Introduction to CSS

ABSTRACT

This module introduces CSS, its types (inline, internal, external), and selectors with specificity. It covers key properties for backgrounds, text, fonts, borders, and the box model. Learners also explore layout techniques using display, flexbox, grid, and positioning for effective web page design.

Unit-4 Introduction to CSS

Significance of CSS

CSS stands for **Cascading Style Sheet**.

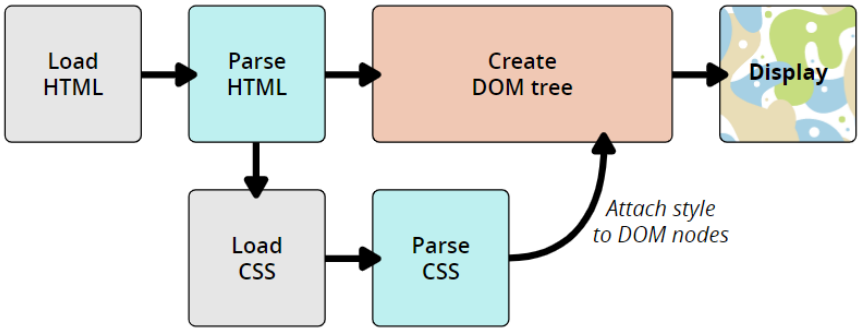
- **CSS** is a style sheet language used to control the **presentation** (look and layout) of web pages written in HTML.
- It defines how elements should be **displayed**, including colors, fonts, spacing, and overall layout.
- Without CSS, web pages use the **browser's default styles**, resulting in a plain and basic appearance.

- **CSS allows developers to:**
  - ✓ Create visually appealing, consistent web pages.
  - ✓ Maintain **separation of content (HTML)** and **presentation (CSS)** for better organization.
  - ✓ Reuse the same stylesheet across multiple web pages, saving time and effort.
  - ✓ Enhance accessibility and improve user experience across devices and screen sizes.

Initially, without adding a CSS, What you are seeing are the browser's default styles, very basic styles that the browser applies to HTML to make sure that the page will be basically readable even if no explicit styling is specified by the author of the page.

The web would be a boring place if all websites looked like that. **Using CSS**, you can control exactly how HTML elements look in the browser, presenting your markup using whatever design you like.

How CSS works with HTML?



How CSS Works

1. **Load HTML** – The browser first loads the web page's structure, which contains all the text, headings, and images.
2. **Parse HTML** – It then reads and understands the HTML elements on the page (like titles, paragraphs, etc.).
3. **Create DOM Tree** – The browser builds a map (a tree-like structure) showing how all the elements are connected.
4. **Load and Parse CSS** – Next, the browser loads the CSS file and reads the styles , such as colors, fonts, and layouts.
5. **Attach Styles to HTML Elements** – The browser matches each style rule to the right HTML elements in the DOM.
6. **Display the Page** – Finally, the browser combines the structure (HTML) and the style (CSS) and shows the beautifully designed page on your screen.

Imagine opening a recipe website:

- HTML gives you the **content** —> the title, ingredients, and steps.
- CSS makes it **look nice** —> colorful headings, clean layout, and easy-to-read text.
- The browser mixes both together and displays a clear, well-styled recipe page for you.

Structure or Syntax of CSS

- ✓ The syntax of CSS is slightly different from that of an HTML.
- ✓ CSS uses **(curly braces { }), (colons :)** and **(semicolon ;)**.

Syntax:

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

- **selector:** Specifies which HTML element(s) to style (e.g., tag(element), id, class, etc.).
- **property:** The aspect of the element you want to change (e.g., color, font-size).
- **value:** The specific setting for that property.

Example:

```
p {
color: blue;
text-align: center;
}
```

- ✓ **p is a selector(element).** It styles all the **<p>** element of the document with color blue and aligned the content in the center of the screen.
- ✓ **color and text-align** are **properties**, and **blue and center** are the respective property **values**.

Types of CSS

Type of CSS	Description	Where It Is Written	Syntax (Example)
1. Inline CSS	Used to style a **single HTML element** directly. It has the **highest priority**.	Inside the HTML tag using the style attribute.	Heading
2. Internal (Embedded) CSS	Used to style elements **within the same HTML page**.	Inside the <style> tag in the <head> section of the HTML document.	<head><style> p { color: red; font-size: 18px; } </style></head>
3. External CSS	Used to apply styles to **multiple web pages** using a separate .css file.	In an external stylesheet linked using the <link> tag in the <head> section.	<head><link rel="stylesheet" href="style.css"></head>*(style.css file contains CSS rules)*

1) Inline CSS

- Used to apply a **unique style** to a single HTML element.
- Uses the **style attribute** inside the element tag.

Syntax:

```
<element style="property:value; property:value;"></element>
```

Example:

```
<h1 style="color:green; font-size:30px;">A Green Heading</h1>
```

Explanation:

- The text color of the <h1> element is set to **green** and the font size to **30px**.
- Inline CSS has the **highest priority** among all CSS types.

2) Internal (Embedded) CSS

- Used to define styles for **a single HTML page**.
- Written inside the <style> tag within the <head> section of the page.

Syntax:

```
<head>
 <style>
  selector {
    property-name1: value;
    property-name2: value;
  }
 </style>
</head>
```

Example:

```
<head>
  <style>
    body { background-color: pink; }
    h1 { color: blue; }
    p { color: purple; font-size: 18px; }
  </style>
</head>
```

Explanation:

- The background color of the page becomes **pink**.
- All headings (<h1>) are **blue**.
- Paragraphs (<p>) are **purple** with font size **18px**.

3) External CSS

- Styles are written in a **separate .css file**.
- The CSS file is linked to the HTML page using the <link> tag inside the <head> section.
- It helps keep HTML clean and allows styles to be reused across multiple pages.

Example

(HTML file):

```
<head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>
```

(style.css file):

```
body { background-color: pink; }
h1 { color: blue; }
p { color: purple; font-size: 18px; }
```

Why Use External CSS

- ✓ Keeps **design and content separate** for better structure.
- ✓ Allows **reusability**, one CSS file can style multiple pages.
- ✓ Simplifies **maintenance**, change one file to update the entire website.
- ✓ Ensures **consistent design** across all web pages.

Various CSS Selectors

A **CSS selector** is used to **target HTML elements** so you can **apply styles** to them.

- Select elements to style.
- Apply styles efficiently to many elements at once.
- Control which styles take priority (specificity).
- Enable dynamic effects with pseudo-classes/elements.

