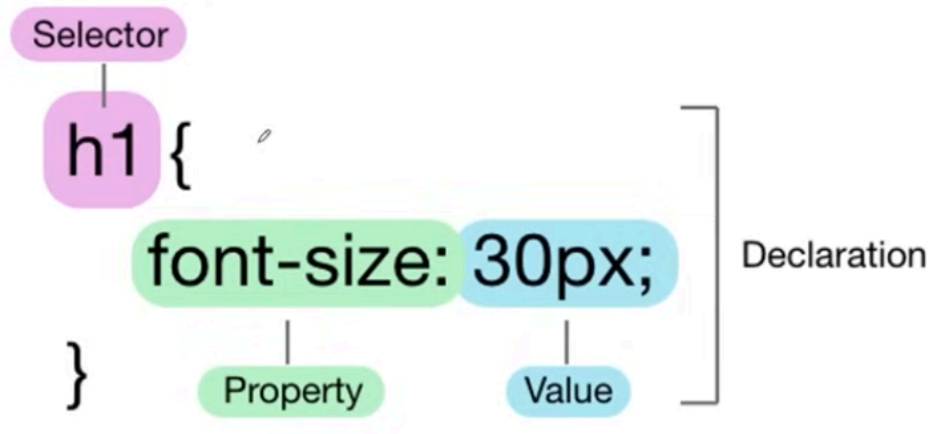
Value → 30px

Complete Structure

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

Example:

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



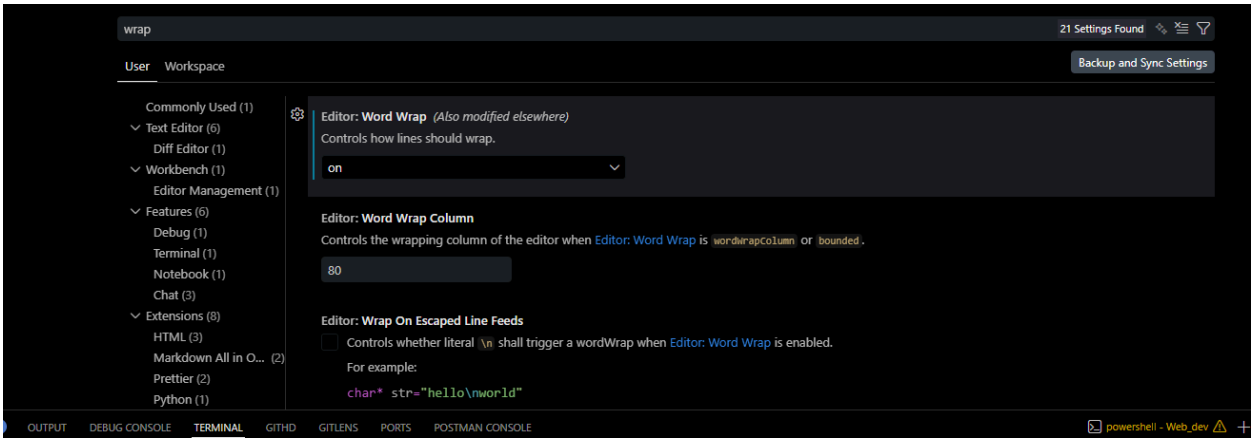
CSS Color Property

Color property is used to change the text color of an HTML element.

```
h1 {
  color: blue;
}

span {
  color: green;
}
```

color property changes the text color. 🎨

How to Wrap a Code in a Vs Code Setting



Inline Styling

Inline styling is used to apply CSS directly inside an HTML element using the `style` attribute.

Example

```
<h1 style="color: red;">Inline Styling</h1>
<p style="color: green;">Paragraph is green</p>
```

Important Note

Used for single element styling only. Not good for large projects.

Internal Styling

Internal CSS is written inside the `<style>` tag in the `<head>` section of HTML.

It is used to style multiple elements on a single page.

It is more organized and reusable than inline CSS.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>Internal Styling</title>

  <style>
    h1 {
      color: red;
    }

    p {
      color: green;
    }
  </style>

</head>
```

```
<body>

  <h1>Internal Styling</h1>
  <p>Paragraph is green</p>

</body>
</html>
```

Important Note

Internal CSS is used to style a full page using the `<style>` tag.

CSS Hierarchy (Priority Order)

Inline CSS > Internal CSS > External CSS

Priority Order with Example

1. Inline CSS (Highest Priority) 🥇

```
<h1style="color: red;">Hello World</h1>
```

This will override internal and external CSS.

2. Internal CSS (Medium Priority) 🥈

```
<style>h1 {color: green;
}</style>
```

This overrides external CSS but not inline CSS.

3. External CSS (Lowest Priority) 🥉

```
h1 {color: blue;
```

```
}
```

This works only if inline and internal CSS are not present.

Example Showing All Three

External CSS (style.css)

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

Internal CSS

```
<style>h1 {color: green;
}</style>
```

Inline CSS

```
<h1style="color: red;">Hello World</h1>
```

Final Output Color 👉 Red

Because:

Inline > Internal > External

One Line to Remember ⭐

Inline CSS has highest priority, External CSS has lowest priority.

Step 1: External CSS file (style.css)

```
h1 {color: blue;
```

```
}
```

Step 2: HTML file (index.html)

```
<!DOCTYPE html><html><head><!-- External CSS --><linkrel="sty
lesheet"href="style.css"><!-- Internal CSS --><style>h1 {colo
r: green;
    }</style></head><body><!-- Inline CSS --><h1style="color:
red;">Hello World</h1></body></html>
```

What happens here?

External CSS → blue ❌ overridden

Internal CSS → green ❌ overridden

Inline CSS → red ✅ applied

Final Output

Text color will be RED

(IMP)

What is DOM (Document Object Model)

DOM is a tree structure of an HTML webpage that allows JavaScript to access and modify elements.

It converts HTML into objects that the browser can understand and control.

Simple Example

HTML

```
<html><body><h1>Hello</h1><p>Paragraph</p></body></html>
```

DOM Structure

```
Document
 └──html
      ├──body
           ├──h1 → "Hello"
           └──p  → "Paragraph"
```

Important Points

- DOM represents HTML as a tree 🌳
- Each HTML element becomes an object
- JavaScript can access and change elements using DOM
- Allows dynamic changes like text, color, style

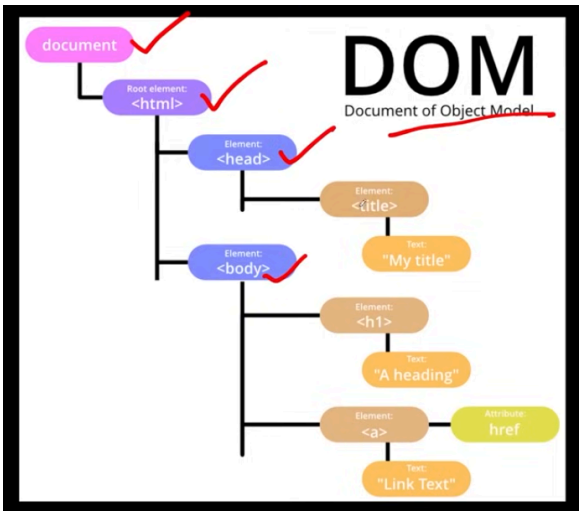
Example using JavaScript

```
<h1id="title">Hello</h1><script>document.getElementById("titl
e").style.color ="red";</script>
```

This changes the text color to red.

One Line Definition ⭐

DOM is the structure of a webpage that allows JavaScript to control HTML elements.



CSS Comment Syntax

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

Example