Selector Type	Description	Syntax	Example
Element Selector	Selects all elements of a specific type/tag. Lowest specificity (except universal).	**element**	p { color: black; }
ID Selector	Selects a single element with a specific id attribute. Very high specificity.	**#id**	#header { color: red; }
Class Selector	Selects elements with a specific class. Medium specificity.	**.class**	.menu { font-size: 16px; }
Attribute Selector	Selects elements based on an attribute or attribute value.	**[attr], [attr=value]**	[type="text"] { border: 1px solid #ccc; }
Universal Selector	Selects all elements. Lowest specificity.	*****	* { margin: 0; padding: 0; }
Descendant Selector	Selects elements that are descendants (any level) of a specified ancestor.	**ancestor descendant**	div p { color: blue; }
Child Selector	Selects elements that are direct children of a specified parent.	**parent > child**	ul > li { list-style: none; }
Grouping Selector	Groups multiple selectors and applies the same styles.	**selector1, selector2**	h1, h2, h3 { font-family: Arial; }

1. Element Selector

Syntax:

element-name { property: value; property: value; }

Example:

Here, all <p> elements on the page will be center-aligned, with a red text color:

```
<html>
<head>
<style>
        p {
        text-align: center;
        color: red;
        }
</style>
</head>
<body>
        <p>Test<p>
        <p>Hello!</p>
</body>
</html>
```

Output:

Test

Hello!

---

2. CSS ID Selector

- ✓ The id selector uses the **id attribute** of an HTML element to select a specific element.
- ✓ The id of an element is unique within a page, so the id selector is used to select one unique element!
- ✓ To select an element with a specific id, write a **hash (#)** character, followed by the **id of the element**.

Note: id attribute must begin with a letter and is case sensitive

Synatx:

#element-id{ property1: value; property2: value; }

Example:

```
<html>
<head>
<style>
#para1 {
        text-align: center;
        color: red;
}
</style>
</head>
```

```
<body>
        <p id="para1">Paragraph with ID</p>
        <p>Paragraph without ID</p>
</body>
</html>
```

Output:

Paragraph with ID

Paragraph without ID

---

3. CSS Class Selector

- ✓ The class selector selects HTML elements with a specific **class attribute**.
- ✓ To select elements with a specific class, write a **period (.)** character, followed by the **class name**.

Syntax:

.element-classname { property: value; }

Example1:

```
<head>
<style>
        .center{
        text-align: center;
        color: red;
        }
</style>
</head>
<body>
        <h1 class="center">H1 tag using class selector</h1>
        <p class="center">P tag using class selector</p>
</body>
```

H1 tag using class selector

P tag using class selector

Example 2:

✓ **You can also specify that only specific HTML elements should be affected by a class.**

✓ In this example only <p> elements with class="center" will be affected.

```
<head>
<style>
        p.center {
        text-align: center;
        color: red;
        }
</style>
</head>
<body>
        <h1 class="center">No effect of center class</h1>
        <p class="center">Red and Center aligned</p>
        <p>No effect of center class </p>
</body>
```

Output:

No effect of center class

Red and Center aligned

No effect of center class

4. Universal Selector

- ✓ The universal selector (*) selects all HTML elements on the page.

Syntax:

```
* { property: value; }
```

Example:

- ✓ The CSS rule below will affect every HTML element on the page:

```
<head>
<style>
* {
text-align: center;
color: blue;
}
</style>
</head>
<body>
<h1>Universal Selector</h1>
<p>Using this</p>
<pre>Every element on the page will be affected by the style.</pre>
</body>
```

Output:

Universal Selector

Using this

Every element on the page will be affected by the style.

By adding a css to particular element as shown below. It will override the color of p element(s).

```
<style>
* {
text-align: center;
color: blue;
}
p{
  color: red;
}
</style>
```

Universal Selector

Using this

Every element on the page will be affected by the style.

---

5. Attribute selector:

- ✓ It is possible to style HTML elements that have specific attributes or attribute values.
- ✓ The [attribute] selector is used to select elements with a specified attribute.
- ✓ The attribute selectors can be useful for styling forms or any other elements using their attribute.

Syntax:

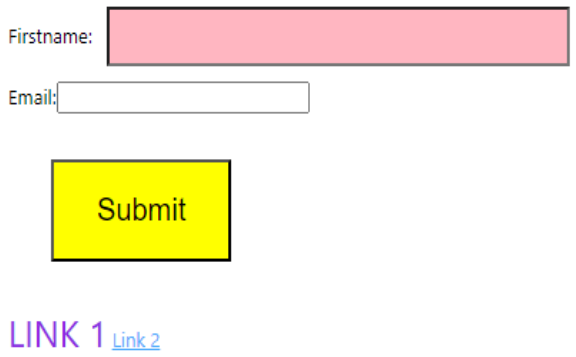
```
element-name[type=”text”]
{
property: value;
}
```

Example:

```
<head>
<style>
input[type=text] {
        width: 300px;
        margin-bottom: 10px;
        background-color: lightpink;
        color:rgb(125, 73, 247);
        padding:10px;
        margin: 10px;
}
input[type=button] {
        margin:30px;
        background-color: yellow;
        padding:20px 30px;
        font-size:20px;
}
a[target=_blank] {
        color: blueviolet;
        font-size: x-large;
        text-decoration: none;
        text-transform: uppercase;
}
</style>
</head>
```

```
<body>
<form name="input" action="" method="get">
Firstname:<input type="text" name="Name"> <br>
Email:<input type="email" name="Name"><br>
<input type="button" value="Submit">
</form>
<a href="#" target="_blank">Link 1</a>
<a href="#">Link 2</a>
</body>
```

Output:



CSS Combinators

- ✓ A combinator is something that explains the relationship between the selectors.
- ✓ A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

Combinators in CSS:

- **descendant selector (space)**
- **child selector (>)**

6. Descendant Selector

- ✓ If some tag is nested in the other tag then nested tag is called as descendant of parent tag.
- ✓ The Descendant Selectors can be any selector having the white-space in between the elements without using any combinators. Descendant is a manner to nested anywhere within the DOM tree. This selector is used to select all the child elements of the specified tag.

Syntax:

```
element1 element2
{
    property: value;
}
```

Example :

```
<head>
<style>
div.d1 p {
  background-color: lightblue;
}
</style>
</head>
<body>
<div class="d1">
  <p>Paragraph 1 in the div.</p>
   <section>
         <p>Paragraph 2 in the div.</p>
   </section>
   <p>Paragraph 3 in the div.</p>
</div>

<p>Paragraph 4 After a div.</p>
</body>
```

Output:

It applies background color to the all nested **p** elements of the **div** element. (child, grandchild, great grandchild …..)

7. Child Selector >

- ✓ The child selector selects all elements that are the children of a specified element.
- ✓ To apply CSS, nested tag must be a direct child of previous tag.
- ✓ The following example selects all <p> elements that are children of a element:

Syntax:

```
element1 > element2
{
property:value;
}
```

Example 1:

```
<head>
  <style>
  .d1>p {
    background-color: lightblue;
  }
  </style>
  </head>
  <body>

  <div class="d1">
    <p>Paragraph 1 in the div.</p>
     <section>
        <p>Paragraph 2 in the div.</p>
     </section>
     <p>Paragraph 3 in the div.</p>
  </div>

  <p>Paragraph 4 After a div.</p>
  </body>
```

Output:

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3 in the div.

Paragraph 4 After a div.

- ✓ It applies background color to only direct child p element/s of div element.
- ✓ Grandchild , great grandchild etc will not get affected in child selector.

8. Grouping Selector

The **Grouping Selector** allows you to **apply the same style to multiple elements at once**, saving time and reducing repetition in your CSS.

Syntax:

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

Example:

```
h1, h2, h3 {
   font-family: Arial, sans-serif;
   color: darkblue;
}
```

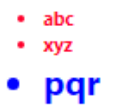
- This rule applies the same **font** and **color** to all <h1>, <h2>, and <h3> elements.

Example:

Write CSS to perform the tasks as asked below.

1. **Add unordered list with 3 list items.**
2. **If direct child of the ul element is <b> then apply font color blue and font size should be 20px.**
3. **If direct child of the ul element is <li> then item should be displayed in red color and in smaller font size.**

```
<head>
   <style>
   ul>b {
      font-size:20px;
      color:blue;
   }
   ul>li {
      font-size:smaller;
      color:rgb(255, 0, 43);
   }
   </style>
</head>
<body>
   <ul> <li><b>abc</b></li>
      <li><b>xyz</b></li>
      <b><li>pqr</li></b>
   </ul>
</body>
```



CSS Specificity (Priority Order)

When multiple CSS rules target the same element, the browser decides which one to apply based on **specificity** (priority):

Priority Level	Selector Type	Example
1 (Highest)	Inline Styles	Heading
2	IDs	#title { color: red; }
3	Classes, pseudo-classes, attribute selectors	.title { color: green; }, :hover, [type='text']
4 (Lowest)	Elements and pseudo-elements	h1 { color: blue; }, ::before

```
<head>
  <style>
    h1 { color: blue; }          /* Element selector */
    .main { color: green; }      /* Class selector */
    #title { color: red; }       /* ID selector */
  </style>
</head>
<h1 id="title" class="main" style="color:pink;">Welcome!</h1>
```

Rule Type	Color	Priority Level	Who Wins?
h1 { color: blue; }	Blue	Lowest (Element)	✗ Overridden
.main { color: green; }	Green	Medium (Class)	✗ Overridden
#title { color: red; }	Red	High (ID)	✗ Overridden
style="color:pink;"	Pink	Highest (Inline)	✓ **Applied**

Text Properties

These properties are used to **style, format, and control the appearance** of text on a webpage, including alignment, spacing, transformation, decoration, font type, and size.

Text Properties

Property	Description	Syntax	Possible Values
color	Sets the color of the text	color: color;	Named colors (e.g., red), HEX (#ff0000), RGB (rgb(255,0,0)), RGBA, HSL, HSLA
text-align	Aligns text inside an element	text-align: value;	left, right, center, justify, start, end
text-indent	Adds indentation to the first line	text-indent: length;	Any length (px, em, %)
text-transform	Controls text capitalization	text-transform: value;	none, capitalize, uppercase, lowercase
text-decoration	Adds decoration like underline or line-through	text-decoration: value;	none, underline, overline, line-through, underline overline
letter-spacing	Controls space between letters	letter-spacing: length;	Normal (normal), or custom spacing (px, em)
word-spacing	Controls space between words	word-spacing: length;	Normal (normal), or custom spacing (px, em)
line-height	Sets space between lines	line-height: value;	normal, number (e.g., 1.5), length (px, em), %
text-shadow	Adds shadow to text	text-shadow: h-shadow v-shadow blur color;	none or values like 2px 2px 5px gray (can add multiple shadows separated by commas)

Example:

```
<head>
  <title>CSS Text Properties Example</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      padding: 20px;
      line-height: 1.6;
    }

    /* Underline text */
    .underline {
      text-decoration: underline;
      color: blue;
      text-align: left;
      text-transform: capitalize;
      text-indent: 30px;
```

```
        letter-spacing: 2px;
        word-spacing: 5px;
        text-shadow: 2px 2px 3px gray;
    }

    /* Overline text */
    .overline {
        text-decoration: overline;
        color: green;
        text-align: center;
        text-transform: uppercase;
        text-indent: 0px;
        letter-spacing: 1px;
        word-spacing: 3px;
        text-shadow: 1px 1px 2px black;
    }

    /* Line-through text */
    .line-through {
        text-decoration: line-through;
        color: red;
        text-align: right;
        text-transform: lowercase;
        text-indent: 0px;
        letter-spacing: 1px;
        word-spacing: 2px;
        text-shadow: 1px 1px 3px gray;
    }

    /* Combined text-decoration */
    .combined {
        text-decoration: underline overline;
        color: purple;
        text-align: justify;
        text-transform: capitalize;
        text-indent: 40px;
        letter-spacing: 3px;
        word-spacing: 6px;
        text-shadow: 3px 3px 5px gray;
    }
  </style>
</head>
<body>

<h2>CSS Text Properties Demo</h2>
<p class="underline">
This text demonstrates <b>underline</b> with all other text properties applied.
```

```
</p>

<p class="overline">
This text demonstrates <b>overline</b> with all other text properties applied.
</p>
<p class="line-through">
This text demonstrates <b>line-through</b> with all other text properties applied.
</p>
<p class="combined">
This text demonstrates <b>combined underline + overline</b> with all other text properties applied.
</p>
</body>
```

- .underline → blue, left-aligned, capitalized, indented, spaced letters/words, with shadow.
- .overline → green, centered, uppercase, spaced letters/words, with shadow.
- .line-through → red, right-aligned, lowercase, spaced letters/words, with shadow.
- .combined → purple, justified, capitalized, combined underline + overline, letter & word spacing, shadow.

Note:
Use left / right if your site is **single-language** and fixed layout.
Use start / end for **multilingual, adaptive layouts** — it ensures your design **works for LTR and RTL automatically**.

Font Properties

Property	Description	Syntax	Example	Possible Values
font-family	Specifies the font type	font-family: "FontName", fallback;	font-family: "Arial", sans-serif;	Any font name: "Arial", "Times New Roman", "Courier New", generic families: serif, sans-serif, monospace, cursive, fantasy
font-size	Sets the font size	font-size: size;	font-size: 20px;	Length units: px, em, rem, %; Keywords: xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger
font-style	Defines style of text	font-style: value;	font-style: italic;	normal, italic
font-weight	Sets font thickness	font-weight: value;	font-weight: bold;	normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900
font-variant	Displays text in small-caps	font-variant: value;	font-variant: small-caps;	normal, small-caps

Example:

```
<head>
  <title>CSS Font Properties Demo</title>
  <style>
    /* Individual font properties */
    .font1 {
      font-family: "Times New Roman", serif;
      font-style: italic;       /* normal, italic*/
      font-size: 25px;          /* Size in pixels */
      font-weight: bold;        /* normal, bold, 100-900 */
      font-variant: small-caps;  /* normal, small-caps */
    }
  </style>
</head>
<body>

<p class="font1">
This paragraph demonstrates <b>individual font properties:</b> Times New Roman, italic, bold, small-caps, 25px size.
</p>

</body>
```

THIS PARAGRAPH DEMONSTRATES INDIVIDUAL FONT PROPERTIES: TIMES NEW ROMAN, ITALIC, BOLD, SMALL-CAPS, 25PX SIZE.