```
/* Change text color */h1 {color: red;
}
```

Important Points

- Used to add notes 📝
- Not visible on webpage
- Helps in code understanding
- Can be single-line or multi-line

Official CSS Resource (MDN)

MDN (Mozilla Developer Network) is the official and best website to learn CSS.

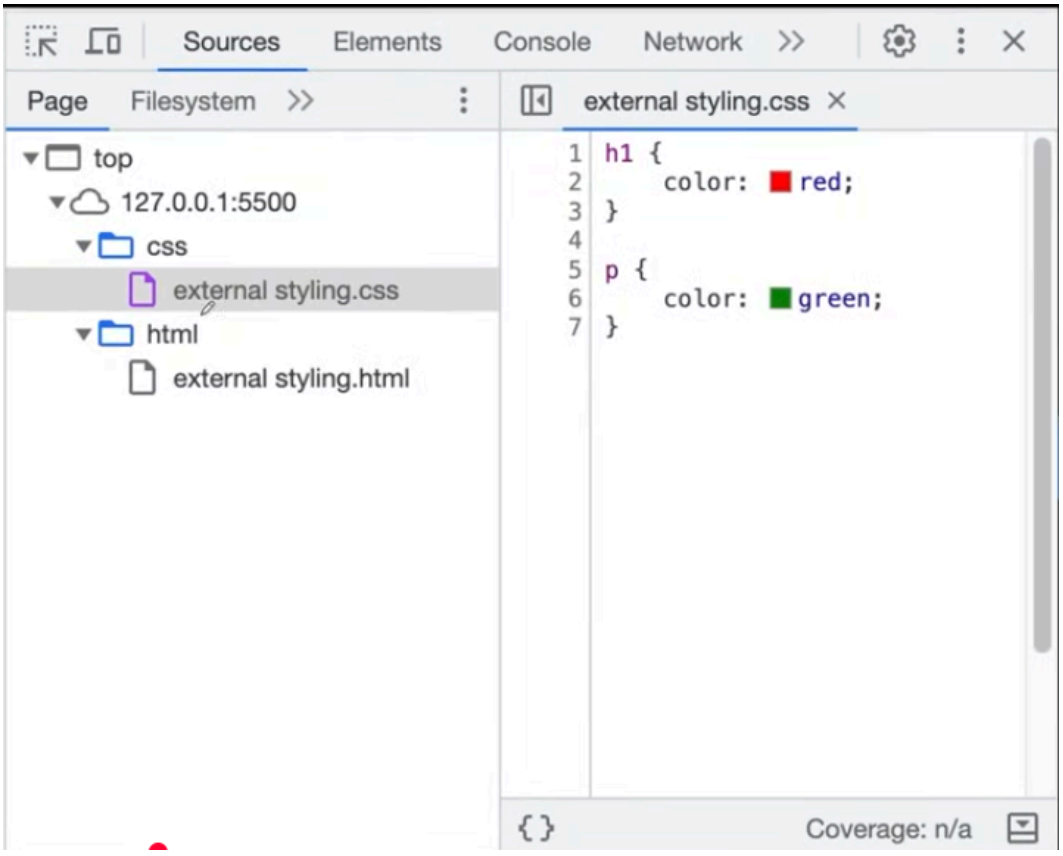
Website Link

https://developer.mozilla.org/

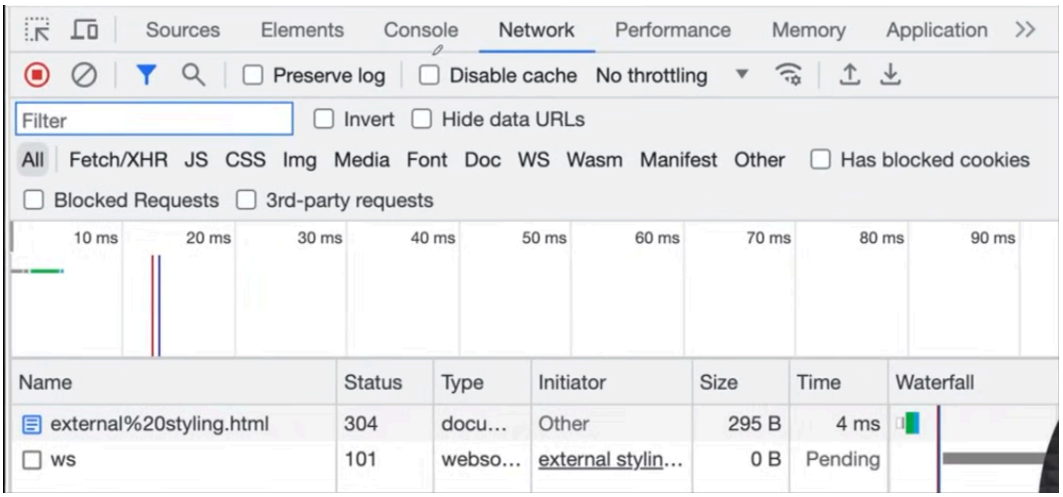
Important Points

- Official documentation for CSS
- Contains real-world examples
- Updated with latest CSS3 features
- Trusted by developers worldwide

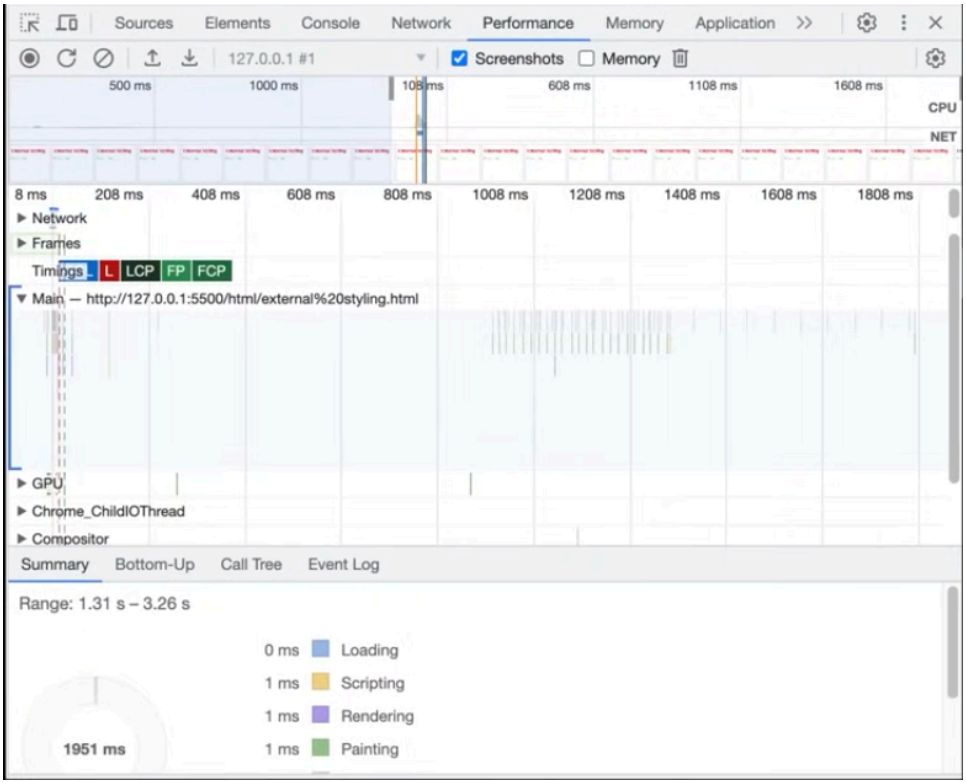
Browser Tools (Source Tab)



Network Tab



Performance Tab



Element Selector in CSS

Element selector is used to select HTML elements using their tag name.

Syntax

```
element {
  property: value;
}
```

Example

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

HTML Example

```
<h1>Hello World</h1><h1>This is heading</h1><p>This is paragr
aph</p>
```

Output

All `<h1>` elements will become red.

Important Points

- Selects elements using tag name
- Applies style to all same elements
- Easy and commonly used

Universal Selector in CSS

Universal selector is used to select all HTML elements on a webpage.

Example

```
* {
color: red;
}
```

Important Points

- Uses asterisk symbol
- Selects all elements
- Applies same style everywhere
- Mostly used for reset styles

ID and Class Selector in CSS

Both are used to select specific HTML elements.

ID Selector

ID is used to select one unique element.

Syntax

```
#idname {
  property: value;
}
```

Example

```
<h1id="title">Hello World</h1>
```

```
#title {color: red;
}
```

Class Selector

Class is used to select multiple elements.

Syntax

```
.classname {
  property: value;
}
```

Example

```
<pclass="text">Paragraph 1</p><pclass="text">Paragraph 2</p>
```

```
.text {color: green;
}
```

Difference Between ID and Class

ID	Class
Used for one element	Used for multiple elements
Uses `#` symbol	Uses `.` symbol
Unique	Reusable

ID is unique, Class is reusable.

ID Selector in CSS

ID selector is used to select one unique HTML element using its ID.

Example

HTML

```
<div id="first">First Div</div>
<div id="second">Second Div</div>
```

CSS

```
#first {
  color: red;
}

#second {
  color: green;
}
```

Important Points

- Uses `#` symbol
- Selects only one unique element
- Cannot be reused

Group Selector in CSS

Group selector is used to select multiple elements and apply the same style.

Syntax

```
selector1, selector2, selector3 {
  property: value;
}
```

Example

```
h1,h2,h3 {
     color: red;
}
```

- **Group selector selects multiple elements using comma.**

Descendant Selector in CSS

Descendant selector is used to select elements inside another element.

Syntax

```
parent child {
  property: value;
}
```

Example

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

HTML Example

```
<div>
  <p>This is inside div</p>
</div>

<p>This is outside div</p>
```

Output

Inside div paragraph → Red

Outside div paragraph → Not affected

Background Color in CSS

Background color is used to set the background color of an HTML element.

Example

```
div {
  background-color: red;
}
```

HTML Example

```
<div>First</div>
<button>Click Me</button>
```

```
div {
  background-color: red;
}
```

```css
button {
  background-color: blue;
}
```

Output

Div background → Red

Button background → Blue

Color System (RGB) in CSS 🎨

CSS uses the RGB color system to create colors.

RGB Meaning 🌈

R = Red 🔴

G = Green 🟢

B = Blue 🔵

Colors are made by mixing these three colors.

Example 💻

```css
h1 {
  color: rgb(255, 0, 0); /* Red 🔴 */
}

p {
  color: rgb(0, 255, 0); /* Green 🟢 */
}

div {
```

```
  color: rgb(0, 0, 255); /* Blue 🔵 */
}
```

Important Points 📌

- Uses Red 🔴, Green 🟢, Blue 🔵
- Values range from 0 to 255 🔢
- Used to create all colors 🎨

One Line Definition ⭐

RGB creates colors using Red 🔴, Green 🟢, and Blue 🔵.

RGB Color Model in CSS 🎨

RGB is used to create colors using Red, Green, and Blue.

Syntax 💻

```
rgb(red, green, blue)
```

Values range from **0 to 255** 🔢

Example 🌈

```
h1 {
  color: rgb(255, 0, 0); /* Red 🔴 */
}

p {
  color: rgb(0, 255, 0); /* Green 🟢 */
}
```

```css
div {
  color: rgb(0, 0, 255); /* Blue 🔵 */
}
```

One Line Definition ⭐

RGB creates colors using rgb(r, g, b) values from 0 to 255.

```html
<!doctype html>
<html lang="en">
  <head>
    <title>RGB Color</title>
  </head>
  <body>
    <div style="background-color: rgb(255, 0, 0)">First</div>
    <div style="background-color: rgb(0, 255, 0)">Second</div
>
    <div style="background-color: rgb(0, 0, 255)">Third</div>
    <div style="background-color: rgb(29, 133, 48)">Fourth</d
iv>
  </body>
</html>
```

HEX Color Model in CSS 🎨

HEX is used to define colors using hexadecimal values.

Syntax 💻

```
#RRGGBB
```

Example 🌈

```css
h1 {
  color: #FF0000; /* Red 🔴 */
}

p {
  color: #00FF00; /* Green 🟢 */
}

div {
  color: #0000FF; /* Blue 🔵 */
}
```

Important Points 📌

- Uses `#` symbol
- Contains 6 characters (0–9, A–F)
- Represents Red, Green, Blue

One Line Definition ⭐

HEX defines colors using #RRGGBB format.

RGBA Color Model (Alpha Channel) 🎨

RGBA is used to add transparency to colors.

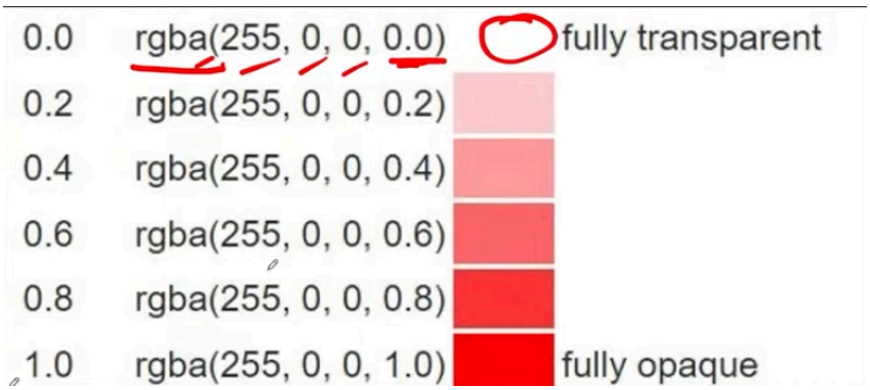
Syntax 💻

```css
rgba(red, green, blue, alpha)
```

Alpha value range: 0 to 1

```css
div {
  background-color: rgba(255, 0, 0, 0.5);
}
```

Red color with 50% transparency.




```html
<!DOCTYPE html>
<html>
<head>
    <title>RGBA Color Example</title>
</head>
<body>

<h1 style="color: rgba(255, 0, 0, 0.1);">First</h1>
<h1 style="color: rgba(255, 0, 0, 0.25);">First</h1>
<h1 style="color: rgba(255, 0, 0, 0.5);">First</h1>
<h1 style="color: rgba(255, 0, 0, 0.75);">First</h1>
<h1 style="color: rgba(255, 0, 0, 1.0);">First</h1>

</body>
</html>
```

Absolute Units (px) in CSS 📏

Pixel (px) is a fixed-size unit used to define size in CSS.

Example 💻

```
h1 {
  font-size: 30px;
}

div {
  width: 200px;
  height: 100px;
}
```

Important Points 📌

- px = pixel (fixed size)
- Used for font, width, height
- Gives exact size control

One Line Definition ⭐

px is a fixed unit used to set exact size in CSS.

Height and Width Property in CSS 📦

Height and Width are used to set the size of an element.

Syntax 💻

Example 🌈

```css
div {
  height: 100px;
  width: 100px;
  background-color: red;
}
```

HTML Example

```html
<div>Box</div>
```

Important Points 📌

- height = element height
- width = element width
- Uses units like px

One Line Definition ⭐

Height and width set the size of an element.

Background Image Property in CSS 🖼️

The **background-image** property is used to add an image as the background of an HTML element.

1. Usage 📌

Adds an image as a background to elements like `div`, `body`, etc.

```css
background-image: url("image-path");
```

2. Syntax 💻

```
    selector {
    background-image:url("path/to/image");
}
```

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

3. Repetition Control 🔁

Controls whether the image repeats or not using **background-repeat**.

```
background-repeat: no-repeat;
```

Options:

```
background-repeat: repeat;
background-repeat: no-repeat
background-repeat: repeat-x;
background-repeat: repeat-y;
```

4. Position Control 📍

Controls the position of the background image using **background-position**.

```
background-position: center;
```

Examples:

```
background-position: top;
background-position: bottom;
background-position: left;
background-position: right;
background-position: center;
```

5. Size Control 📏

Controls the size of the background image using **background-size**.

```
background-size: cover;
```

Examples:

```
background-size: cover;
background-size: contain;
background-size:100px100px;
```

---

6. Background Attachment 📎

Controls whether the image scrolls or stays fixed.

```
background-attachment: fixed;
```

Options:

```
background-attachment: scroll;
background-attachment: fixed;
```

---

7. Shorthand Property ⚡

You can write all background properties in one line.

```
background: color image repeat attachment position;
```

Example:

```
background: redurl("image.jpg") no-repeat fixed center;
```

Visibility Property in CSS 👁️

visibility is used to show or hide an element without removing its space.

Syntax 💻

```
selector {
  visibility: value;
}
```

Values 📌

```
visibility: visible;
visibility: hidden;
```

Example 🌈

```
#box1 {
   visibility: hidden;
}
#box2 {
   visibility: visible;
}
```

Text-Align Property in CSS 📝

text-align is used to align text horizontally inside an element.

Syntax 💻

```
selector {
  text-align: value;
}
```

Values 📌

```
text-align: left;
text-align: right;
text-align: center;
text-align: justify;
```

Example 🌈

```
<!DOCTYPE html>
<html>
<head>
    <title>Text Align Example</title>

    <style>
        #left {
            text-align: left;
        }

        #center {
            text-align: center;
        }

        #right {
```

```
            text-align: right;
        }

        #justify {
            text-align: justify;
        }
    </style>

</head>
<body>

    <p id="left">Left aligned text</p>
    <p id="center">Center aligned text</p>
    <p id="right">Right aligned text</p>
    <p id="justify">Justify aligned text example</p>

</body>
</html>
```

Text-Decoration Property in CSS ✏️

text-decoration is used to add or remove decoration from text.

Values 📌

```
text-decoration: underline;
text-decoration: overline;
text-decoration: line-through;
text-decoration: none;
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Text Decoration</title>
```

```
    <style>
        #box1 {
            text-decoration: underline;
        }

        #box2 {
            text-decoration: overline;
        }

        #box3 {
            text-decoration: line-through;
        }

        a {
            text-decoration: none;
        }
    </style>

</head>
<body>

    <h3 id="box1">Box1</h3>
    <h3 id="box2">Box2</h3>
    <h3 id="box3">Box3</h3>

    <a href="#">Link without underline</a>

</body>
</html>
```

Text-decoration-style

```
<!doctype html>
<html lang="en">
  <head>
    <title>Text Decoration Style</title>

    <style>
      .box {
        text-decoration: underline;
      }

      #box1 {
        text-decoration-style: dashed;
      }

      #box2 {
        text-decoration-style: double;
      }

      #box3 {
        text-decoration-style: solid;
      }

      #box4 {
        text-decoration-style: wavy;
      }
    </style>
  </head>
  <body>
    <h3 id="box1" class="box">Box1</h3>
    <h3 id="box2" class="box">Box2</h3>
    <h3 id="box3" class="box">Box3</h3>
    <h3 id="box4" class="box">Box4</h3>
  </body>
</html>
```

Box1

Box2

Box3

Box4

---

text-decoration-color in CSS 🎨

text-decoration-color is used to change the color of underline, overline, or line-through.

```
<!DOCTYPE html>
<html>
<head>
    <title>Text Decoration Color</title>

    <style>
        #box1 {
            text-decoration: underline;
            text-decoration-color: red;
        }

        #box2 {
            text-decoration: underline;
            text-decoration-color: blue;
        }

        #box3 {
            text-decoration: line-through;
            text-decoration-color: green;
```

```
        }
    </style>

</head>
<body>

    <h2 id="box1">Red Underline</h2>
    <h2 id="box2">Blue Underline</h2>
    <h2 id="box3">Green Line Through</h2>

</body>
</html>
```

Text-Transform Property in CSS 🔤

text-transform is used to change the case of text (uppercase, lowercase, etc.).

```
selector {
  text-transform: value;
}
```

Values 📌

```
text-transform: uppercase;
text-transform: lowercase;
text-transform: capitalize;
text-transform: none;
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Text Transform</title>
```

```
    <style>
        #upper {
            text-transform: uppercase;
        }

        #lower {
            text-transform: lowercase;
        }

        #cap {
            text-transform: capitalize;
        }
    </style>

</head>
<body>

    <p id="upper">hello world</p>
    <p id="lower">HELLO WORLD</p>
    <p id="cap">hello world</p>

</body>
</html>
```

Line-Height Property in CSS 📏

line-height is used to control the space between lines of text.

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

    <title>Line Height</title>
```

```
    <style>
        #first {
            line-height: 6px;
        }

        #second {
            line-height: 30px;
        }
    </style>

</head>
<body>

    <p id="first">
        Lorem ipsum dolor sit amet consectetur adipisicing el
it.
        Numquam facere maxime autem, culpa
    </p>

    <hr>

    <p id="second">
        Lorem ipsum dolor sit amet consectetur adipisicing el
it.
        Numquam facere maxime autem, culpa
    </p>

</body>
</html>
```

Lorem ipsum dolor sit amet consectetur adipisicing elit. Numquam facere maxime autem, culpa

---

Lorem ipsum dolor sit amet consectetur adipisicing elit. Numquam facere maxime autem, culpa

Font-Style Property in CSS 🔤

font-style is used to change the style of text (italic, oblique, normal).