Google Fonts

Why Use Google Fonts?

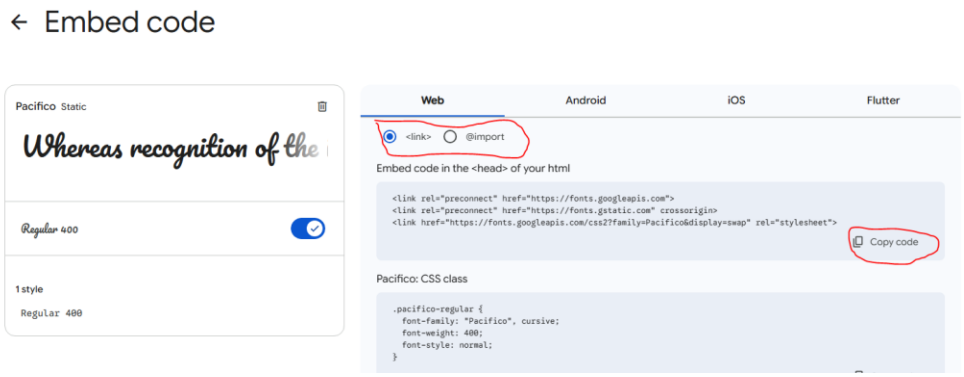
1. **Variety of Fonts** – Access hundreds of high-quality, free fonts beyond standard system fonts.
2. **Cross-Browser & Cross-Device** – Fonts are hosted online, so they render consistently across browsers and devices.
3. **Performance Optimized** – Google Fonts servers are fast and optimized for web use.
4. **Easy to Use** – Simple embed methods (<link> or @import).
5. **Customizable Weights & Styles** – Choose different weights (400, 600, 700) and styles (italic, normal).

How to Embed Google Fonts

Method 1: Using <link> tag (Recommended)

1. **Go to Google Fonts.**
2. **Search and select your desired font (e.g.,** Pacifico**).**
3. **Choose the styles/weights you need (e.g., 400).**

4. **Copy the <link> tag under Embed section:**



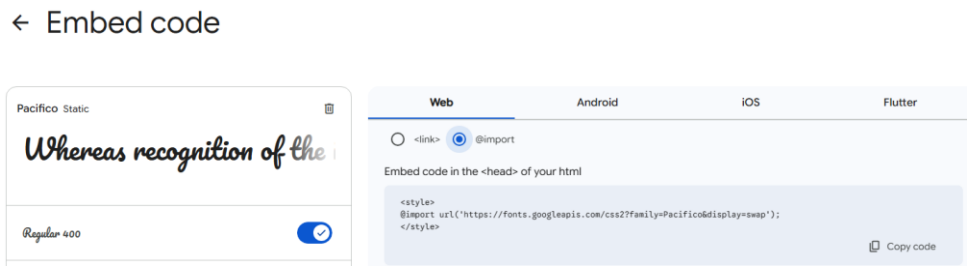

```
<link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">
```

5. **Paste it inside the <head> of your HTML file.**
6. **Apply the font in CSS:**

```
body {
    font-family: "Pacifico";
}
```

Method 2: Using @import in CSS

1. **Go to Google Fonts and select your font.**



2. **Copy the @import code:**
   @import url('https://fonts.googleapis.com/css2?family=Pacifico&display=swap');
3. **Paste it at the top of your CSS file, before other CSS rules.**
4. **Use the font in CSS:**
   body { font-family: "Pacifico"; }

Example:

font_example.html

```
<head>
<link href="https://fonts.googleapis.com/css2?family=Pacifico&display=swap" rel="stylesheet">
<link rel="stylesheet" href="style.css">
<style>
   h1{
      font-family: "Pacifico";
      font-weight: 400;
      font-style: normal;
   }
</style>
</head>
<body>
   <h1>google font Using link</h1>
   <h3>Google font using import</h3>
</body>
```

style.css

```
@import url('https://fonts.googleapis.com/css2?family=Pacifico&display=swap');
h3{
   font-family: "Pacifico";
   color: blue;
   font-size:30px;
}
```

CSS Borders

A **border** is a line that wraps around an HTML element's **content and padding**. It visually separates elements or highlights them on a web page.

Property	Description	Why to Use	Possible Values / Syntax
border-style	Defines the style of the border	To give different visual effects like solid, dashed, dotted, etc.	none, solid, dashed, dotted, double
border-width	Sets the thickness of the border	To adjust border size	thin, medium, thick, or length (px, em, rem)
border-color	Sets the color of the border	To visually match design or highlight elements	Named colors (red), hex (#FF0000), RGB (rgb(255,0,0))
border (shorthand)	Combines style, width, color in one line	To quickly define a complete border	border: 2px solid red;
border-top	Sets border properties for the top side	To style only the top edge	border-top: 3px dashed blue;
border-right	Sets border properties for the right side	To style only the right edge	border-right: 2px solid green;
border-bottom	Sets border properties for the bottom side	To style only the bottom edge	border-bottom: 4px dotted orange;
border-left	Sets border properties for the left side	To style only the left edge	border-left: 5px double purple;
border-radius	Rounds the corners of an element's border box.	To make elements look smoother or create circular shapes.	border-radius: 10px;

Note:

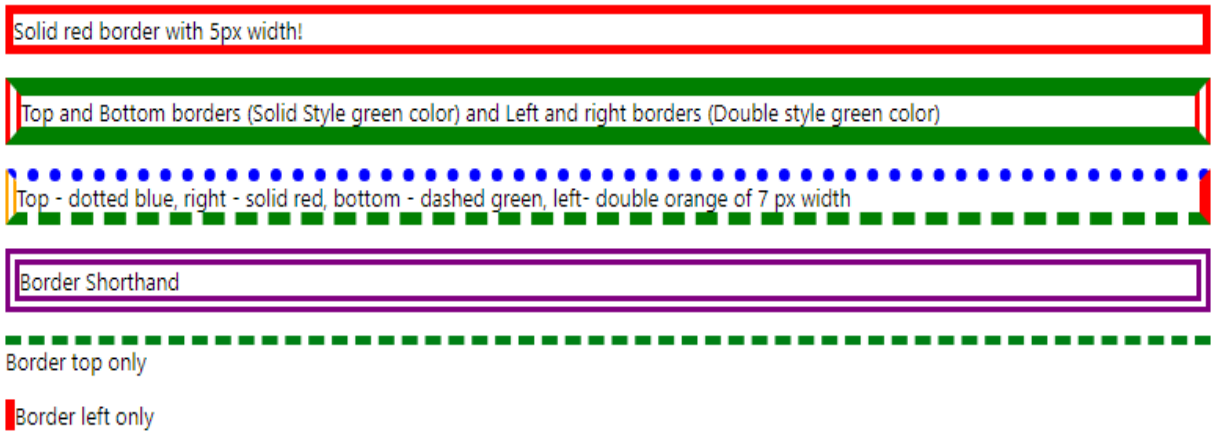
- ✓ The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.
- ✓ If border-color is not set, it inherits the color of the element.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  border-style: solid;
  border-color: red;
  border-width:5px;
}
p.two {
  border-style: solid double;
```