Values 📌

```
font-style: normal;
font-style: italic;
font-style: oblique;
```

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

    <title>Font Style</title>

    <style>

        #first {
            font-style: italic;
        }

        #second {
            font-style: oblique;
        }
```

```
        #third {
            font-style: normal;
        }

    </style>

</head>
<body>

    <p id="first">Lorem ipsum dolor sit amet</p>
    <p id="second">Lorem ipsum dolor sit amet</p>
    <p id="third">Lorem ipsum dolor sit amet</p>

</body>
</html>
```

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet

Font-Family Property in CSS 🔤

font-family is used to change the font of text.

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

#second {
```

```
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-seri
f;
}

#third {
  font-family: 'Oswald', sans-serif;
}
```

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

    <title>Font Family</title>

    <!-- Google Font -->
    <link rel="stylesheet" href="https://fonts.googleapis.co
m/css2?family=Oswald&display=swap">

    <style>

        #first {
            font-family: Arial, Helvetica, sans-serif;
        }

        #second {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;
        }

        #third {
            font-family: 'Oswald', sans-serif;
        }

    </style>
```

```
</head>
<body>

    <p id="first">Lorem ipsum dolor sit amet</p>

    <p id="second">Lorem ipsum dolor sit amet</p>

    <p id="third">Lorem ipsum dolor sit amet</p>

</body>
</html>
```

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet

Lorem ipsum dolor sit amet

Icons using Font Awesome in HTML ⭐

Font Awesome is used to add icons using fonts.

Step 1: Add Font Awesome Script in `<head>` 💻

```
<script src="https://kit.fontawesome.com/43290fa92d.js" cross
origin="anonymous"></script>
```

Step 2: Use Icons in `<body>` 🎨

```
<i class="fa-solid fa-house"></i>
<i class="fa-brands fa-facebook"></i>
<i class="fa-brands fa-tiktok"></i>
<i class="fa-solid fa-poo" style="font-size: 40px;"></i>
```

Google Font Icons

https://fonts.google.com/icons

CSS Box Model 📦

Box Model is used to define the layout and spacing of elements in CSS.

Every element is a box made of 4 parts:

---

1. Content 📝

The actual text, image, or element.

```
width:200px;
height:100px;
```

---

2. Padding 📏

Space between content and border.

```
padding:10px;
```

---

3. Border 🟦

Outline around padding and content.

```
border:2px solid black;
```

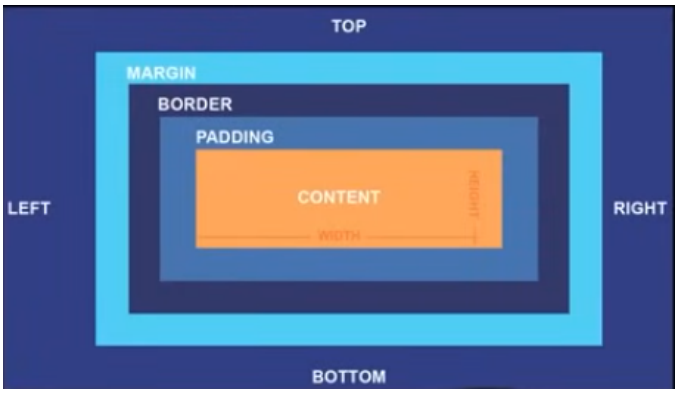
4. Margin 📐

Space outside the border.

```
margin:20px;
```

Complete Example 💻

```
<!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>Hello Box Model</div>

</body>
</html>
```



Box-Sizing Property 📦

box-sizing controls how width and height are calculated.

Values 📌

```
box-sizing: content-box;box-sizing: border-box;
```

Difference ⭐

content-box (default)

Width = content only

Padding and border added extra

border-box

Width includes content + padding + border

Example 🌈

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

```
<style>

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

</style>
</head>
<body>

<div>Box Sizing Example</div>

</body>
</html>
```

Display Property in CSS 📦

display property defines how an element appears on the webpage.

Display Values 📌

```
display: block;
display: inline;
display: inline-block;
display: none;
```

1. Block Elements 🟦

Block elements start on a new line and take full width.

Features

- Starts on new line

- Takes full width
- Can set width and height

Examples

```
<div>Block</div>
<p>Block</p>
<h1>Block</h1>
```

2. Inline Elements 🟩

Inline elements stay in the same line.

Features

- No new line

- Takes only required width
- Width and height cannot be set

Examples

```
<span>Inline</span>
<a href="#">Link</a>
```

3. Inline-Block 🟨

Inline-block stays inline but allows width and height.

```
div {
        display: inline-block;
```

```
        width:100px;
        height:100px;
}
```

4. Display None ❌

Hides the element completely.

```
div {
    display: none;
}
```

Display: Inline-Block in CSS 🟨

inline-block allows elements to stay in the same line and also set width and height.

It combines features of **inline + block**.

Example 🌈

```
div {
    display: inline-block;
    width:100px;
    height:100px;
    background-color: red;
}
```

Difference between `display: none` and `visibility: hidden` 🔍

Both hide elements, but behave very differently.

---

1. display: none ❌

Completely removes the element from the page.

Features

- Element is hidden
- Does NOT occupy space
- Layout changes

Example

```
#box {

    display: none;
}
```

Result: Element disappears completely.

---

2. visibility: hidden 👻

Hides the element but keeps its space.

Features

- Element is hidden
- Space is still occupied
- Layout remains same

Example

```
#box {
    visibility: hidden;
```

```
}
```

Result: Element invisible but space remains.

Responsive Website 📱💻

Responsive website adjusts layout based on screen size.

Important Points 📌

- Adapts to different screen sizes 📱
- Uses flexible layouts 📦
- Optimizes images and content 🖼️
- Works on mobile and desktop 💻

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

<meta name="viewport" content="width=device-width, initial-sc
ale=1.0">

<style>

div {
  width: 100%;
  background: red;
}

@media (max-width: 600px) {
  div {
    background: blue;
  }
}
```

```
</style>

</head>
<body>

<div>Responsive Box</div>

</body>
</html>
```

Relative Units in CSS 📏

Relative units change size based on parent, root, or screen size.

---

1. Percentage (%) 📊

Relative to parent element size.

```
div {width:50%;
}
```

---

2. em 🔤

Relative to parent font size.

```
p {font-size:2em;
}
```

---

3. rem 🔠

Relative to root (`html`) font size.

```
p {font-size:2rem;
}
```

4. vh 📱

Relative to viewport height.

```
div {height:50vh;
}
```

5. vw 💻

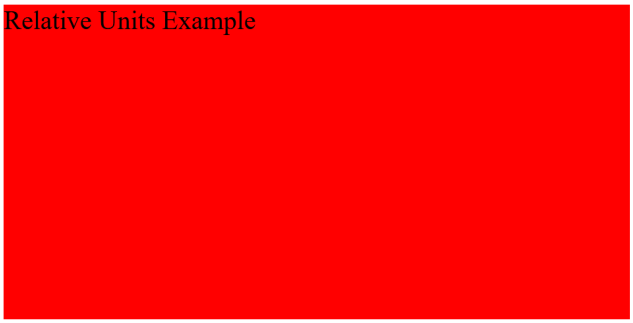
Relative to viewport width.

```
div {width:50vw;
}
```

Complete Example 💻

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

div {
  width: 50%;
  height: 50vh;
  font-size: 2rem;
  background: red;
}

</style>
```

```
</head>
<body>

<div>Relative Units Example</div>

</body>
</html>
```



Relative Units (Percentage %) in CSS 📊

Percentage (%) is used to set size relative to the parent element.

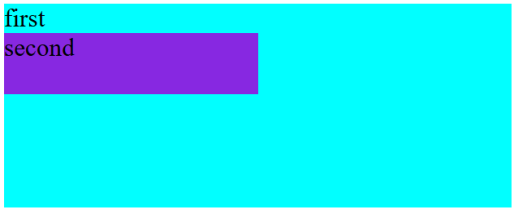
Syntax 💻

```
selector {
    width:50%;
    height:30%;
}
```

Complete Code Example 🌈

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

```
<head>

<style>

#first {
  height: 200px;
  width: 200px;
  background-color: aqua;
  font-size: 25px;
}

#second {
  background-color: blueviolet;
  width: 50%;
  height: 30%;
}

</style>

</head>
<body>

<div id="first">
  first

  <div id="second">
    second
  </div>

</div>

</body>
</html>
```

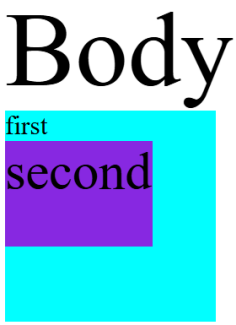


Relative Unit (em) in CSS 🔤

em is relative to the font size of the parent element.

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

<style>

body {
  font-size: 100px;
}

#first {
  height: 200px;
  width: 200px;
  background-color: aqua;
  font-size: 25px;
}

#second {
  background-color: blueviolet;
  width: 70%;
  height: 50%;
  font-size: 2em;
```

```
}

</style>

</head>
<body>

Body

<div id="first">
  first

  <div id="second">
    second
  </div>

</div>

</body>
</html>
```



Explanation 📌

- Parent font-size = 25px

- Child font-size = 2em

👉 2em = 2 × 25px = **50px**

Important Points ⭐

- Relative to parent font size
- Used for scalable design
- Helps in responsive layout

One Line Definition ⭐

em is relative to parent font size.

Relative Units (vw / vh) in CSS 📱💻

vw and vh are relative to the viewport (screen size).

Definitions 📌

- **1vw** = 1% of screen width 💻
- **1vh** = 1% of screen height 📱

Example 💻

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

<style>

#first {
  height: 50vh;
  width: 90vw;
  background-color: red;
```

```
}

</style>

</head>
<body>

<div id="first"></div>

</body>
</html>
```

Explanation ⭐

If screen size = 1000px width and 800px height

- 90vw = 900px width
- 50vh = 400px height

Important Points 🚀

- Relative to screen size
- Used for responsive design
- Automatically adjusts

One Line Definition ⭐

vw and vh are relative to viewport width and height.

Position Property in CSS 📍

Position property controls where an element appears on the page.

Types of Position 🚀

1. Static (default) 📄

- Default position

- Follows normal page flow
- `top, right, bottom, left` do NOT work

```
div {
    position: static;
}
```

2. Relative 🔄

- Moves relative to its normal position

- Original space is preserved

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

3. Absolute 🎯

- Positioned relative to nearest positioned parent

- Removed from normal flow

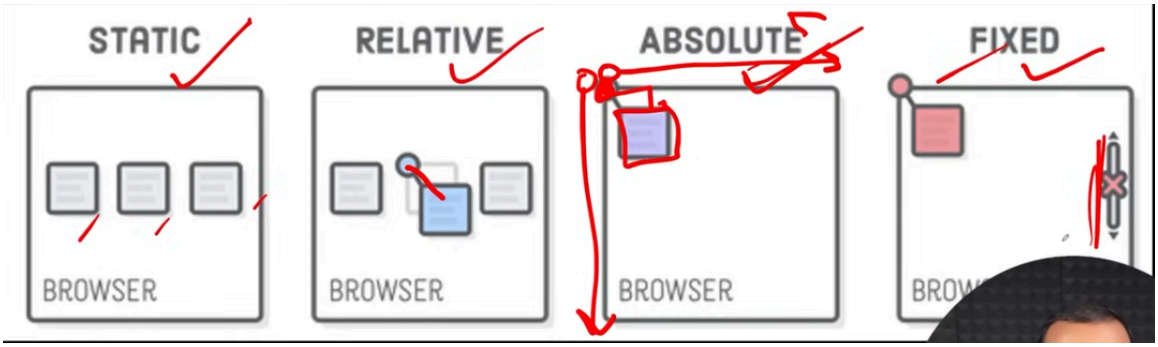
```
div {
        position: absolute;
        top:0;
        right:0;
}
```

4. Fixed 📌

- Positioned relative to viewport (screen)

• Does NOT move on scroll

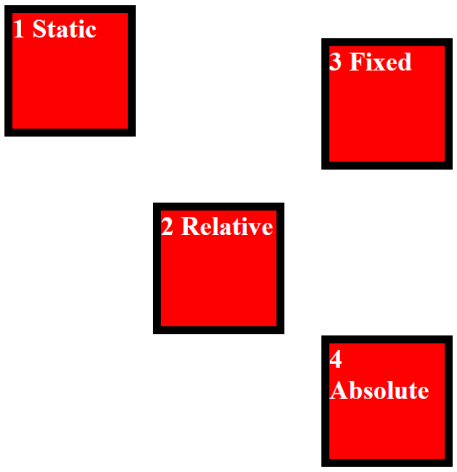
```
div {
        position: fixed;
        bottom:10px;
        right:10px;
}
```




```
<!doctype html>
<html lang="en">
  <head>
    <title>Position Property Example</title>

    <style>
      /* Common style for all divs */
      div {
        height: 70px;
        width: 70px;
        background-color: red;
        border: 5px solid black;
        margin: 20px;
        color: white;
        font-weight: bold;
```

```
      }

      /* Static (default) */
      #div1 {
        position: static;
      }

      /* Relative */
      #div2 {
        position: relative;
        top: 20px;
        left: 90px;
      }

      /* Fixed */
      #div3 {
        position: fixed;
        top: 20px;
        left: 200px;
      }

      /* Absolute */
      #div4 {
        position: absolute;
        top: 200px;
        left: 200px;
      }
    </style>
  </head>

  <body>
    <div id="div1">1 Static</div>

    <div id="div2">2 Relative</div>

    <div id="div3">3 Fixed</div>
```

```
    <div id="div4">4 Absolute</div>
  </body>
</html>
```



z-index Property in CSS 🔢

z-index controls which element appears on top when elements overlap.

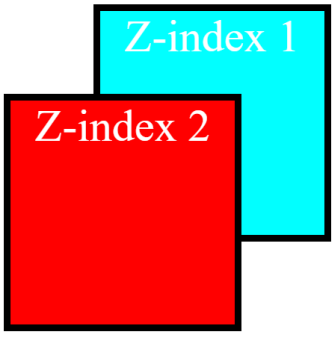
Important Points 📌

- Works only with `position: relative, absolute, fixed, sticky`
- Higher value = appears on top ⬆️
- Lower value = appears behind ⬇️
- Default value = 0

Example 💻

```
<!DOCTYPE html>
<html>

<head>

<title>Z-Index Example</title>

<style>

.container {
  position: relative;
}

/* Common box style */
.box {
  position: absolute;
  width: 100px;
  height: 100px;
  border: 3px solid black;
  text-align: center;
  font-size: 20px;
  color: white;
}

/* Box 1 */
.box1 {
  background-color: red;
  left: 20px;
  top: 60px;
  z-index: 2;
}

/* Box 2 */
.box2 {
  background-color: aqua;
```

```
  left: 60px;
  top: 20px;
  z-index: 1;
}

</style>

</head>

<body>

<div class="container">

  <div class="box box1">Z-index 2</div>

  <div class="box box2">Z-index 1</div>

</div>

</body>

</html>
```



Result 🧠

- Red box (z-index: 2) → on top
- Aqua box (z-index: 1) → behind

One Line Definition ⭐

Higher z-index value means the element appears above others.

Float Property in CSS 🌊

Float is used to move elements to the left or right inside a container.

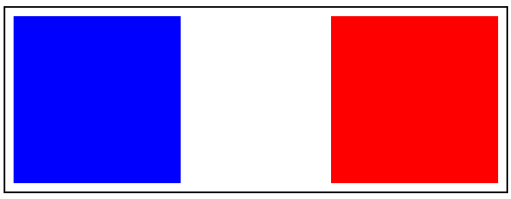
Important Points 📌

- Align element left or right
- Values: `left`, `right`, `none`
- Used for layout alignment
- Old technique (Flexbox preferred now)

Example 💻

```
<!doctype html>
<html>
  <head>
    <title>Float Example</title>

    <style>
      .container {
        height: 110px;
        width: 300px;
        border: 1px solid black;
      }

      /* Common box style */
      .box {
```

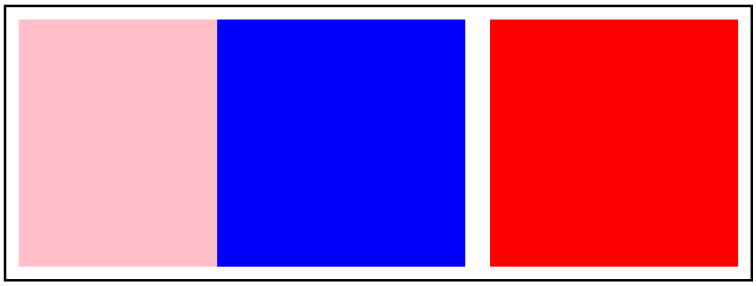
```
        width: 100px;
        height: 100px;
        margin: 5px;
      }

      /* Float Right */
      .box1 {
        background-color: red;
        float: right;
      }

      /* Float Left */
      .box2 {
        background-color: blue;
        float: left;
      }
    </style>
  </head>

  <body>
    <div class="container">
      <div class="box box1"></div>

      <div class="box box2"></div>
    </div>
  </body>
</html>
```



float : none

```
<!doctype html>
<html>
  <head>
    <title>Float Example</title>

    <style>
      .container {
        height: 110px;
        width: 300px;
        border: 1px solid black;
      }

      /* Common box style */
      .box {
        width: 100px;
        height: 100px;
        margin: 5px;
      }

      /* Float Right */
      .box1 {
        background-color: red;
        float: right;
      }

      /* Float Left */
      .box2 {
        background-color: blue;
        float:right;
      }
        .box3 {
        background-color: pink;
        float: none;
      }
```

```
    </style>
  </head>

  <body>
    <div class="container">
      <div class="box box1"></div>

      <div class="box box2"></div>
      <div class="box box3"></div>
    </div>
  </body>
</html>
```



What is Flexbox? 📦➡️📦

Flexbox is a layout system used to arrange items in a row or column.

Important Points 📌

- One-dimensional layout (row or column)
- Items can expand or shrink
- Makes layouts flexible and responsive
- Uses a container and items

Basic Example 💻

```
<!DOCTYPE html>
<html>

<head>

<title>Flexbox Example</title>

<style>

.container {
  display: flex;
  background-color: lightgray;
}

.item {
  background-color: pink;
  margin: 10px;
  padding: 20px;
}

</style>

</head>

<body>

<div class="container">

  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item">Item 3</div>
  <div class="item">Item 4</div>
```