```
  border-color: green red;
  border-width:10px;
}
p.three {
  border-style: dotted solid dashed double;
  border-color: blue red green orange;
  border-width:7px;
}
p.four {
  border:9px double purple;
}
p.five {
  border-top:5px dashed rgb(0, 128, 21);
}
p.six {
  border-left:6px solid red;
}
</style>
</head>
<body>
<p class="one">Solid red border with 5px width!</p>
<p class="two">Top and Bottom borders (Solid Style green color) and Left and right borders (Double style green color)</p>
<p class="three">Top - dotted blue, right - solid red, bottom - dashed green, left- double orange of 7 px width</p>
<p class="four">Border Shorthand </p>
<p class="five">Border top only </p>
<p class="six">Border left only </p>
<p><b>Note:</b> The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.</p>
</body>
</html>
```

Output:



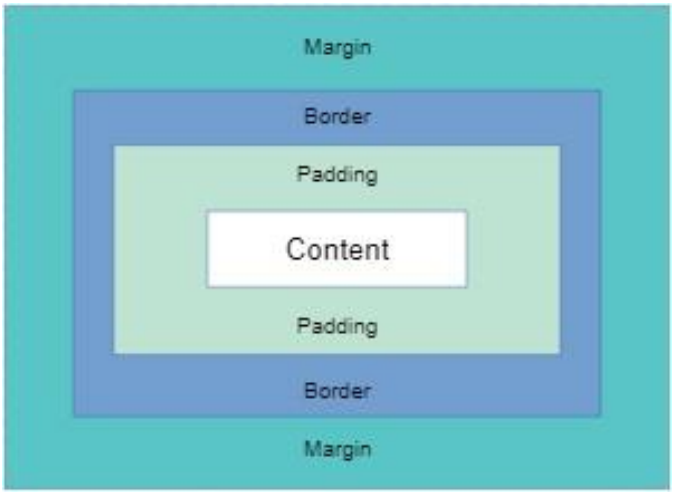
Box Properties

Box properties control the **size**, **spacing**, and **appearance** of HTML elements.
They are part of the **CSS Box Model**, which consists of:

- **Content** – the actual text or image inside the box
- **Padding** – space between content and border
- **Border** – the edge surrounding the padding
- **Margin** – space outside the border

Box properties allow you to define:

- How much **space an element occupies** (using width and height)
- How far it is from **other elements** (margin)
- The **space between content and border** (padding)
- **Borders and shadows** to highlight or style elements (box-shadow)
- How **width and height** are calculated (box-sizing)



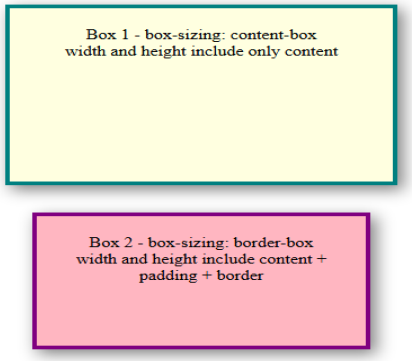
Property	Description	Why to Use	Possible Values / Syntax / Examples
margin	Creates **space outside** the element (outside border).	To **separate elements** on the page.	**Values:** • **auto** → browser calculates margin (useful for centering) • **length** → px, em, %, negative values allowed **Shorthand examples:** • 4 values → margin: 25px 50px 75px 100px; → top=25px, right=50px, bottom=75px, left=100px • 2 values → margin: 25px 50px; → top/bottom=25px, right/left=50px • 1 value → margin: 25px; → all sides=25px

Property	Description	Why to Use	Possible Values / Syntax / Examples
padding	Creates **space inside** the element (between content & border).	To **add inner spacing** inside the element.	**Values:** • length → px, em, % **Shorthand examples:** • 4 values → padding: 25px 50px 75px 100px; → top=25px, right=50px, bottom=75px, left=100px • 2 values → padding: 25px 50px; → top/bottom=25px, right/left=50px • 1 value → padding: 25px; → all sides=25px
width	Specifies the **width** of an element's content area.	To **control horizontal size** of the element.	**Values:** auto, length (px, em, %),
height	Specifies the **height** of an element's content area.	To **control vertical size** of the element.	**Values:** auto, length (px, em, %),
box-sizing	Defines **how width and height are calculated** — whether padding and borders are included.	To **control layout behavior** when adding padding/borders.	**Values:** • content-box (default): width/height exclude padding & border. • border-box: width/height include padding & border.
box-shadow	Adds **shadow effects** around an element's frame.	To **add depth**, **hover effects**, or **visual highlights**.	**Syntax:** box-shadow: h-offset v-offset blur spread color; **Example:** box-shadow: 2px 2px 5px gray;

Example

```
<head>
<style>
.box1 {
   width: 300px;              /* width of box */
   height: 150px;             /* height of box */
   margin: 30px auto;          /* outer spacing & centered horizontally */
   padding: 20px;             /* inner spacing */
   border: 4px solid teal;       /* border around the box */
   box-sizing: content-box;      /* width/height includes content only */
   box-shadow: 5px 5px 15px gray; /* shadow around the box */
   background-color: lightyellow;
   text-align: center;
}
.box2 {
   width: 300px;
   height: 150px;
   margin: 30px auto;
   padding: 20px;
   border: 4px solid purple;
```