```
</div>

</body>

</html>
```

Item 1 Item 2 Item 3 Item 4

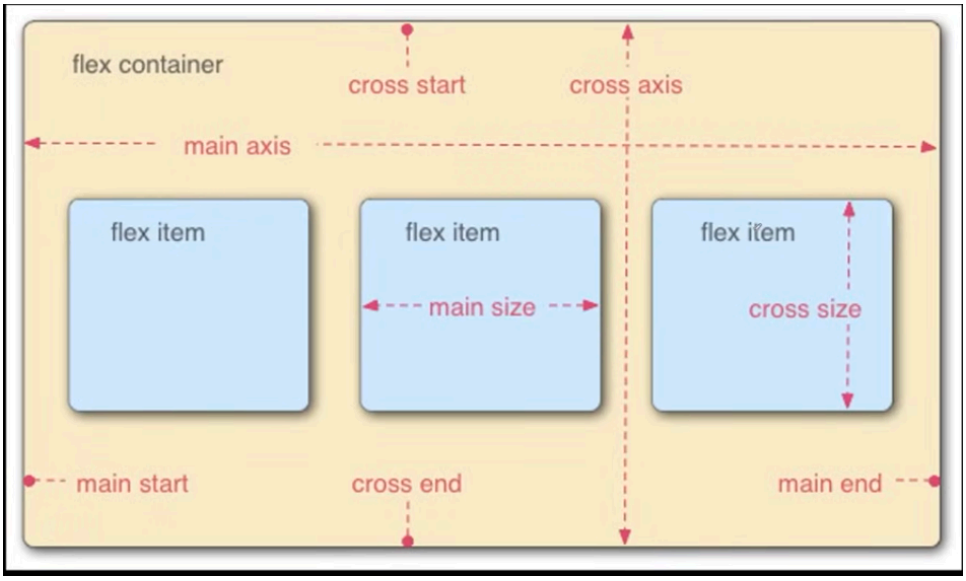
Result 🧠

Items will appear in a row automatically.

One Line Definition ⭐

Flexbox is used to arrange elements in rows or columns easily and flexibly.

Flex Model



Flexbox Direction (flex-direction) 📦➡️⬇️

flex-direction defines the direction of flex items inside a flex container.

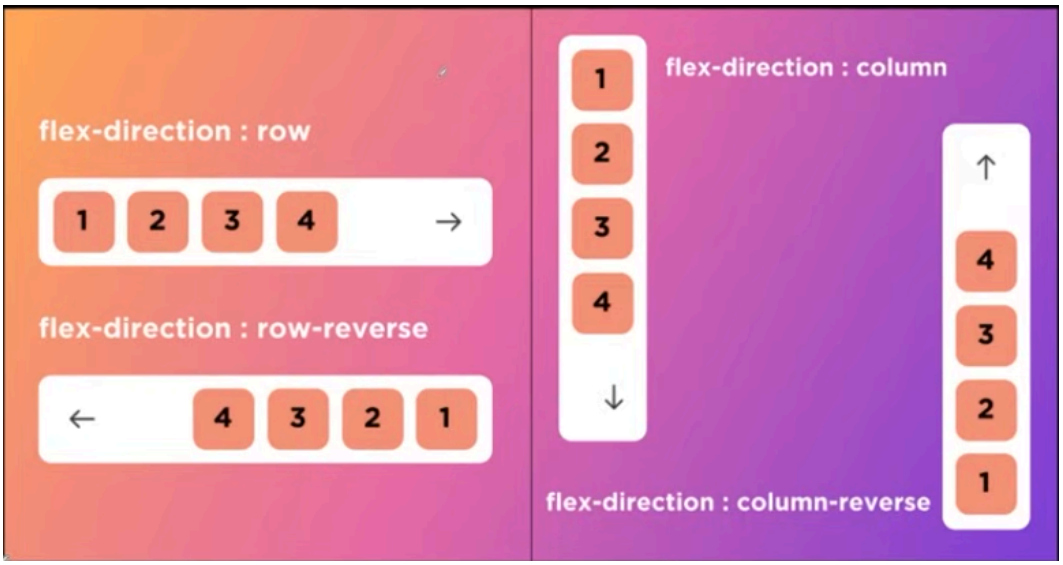
Values 📌

- `row` → horizontal (left → right) ➡️
- `row-reverse` → horizontal (right → left) ⬅️
- `column` → vertical (top → bottom) ⬇️
- `column-reverse` → vertical (bottom → top) ⬆️

Example 💻

```
<!DOCTYPE html>
<html>

<head>

<title>Flex Direction Example</title>

<style>

.container {
  display: flex;
  flex-direction: row; /* change to column, row-reverse, colu
mn-reverse */
  background-color: lightgray;
  padding: 10px;
}

.item {
  background-color: orange;
```

```
  margin: 5px;
  padding: 20px;
  font-size: 20px;
}

</style>

</head>

<body>

<div class="container">

  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>

</div>

</body>

</html>
```

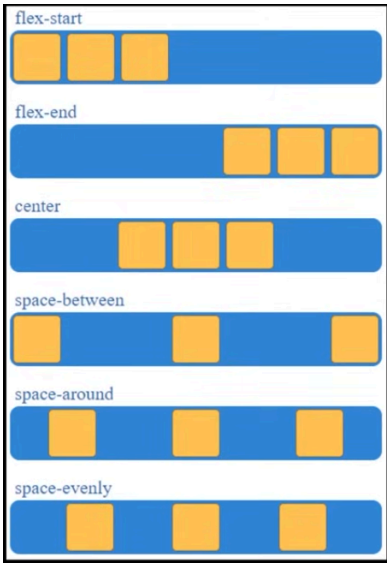


Justify Content (Flexbox) 📦↔

justify-content aligns flex items horizontally (main axis).

Values 📌

- `flex-start` → items at start ⬅️
- `flex-end` → items at end ➡️
- `center` → items at center 🎯
- `space-between` → space between items ↔
- `space-around` → space around items 🔄
- `space-evenly` → equal space everywhere ⚖️



Example 💻

```
<!DOCTYPE html>
<html>

<head>

<title>Justify Content Example</title>

<style>

.container {
  display: flex;
  justify-content: center; /* change value here */
  background-color: lightgray;
  padding: 10px;
}

.item {
  background-color: orange;
  margin: 5px;
  padding: 20px;
}
```

```
</style>

</head>

<body>

<div class="container">

  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>

</div>

</body>

</html>
```

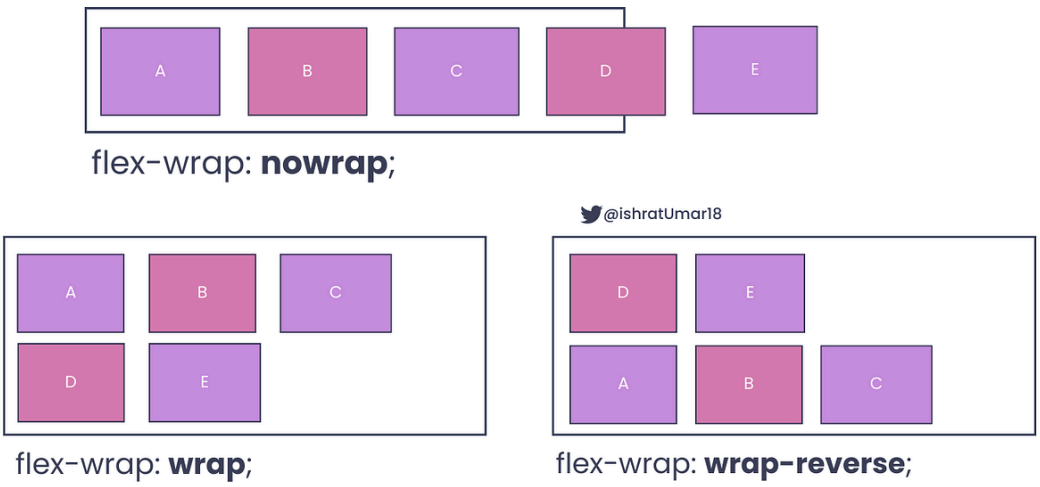
Result 🧠

- Controls horizontal alignment
- Moves items left, right, or center

One Line Definition ⭐

justify-content aligns flex items horizontally inside a flex container.

Flex Container (Flex-wrap)



Align Items (Flexbox) 📦↕

align-items aligns flex items vertically (cross-axis).

Values 📌

- `flex-start` → top ⬆️
- `flex-end` → bottom ⬇️
- `center` → center 🎯
- `stretch` → fill full height 📏
- `baseline` → align by text baseline 📝

Example 💻

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

```
<head>

<title>Align Items Example</title>

<style>

.container {
  display: flex;
  height: 200px;
  background-color: lightgray;

  justify-content: center; /* horizontal */
  align-items: center;     /* vertical */
}

.item {
  background-color: orange;
  margin: 5px;
  padding: 20px;
}

</style>

</head>

<body>

<div class="container">

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

</div>

</body>
```

```
</html>
```

Result 🧠

- Items move vertically
- Center, top, or bottom alignment possible

One Line Definition ⭐

align-items aligns flex items vertically inside a flex container.

Align Content (Flexbox) 📦📏

align-content controls the spacing between flex lines inside a flex container.

👉 Works only when:

- `display: flex`
- `flex-wrap: wrap`
- Multiple rows exist

Values 📌

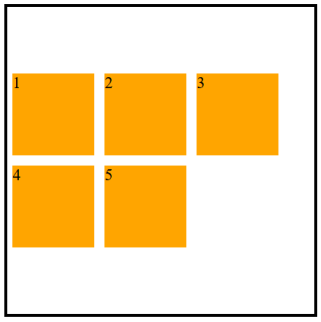
- `flex-start` → lines at top ⬆️
- `flex-end` → lines at bottom ⬇️
- `center` → lines at center 🎯
- `space-between` → equal space between lines 📏

- `space-around` → space around lines
- `space-evenly` → equal space everywhere

Example 💻

```
<!DOCTYPE html>
<html>

<head>

<title>Align Content Example</title>

<style>

.container {
  display: flex;
  flex-wrap: wrap;
  align-content: center;

  height: 300px;
  width: 300px;
  border: 3px solid black;
}

.item {
  width: 80px;
  height: 80px;
  margin: 5px;
  background-color: orange;
}

</style>

</head>
```

```
<body>

<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>

</body>

</html>
```



Align Self (Flex Item Property) 🎯

align-self allows a single flex item to override the container's align-items property.

👉 It aligns one specific item on the cross-axis (vertical axis in row direction).

Why use align-self? 🧠

Normally:

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

All items go to top ⬆️

But with `align-self`, one item can move differently.

Values 📌

- `auto` → default (uses container value)
- `flex-start` → top ⬆️
- `flex-end` → bottom ⬇️
- `center` → center 🎯
- `stretch` → fill height

Example 💻

```
<!DOCTYPE html>
<html>

<head>

<title>Align Self Example</title>

<style>

.container {
  display: flex;
  height: 200px;
  border: 3px solid black;
```

```
  align-items: flex-start;
}

.item {
  width: 80px;
  height: 80px;
  margin: 10px;
  background-color: skyblue;
}

/* Override only this item */
.special {
  background-color: orange;
  align-self: center;
}

</style>

</head>

<body>

<div class="container">

  <div class="item">Item 1</div>
  <div class="item">Item 2</div>
  <div class="item special">Item 3</div>
  <div class="item">Item 4</div>

</div>

</body>

</html>
```

Flex Shrink Property 📦⬇️

flex-shrink controls how much a flex item will shrink when there is not enough space in the flex container.

Simple Definition 🧠

When container size is small, items shrink.

flex-shrink decides which item shrinks more or less.

Syntax 📌

```
flex-shrink: number;
```

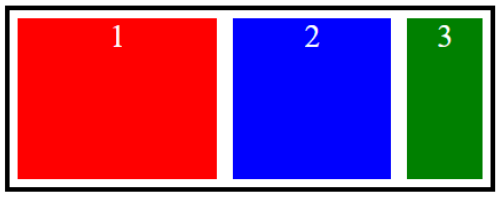
- Default value → `1`
- Higher value → Shrinks more
- Lower value → Shrinks less
- `0` → Will NOT shrink

Example 💻

```
<!DOCTYPE html>
<html>

<head>

<title>Flex Shrink Example</title>
```

```
<style>

.container {
  display: flex;
  width: 300px;
  border: 3px solid black;
}

.box {
  width: 150px;
  height: 100px;
  margin: 5px;
  color: white;
  font-size: 20px;
  text-align: center;
}

.box1 {
  background-color: red;
  flex-shrink: 1;
}

.box2 {
  background-color: blue;
  flex-shrink: 2;
}

.box3 {
  background-color: green;
  flex-shrink: 4;
}

</style>

</head>
```

```
<body>

<div class="container">

  <div class="box box1">1</div>
  <div class="box box2">2</div>
  <div class="box box3">3</div>

</div>

</body>

</html>
```



One-Line Interview Definition 🎯

flex-shrink defines how much a flex item shrinks relative to other items when space is limited.

What happens here ⚙️

Container width = 300px

Total boxes width = 450px

So shrinking happens:

- Box1 → shrinks normal
- Box2 → shrinks more
- Box3 → shrinks MOST ⭐

Because:

```
flex-shrink:4;
```

Visual Logic 📊

Value	Shrink Amount
0	No shrink
1	Normal shrink
2	More shrink
4	Maximum shrink

Flex Grow Property 📦⬆️

flex-grow controls how much a flex item will grow when extra space is available in the flex container.

Simple Definition 🧠

When container has extra space, items expand.

flex-grow decides which item grows more.

Syntax 📌

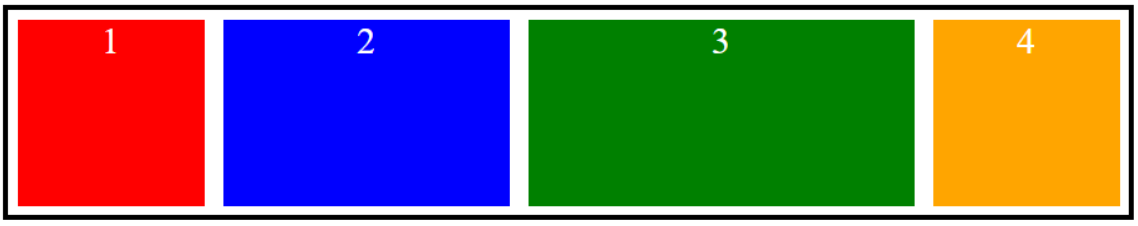
```
flex-grow: number;
```

- Default value → 0
- Higher value → Grows more
- Lower value → Grows less
- 0 → Will NOT grow

Example 💻

```
<!DOCTYPE html>
<html>

<head>

<title>Flex Grow Example</title>

<style>

.container {
  display: flex;
  width: 600px;
  border: 3px solid black;
}

.box {
  width: 100px;
  height: 100px;
  margin: 5px;
  color: white;
  font-size: 20px;
  text-align: center;
}

.box1 {
  background-color: red;
  flex-grow: 0;
}

.box2 {
  background-color: blue;
  flex-grow: 1;
}

.box3 {
```

```
  background-color: green;
  flex-grow: 2;
}

.box4 {
  background-color: orange;
  flex-grow: 0;
}

</style>

</head>

<body>

<div class="container">

  <div class="box box1">1</div>
  <div class="box box2">2</div>
  <div class="box box3">3</div>
  <div class="box box4">4</div>

</div>

</body>

</html>
```



Visual Logic 📊

Value	Growth
0	No grow
1	Normal grow
2	Double grow
3	Triple grow

Flexbox Property: Order 🔢

order property controls the display order of flex items inside a flex container.

It rearranges items visually without changing the HTML structure.

Default Behavior 📌

By default, all flex items have:

```
order: 0;
```

Items appear in the same order as written in HTML.

Syntax 🧾

- Lower number → appears first
- Higher number → appears later
- Negative values allowed ✅

```
order: number;
```

Example (One File) 💻

```
<!doctype html>
<html>
  <head>
    <title>Flex Order Example</title>

    <style>
      .container {
        display: flex;
        border: 3px solid black;
      }

      .box {
        width: 100px;
        height: 100px;
        margin: 10px;
        font-size: 20px;
        color: white;
        text-align: center;
        line-height: 100px;
      }

      .box1 {
        background-color: aqua;
        order: 3;
      }

      .box2 {
        background-color: blueviolet;
        order: 1;
      }

      .box3 {
        background-color: yellow;
        color: black;
        order: 4;
```

```
      }

      .box4 {
        background-color: tomato;
        order: 2;
      }
    </style>
  </head>

  <body>
    <div class="container">
      <div class="box box1">Item 1</div>
      <div class="box box2">Item 2</div>
      <div class="box box3">Item 3</div>
      <div class="box box4">Item 4</div>
    </div>
  </body>
</html>
```

Output Order 🧠

Even though HTML order is:

```
Item1,Item2,Item3,Item4
```

Display order becomes:

```
Item2 →Item4 →Item1 →Item3
```

Because:

Item	order value
Item 2	1
Item 4	2
Item 1	3
Item 3	4

Visual Logic 📊

order value	Position
-1	First
0	Default
1	After 0
2	After 1

Important Interview Point 🎯

order changes visual order, NOT HTML structure.

DOM remains same for:

- SEO
- Screen readers
- Accessibility

Short Definition 🧠

order property defines the sequence in which flex items appear inside a flex container.

Flexbox vs Grid 🧩⚡

Both are CSS layout systems, but they solve different layout problems.

Flexbox 📦 (One-Dimensional)

Flexbox works in **one direction only**:

- Row (horizontal)

OR

• Column (vertical)

Not both at the same time.

Think of Flexbox like arranging books on a single shelf 📚

Example

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

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

.box {
  width: 100px;
  height: 100px;
}

.box1 { background: red; }
.box2 { background: green; }
.box3 { background: orange; }

</style>
</head>
<body>
```

```
<div class="container">
  <div class="box box1"></div>
  <div class="box box2"></div>
  <div class="box box3"></div>
</div>

</body>
</html>
```

Grid 🧱 (Two-Dimensional)

Grid works in **rows AND columns together**

Think of Grid like a chessboard ♟️

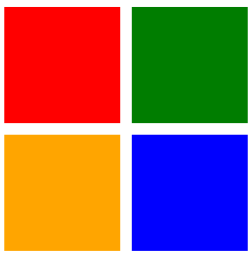
You control:

- Rows
- Columns
- Both together

Example

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

```
.container {
  display: grid;
  grid-template-columns: 100px 100px;
  grid-template-rows: 100px 100px;
  gap: 10px;
}

.box1 { background: red; }
.box2 { background: green; }
.box3 { background: orange; }
.box4 { background: blue; }

</style>
</head>
<body>

<div class="container">
  <div class="box1"></div>
  <div class="box2"></div>
  <div class="box3"></div>
  <div class="box4"></div>
</div>

</body>
</html>
```



CSS Grid 📊 (2D Layout System)

Grid is used to create layouts using **rows and columns together**.

Flexbox = 1D (row OR column)

Grid = 2D (row AND column)

Key Points 🎯

- Grid is a **2D layout system**
- Activate using → `display: grid;`
- Parent = Grid Container
- Children = Grid Items
- Layout defined using → `grid-template-columns` and `grid-template-rows`
- Each box is called a **Grid Cell**

Complete Example Code ✅

```
<!DOCTYPE html>
<html>
<head>
<title>CSS Grid Example</title>

<style>

.container {
  display: grid;

  /* Define 2 columns */
  grid-template-columns: 100px 100px;

  /* Define 2 rows */
  grid-template-rows: 100px 100px;
```

```
  gap: 10px;
}

/* Item 1 */
.item1 {
  grid-column: 1 / 2;
  grid-row: 1 / 2;
  background-color: lightblue;
}

/* Item 2 */
.item2 {
  grid-column: 2 / 3;
  grid-row: 1 / 2;
  background-color: lightgreen;
}

/* Item 3 */
.item3 {
  grid-column: 1 / 2;
  grid-row: 2 / 3;
  background-color: lightpink;
}

/* Item 4 */
.item4 {
  grid-column: 2 / 3;
  grid-row: 2 / 3;
  background-color: lightyellow;
}

.box {
  display: flex;
  align-items: center;
  justify-content: center;
  font-weight: bold;
```

```
}

</style>

</head>
<body>

<div class="container">

  <div class="item1 box">Item 1</div>
  <div class="item2 box">Item 2</div>
  <div class="item3 box">Item 3</div>
  <div class="item4 box">Item 4</div>

</div>

</body>
</html>
```

Media Queries 📱💻

Media Queries are used to make websites **responsive** for different screen sizes.