```
   box-sizing: border-box;       /* width/height includes content + padding + border */
   box-shadow: 5px 5px 15px gray;
   background-color: lightpink;
   text-align: center;
}
</style>
</head>
<body>
<h2>Box Properties Demonstration</h2>
<div class="box1">
   Box 1 - box-sizing: content-box<br>
   width and height include only content
</div>
<div class="box2">
   Box 2 - box-sizing: border-box<br>
   width and height include content + padding + border
</div>
</body>
```



Box-Sizing Comparison

Box 1 – content-box

- Declared size: 300×150 px
- Padding: 20 px, Border: 4 px
- Total width: 300 + 20 + 20 + 4 + 4 = 348 px
- Total height: 150 + 20 + 20 + 4 + 4 = 198 px

Padding and border are added outside the content.

Box 2 – border-box

- Declared size: 300×150 px
- Padding: 20 px, Border: 4 px
- Content width: 300 − (20 + 20 + 4 + 4) = 252 px
- Content height: 150 − (20 + 20 + 4 + 4) = 102 px

Padding and border are included within the declared size.

Box	box-sizing	Declared Size	Rendered Size	Notes
Box 1	content-box	300×150 px	348×198 px	Adds padding + border
Box 2	border-box	300×150 px	300×150 px	Includes padding + border

CSS background properties

CSS background properties are a set of properties that allow you to **control the appearance of the background** of HTML elements. This includes **color, images, repetition, position, size, attachment, and clipping**.

Why use them?

- To **enhance the visual design** of a webpage.
- To **highlight elements** with colors or images.
- To **control the layout and behavior** of background images (repeat, size, position, scroll behavior).
- To **create visually appealing effects** using shorthand or combined properties.

Property	Description	Syntax	Key Values / Notes
background-color	Sets the background color of an element	background-color: color;	Any valid color (red, #ffc0cb, rgb(255,0,0))
background-image	Sets an image as the background	background-image: url("image.jpg");	Use image URL;
background-repeat	Controls how background image repeats	background-repeat: repeat;	repeat, repeat-x, repeat-y, no-repeat default repeats if not specified
background-position	Sets the starting position of a background image	background-position: top;	top, bottom, left, right, center, x% y%
background-size	Specifies the size of the background image	background-size: auto;	auto, cover, contain, width height
background-attachment	Sets whether background scrolls with content	background-attachment: scroll;	scroll, fixed
background-clip	Determines how far the background extends	background-clip: border-box;	border-box, padding-box, content-box
background (shorthand)	Combines color, image, repeat, position, size, attachment	background: [color] [image] [repeat] [position] [size] [attachment];	Example: background: pink url("scenary.jfif") no-repeat top right cover fixed;

Note: Do not prefer to use shorthand property in exam. Prefer to use separate properties to perform the tasks as asked.

background-color

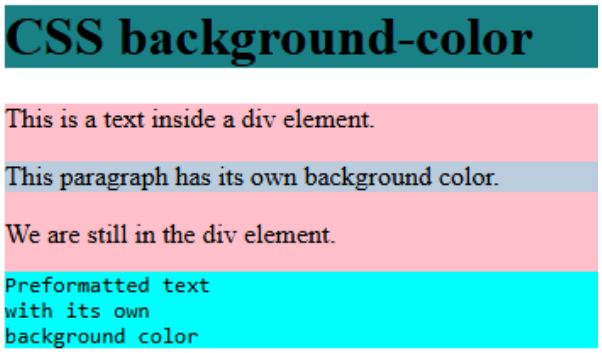
The background-color property specifies the background color of an element.

Example

```
<html>
<head>
<style>
h1 {background-color: rgb(25, 128, 134);}
div {background-color: #FFC0CB;}
p {background-color: #bcd;}
pre {background-color: aqua;}
```

```
</style>
</head>
<body>
<h1>CSS background-color</h1>
<div>
        This is a text inside a div element.
        <p>This paragraph has its own background color.</p>
        We are still in the div element.
        <pre>Preformatted text
        with its own
        background color
        </pre>
</div>
</body>
</html>
```

Output:



Understanding Of color selection

Format	Example	Full Form / Meaning	Value Range	Key Points
HEX (6 digits)	#ffffff	Hexadecimal RGB value	00 → ff	Each pair = Red, Green, Blue; common format for web colors
HEX (3 digits)	#fff	Short form of HEX	0 → f	Shorthand for 6-digit HEX (#fff = #ffffff)
RGB	rgb(255, 255, 255)	Red, Green, Blue	0 → 255	Easy to visualize and edit color intensities
RGBA	rgba(255, 0, 0, 0.5)	Red, Green, Blue, Alpha	0 → 255 (RGB), 0 → 1 (Alpha)	"A" controls transparency (1 = opaque, 0 = fully transparent)

background-image

- ✓ Sets the background image for an element
- ✓ The background-image property specifies an image to use as the background of an element.
- ✓ By default, the image is repeated so it covers the entire element.
- ✓ The background image can also be set for specific elements, like the <p> element.

Note: When using a background image, use an image that does not disturb the text.

Example:

```
<html>
<head>
<style>
body { background-image: url("scenary.jfif"); }
</style>
</head>
<body>
        <h1>BACKGROUND IMAGE</h1>
        <p>By default, the image is repeated so it covers the entire element.</p>
</body>
</html>
```

Output:

background-repeat

- ✓ **By default, the background-image property repeats an image both horizontally and vertically.**
- ✓ Some images should be repeated only horizontally or vertically, or they will look strange.
- ✓ To repeat an image vertically, set background-repeat: repeat-y;
- ✓ To repeat an image horizontally, set background-repeat: repeat-x;
- ✓ To repeat an image only once, set background-repeat: no-repeat;

Example (repeat-x):

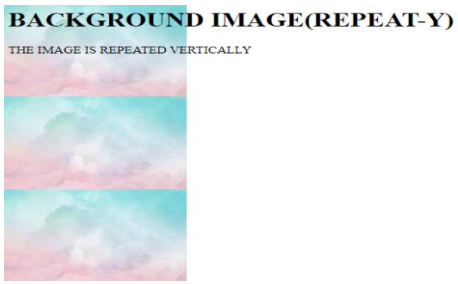
```
<head>
  <style>
  body {
        background-image: url("nature.jfif");
        background-repeat: repeat-X;
        }
  </style>
  </head>
  <body>
        <h1>BACKGROUND IMAGE(REPEAT-X)</h1>
         <p>THE IMAGE IS REPEATED HORIZONATLLY</p>
  </body>
```

Output:

Example (repeat-y):

```
<head>
<style>
body {
         background-image: url("scenary.jfif");
         background-repeat: repeat-y;
        }
</style>
</head>
<body>
        <h1>BACKGROUND IMAGE(REPEAT-Y)</h1>
        <p>THE IMAGE IS REPEATED VERTICALLY</p>
</body>
```

Output:



Example (no-repeat)

```
<head>
<style>
body {
        background-image: url("scenary.jfif");
         background-repeat: no-repeat;
}
</style>
</head>
<body>
        <h1>BACKGROUND IMAGE(no-repeat)</h1>
        <p>THE IMAGE IS REPEATED ONLY ONCE</p>
</body>
```

Output:

background-clip

background-clip defines **how far the background (color or image)** extends **within an element's box** whether it goes under the border, stops at padding, or only fills the content area.

Value	Description	Background Covers	Visual Effect
border-box (default)	Background extends **under the border**	Border + Padding + Content	Background color/image is visible **behind the border**
padding-box	Background is painted **only inside the padding area**	Padding + Content	Background **stops before the border**
content-box	Background is painted **only behind the content**	Content only	Background **ignores border and padding** (appears smaller)

Example:

```
box1 {
        width: 400px;
        height: 200px;
        padding: 50px;
        background-size:cover;
        background-image: url("nature.jpg");
        border: 20px dotted;
        background-color: rgb(85, 75, 77); /*
background color */
        background-clip:border-box;
        text-align: center;
        font-size: 18px;
    }
<div class="box1">
  This box demonstrates <br>
   background-color + background-clip + border +
padding
</div>
```

If value is "content-box" then below will be the possible output.

Background **ignores border and padding** (appears smaller)

If value is "padding-box" then below will be the possible output.

Background **stops before the border**

Background-size

- background-size defines **how a background image fits or scales** inside an element.
- It controls **whether the image repeats, stretches, or maintains proportions** — very useful for responsive design and banners.

Value	Description	When to Use	Example
auto	Default — image keeps its original size	When you don't want to resize the image	background-size: auto;
cover	Scales image to **completely cover** the area (cropping if needed)	For full-screen backgrounds or banners	background-size: cover;
contain	Scales image to **fit inside** the box without cropping	When you want the full image visible	background-size: contain;
100% 100%	Stretches image to fill width and height	When exact fit is needed	background-size: 100% 100%;
width height	Manually set image size	When you want precise control	background-size: 200px 100px;

Example

```
<head>
  <style>
    .box1 {
      width: 600px;
      height: 400px;
      background-size:auto;
      background-image: url("nature.jpg");
    }
  </style>
</head>
<body>
  <div class="box1">
    This box demonstrates <br> background-size property
  </div></body>
```

If value is “cover” possible output is as below.

If value is “contain” possible output is as below.

If value is specified as “width height” possible output is as below. Here value applied is “300px 200px”. Default value for background-repeat is “repeat”

background-attachment

It defines whether the background image scrolls with the page **or** stays fixed **in place.**
This helps create visual effects like *parallax scrolling* or *fixed banners*.

Value	Description	Effect / Behavior	Common Use
scroll *(default)*	Background image **moves with the page** when you scroll	Image scrolls normally along with the content	Default for most elements
fixed	Background image **stays fixed** in the same place while the content scrolls	Creates a "parallax" or fixed-banner effect	Used in headers, full-screen backgrounds

background-position

The background-position property defines **where the background image starts** inside an element.
By default, a background image starts at the **top-left corner** (0% 0%).
But you can **move or align it** anywhere center, bottom, right etc.

- **Control image placement** (not always top-left)
- **Align the image with text or layout**
- **Show a specific part** of a large image (like a logo or cropped photo)
- **Create design balance** (especially in banners or cards)
- **Combine with background-size** for perfect image alignment

Value	Description	Effect
left top	Default	Image starts at top-left corner
Center	Centers both horizontally and vertically	Image stays in the middle
top / bottom / left / right / center	Keywords for alignment	Combine as top center, bottom right, etc.

Example

```
<head>
  <title>Background Position & Attachment Example</title>
  <style>
    body {
      height: 1000px; /* just to make scrolling visible */
      background-color: lightblue; /* fallback color */
      background-image: url("nature.jpg");
      background-repeat: no-repeat;
      /*  controlling where the image appears */
      background-position: bottom right ; /* places image at bottom-right corner */
      /* controlling how it behaves during scroll */
      background-attachment: fixed; /* image stays fixed while content scrolls */

      /* scaling the image */
      background-size: cover;
```

```
    }
  </style>
</head>
<body>
  <div>
    <h1>CSS background-position + background-attachment</h1>
    <p>
      Scroll this page and notice how the background image stays in the same position
      on the screen because <b>background-attachment: fixed;</b> keeps it fixed, while
      <b>background-position: bottom right;</b> keeps the image anchored to the bottom-right corner.
    </p>
    <p>
      Change <code>background-attachment</code> to <code>scroll</code> and reload —
      you'll see the image moves with the page instead.
    </p>
  </div>
</body>
```

Using Shorthand for Combined Effects

- **Effect:** Apply multiple properties at once for **compact code**.
- **Shorthand Properties Used Together:** background-color, background-image, background-repeat, background-position, background-size, background-attachment

Syntax:

```
background: [color] [image] [repeat] [position] / [size] [attachment];
```

Example:

```
background: lightpink url("scenary.jfif") no-repeat top right / cover fixed;
```

Part	Meaning
lightpink	background-color
url("scenary.jfif")	background-image
no-repeat	background-repeat
top right	background-position
/ cover	background-size (the / separates it from position)
fixed	background-attachment

Note:

/ is mandatory only when you specify both position **and** size. If you specify just one of them, no / is needed.

Note: Do not prefer to use shorthand property in exam. Prefer to use separate properties to perform the tasks as asked.

Pseudo classes

- ✓ A Pseudo class in CSS is used to define the special state of an element.
- ✓ It can be combined with a CSS selector to add an effect to existing elements based on their states.
- ✓ For Example, changing the style of an element when the user hovers over it, or when a link is visited. All of these can be done using Pseudo Classes in CSS.

Note that pseudo-class names are not case-sensitive.

Pseudo-class	What It Does
:hover	Changes color when mouse hovers
:active	Changes color when clicked
:focus	Highlights input field when clicked
:link / :visited	Colors for unvisited/visited links
:nth-child(2)	Targets the 2nd list item
:not(.special)	Styles all list items except .special

Syntax:

```
selector : pseudo-class{
   property: value;
}
```

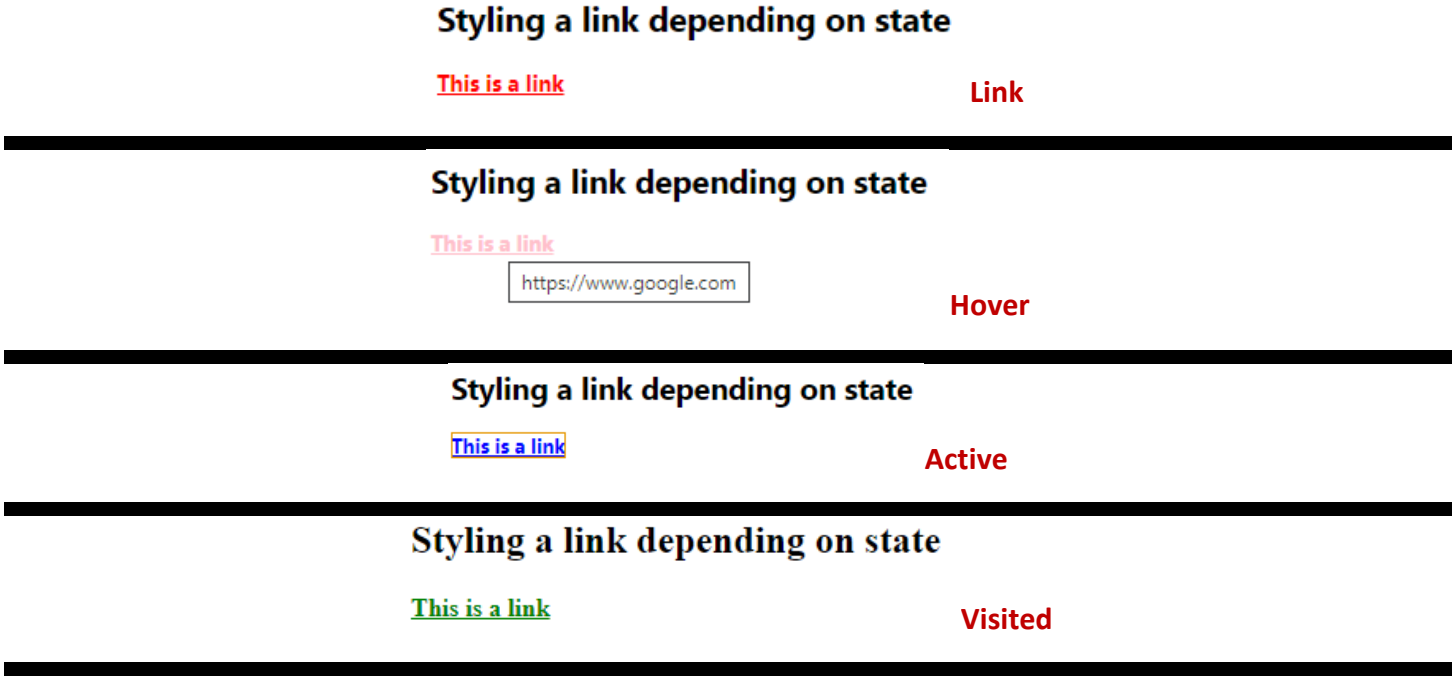
Example