Key Points 🎯

- Used to apply CSS based on **screen size**

- Helps create **responsive websites**
- Commonly used for **mobile, tablet, desktop**
- Syntax:

```
@media (condition) {
  CSS rules
}
```

Most Common Example

```
@media screenand (max-width:768px) {
            body {
            background-color: lightblue;
  }
}
```

Meaning:

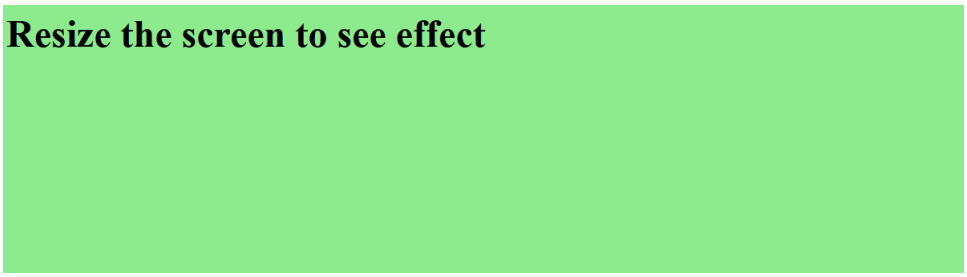
Apply styles when screen width is

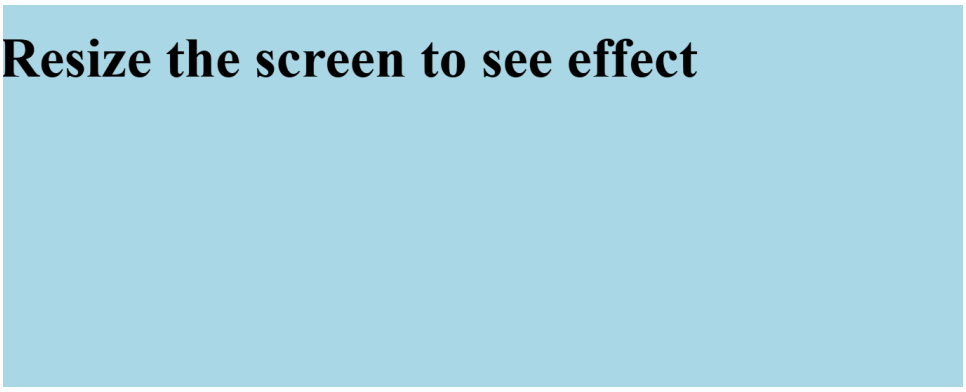
768px or less

Complete Example (Single File) ✅

```
<!DOCTYPE html>
<html>
<head>
<title>Media Query Example</title>

<style>

body {
  background-color: lightgreen;
```

```
}

/* Mobile */
@media screen and (max-width: 768px) {
  body {
    background-color: lightblue;
  }
}

/* Small Mobile */
@media screen and (max-width: 480px) {
  body {
    background-color: lightcoral;
  }
}

</style>

</head>
<body>

<h1>Resize the screen to see effect</h1>

</body>
</html>
```





Interview One-Line Answer 🎤

Media Queries are used to apply CSS based on screen size to create responsive websites.

Pseudo Classes 🎭 (Short & Important)

Pseudo-classes are used to define **special states of elements** like hover, click, first item, etc.

Syntax

```
selector:pseudo-class {
  CSS styles;
}
```

Example:

```css
button:hover {
  background-color: red;
}
```

Common Pseudo Classes

Pseudo Class	Meaning
`:hover`	When mouse is over element
`:active`	When element is clicked
`:first-child`	First child element
`:last-child`	Last child element
`:focus`	When input is focused

Complete Example ✅

```html
<!doctype html>
<html>
  <head>
    <title>Pseudo Class Example</title>

    <style>
      .btn {
        height: 40px;
        width: 120px;
        border: 2px solid blue;
        border-radius: 5px;
        background-color: lightblue;
        cursor: pointer;
      }

      /* Hover state */
      .btn:hover {
```

```
        border: 2px solid red;
        background-color: lightgreen;
      }

      /* Active state (click) */  When a User Clicks on it
      .btn:active {
        background-color: indianred;
        color: white;
      }
    </style>
  </head>
  <body>
    <button class="btn">Click Me</button>
  </body>
</html>
```

What happens 🧠

- Normal → Light blue button
- Hover → Green background
- Click → Red background

Interview One-Line Answer 🎤

Pseudo-classes are used to style elements based on their state, like hover, click, or position.

CSS Transitions 🎬 (Short & Important)

CSS **transition** is used to create **smooth animation between two states** (like hover, color change, size change).

Main Transition Properties

Property	Meaning
`transition-property`	Which property to animate
`transition-duration`	How long animation takes
`transition-timing-function`	Speed curve (ease, linear, etc.)
`transition-delay`	Delay before animation starts

Shortcut Syntax (Most Used)

```
transition: property duration timing-function delay;
```

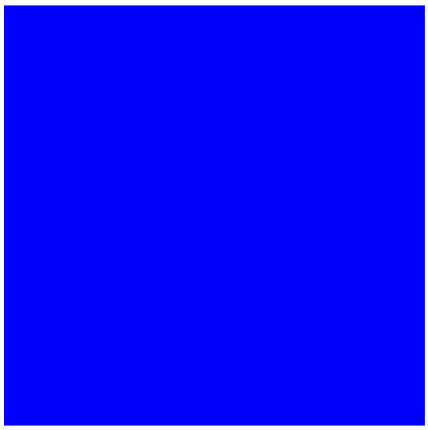
Example:

```
transition: all0.3s ease;
```

Example

```
<!DOCTYPE html>
<html>
<head>
<title>CSS Transition Example</title>

<style>

.box {
  width: 150px;
  height: 150px;
  background-color: blue;
  transition: all 0.5s ease;
}

/* Hover state */
.box:hover {
  background-color: red;
```

```
  width: 200px;
  height: 200px;
}

</style>

</head>
<body>

<div class="box"></div>

</body>
</html>
```



Note 😊

For More Use Refer MDN Documentation

CSS Transform 🧩 (Short & Important)

CSS **transform** is used to **change the position, size, or shape of an element** without affecting other elements.

Main Transform Functions

Function	Meaning
`translate()`	Move element
`scale()`	Increase or decrease size
`rotate()`	Rotate element
`skew()`	Tilt element

Syntax

```
transform: function(value);
```

Example:

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

```
<!DOCTYPE html>
<html>
<head>
<title>CSS Transform Example</title>

<style>
```

```
.box {
  width: 150px;
  height: 150px;
  background-color: orange;
  margin: 100px;
  transition: 0.5s;
}

/* Transform on hover */
.box:hover {
  transform: rotate(45deg) scale(1.2) translate(50px, 20px);
}

</style>

</head>
<body>

<div class="box"></div>

</body>
</html>
```

Animation

CSS Animation 🎬 (Short, Important, and Clear)

CSS **animation** is used to create **movement and visual effects** using `@keyframes`.

It allows elements to change styles automatically over time.

---

1. Key Concept

Animation has 2 main parts:

1. `@keyframes` → Defines animation steps
2. `animation` property → Applies animation to element

---

2. Syntax

```
@keyframes animation-name {

  from {
    /* starting style */
  }

  to {
    /* ending style */
  }
```

```
}
```

OR

```
@keyframes animation-name {

  0% { }
  50% { }
  100% { }

}
```

Example

```
<!doctype html>
<html>
  <head>
    <title>CSS Animation Example</title>

    <style>
      .box {
        width: 100px;
        height: 100px;
        background-color: red;

        animation-name: moveBox;
        animation-duration: 3s;
        animation-iteration-count: infinite;
      }

      /* Animation definition */
      @keyframes moveBox {
        from {
```

```
          transform: translateX(0px);
        }

        to {
          transform: translateX(300px);
        }
      }
    </style>
  </head>
  <body>
    <div class="box"></div>
  </body>
</html>
```

CSS Animation Properties 🎬

Animation properties control **how animation runs**, like time, speed, repeat, and direction.

---

1. animation-name

Defines the name of the animation (`@keyframes`)

```
animation-name: move;
```

Example:

```
@keyframes move {
    from {
        left:0px;
     }
     to {
         left:200px;
```

```
        }
}
```

2. animation-duration

Defines how long animation takes

```
animation-duration:3s;
```

Meaning: animation runs for 3 seconds.

3. animation-timing-function

Controls animation speed curve

Values:

```
linear/* constant speed */
ease/* slow → fast → slow */
ease-in/* slow start */
ease-out/* slow end */
ease-in-out/* slow start and end */
```

Example:

```
animation-timing-function: ease-in;
```

4. animation-delay

Delay before animation starts

```
animation-delay:2s;
```

Animation starts after 2 seconds.

5. animation-iteration-count

Defines number of repeats

```
animation-iteration-count:5;
```

OR infinite loop:

```
animation-iteration-count: infinite;
```

---

6. animation-direction

Defines animation direction

Values:

```
normal
reverse
alternate
alternate-reverse
```

Example:

```
animation-direction: alternate;
```

Moves forward and backward.

```
<!doctype html>
<html>
  <head>
    <title>Animation Properties</title>

    <style>
      .box {
        width: 100px;
        height: 100px;
```

```
        background: red;
        position: relative;

        animation-name: moveBox;
        animation-duration: 3s;
        animation-delay: 1s;
        animation-iteration-count: infinite;
        animation-direction: alternate;
        animation-timing-function: ease-in-out;
      }

      @keyframes moveBox {
        from {
          left: 0px;
        }

        to {
          left: 300px;
        }
      }
    </style>
  </head>
  <body>
    <div class="box"></div>
  </body>
</html>
```