```
<html>
<head>
<style>
/* unvisited link */
a:link {
color: red;
}
/* visited link */
a:visited {
color: green;
}
/* mouse over link */
a:hover {
color: pink;
}
/* selected link */
a:active {
color: blue;
}
</style>
</head>
<body>
```

```
<h2>Styling a link depending on state</h2>
<p><b><a href="https://www.google.com" target="_blank">This is a link</a></b></p>
</body>
</html>
```

a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective.
a:active MUST come after a:hover in the CSS definition in order to be effective.

Output:



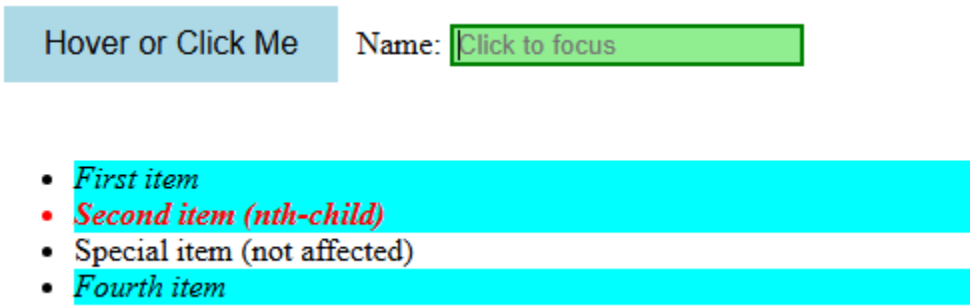
Example

Design a webpage that includes:

1. A **button** that changes its color when hovered over and when clicked.
2. A **text input field** that changes its background color when it gains focus.
3. An **unordered list** where:
    - The **second list item** is styled differently, with **red text color** and **bold font**.
    - One of the list items has a class named **"special"**, and **all other list items** (except the one with this class) have an **aqua background** and **italic font style**.

```
<head><style>
/* ====== BUTTON STATES ====== */
button {
  background-color: lightblue;
  border: none;
  padding: 10px 20px;
  font-size: 16px;
  margin: 5px;
}
button:hover {
```

```
  background-color: navy;
  color: white;
}
button:active {
  background-color: orange;
}
/* ====== INPUT STATES ====== */
input:focus {
  background-color: lightgreen;
  border: 2px solid green;
}
/* ====== LIST ITEMS ====== */
ul li:nth-child(2) {
  color: red;
  font-weight: bold;
}
ul li:not(.special) {
  font-style: italic;
  background-color: aqua;
}
</style></head>
<body>
<!-- Buttons -->
<button>Hover or Click Me</button>
<!-- Input fields -->
  <label for="name">Name:</label>
  <input type="text" id="name" placeholder="Click to focus"><br><br>
<!-- List Example -->
<ul>
  <li>First item</li>
  <li>Second item (nth-child)</li>
  <li class="special">Special item (not affected)</li>
  <li>Fourth item</li>
</ul>
</body>
```



Pseudo Elements

Pseudo-element	Description	Syntax	Example	Result / Effect
::first-line	Styles the **first line** of a block-level element.	p::first-line { property: value; }	p::first-line { color: red; font-weight: bold; }	First line of the paragraph appears red and bold.
::first-letter	Styles the **first letter** of a block-level element.	p::first-letter { property: value; }	p::first-letter { font-size: 50px; color: red; }	First letter of the paragraph becomes large and red.
::before	Inserts content **before** an element's actual content.	selector::before { content: "..."; }	p::before { content: "Note: "; color: red; }	Adds "Note:" before each paragraph.
::after	Inserts content **after** an element's actual content.	selector::after { content: "..."; }	p::after { content: " ✔ "; color: green; }	Adds a green checkmark after the paragraph.
::marker	Styles the **marker (bullet or number)** of list items.	li::marker { property: value; }	li::marker { color: red; font-size: 20px; }	List bullets or numbers appear red and large.
::selection	Styles the **highlighted text** selected by the user.	::selection { property: value; }	::selection { background: blueviolet; color: white; }	Selected text appears white on a blue-violet background.

A CSS pseudo-element is used to style specified parts of an element. It can be used to:

- ✓ A pseudo-element can be used to style the first letter or the first line of an element.
- ✓ The pseudo-elements can also be used to insert the content after or before an element.

Syntax

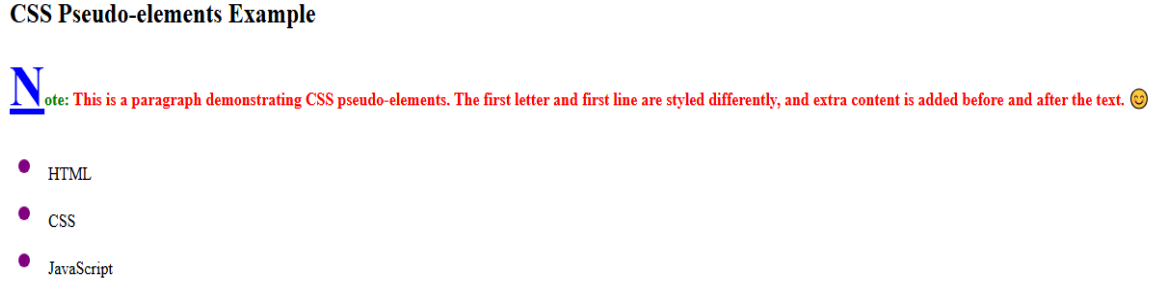
```
selector::pseudo-element {
  property: value;
}
```

We have used the **double colon notation (::pseudo-element)** in the syntax.

In CSS3, the double colon replaced the single colon notation for pseudo-elements. It was an attempt from W3C to differentiate between the pseudo-elements and pseudo-classes. So, it is recommended to use **double colon notation (::pseudo-element)** instead of using single-colon notation **(:)**.

```
<head>
<style>
/* ===== FIRST LINE & FIRST LETTER ===== */
p::first-line {
  color: red;
  font-weight: bold;
}
p::first-letter {
  font-size: 50px;
  color: blue;
  text-decoration: underline;
```

```
}
/* ===== BEFORE & AFTER ===== */
p::before {
  content: " Note: ";
  color: green;
}
p::after {
  content: "😊";
 }
/* ===== MARKER ===== */
ul li::marker {
  color: purple;
  font-size: 40px;
}
/* ===== SELECTION ===== */
::selection {
  background-color: yellow;
  color: red;
}
</style></head>
<body>
<h2>CSS Pseudo-elements Example</h2>
<p>
  This is a paragraph demonstrating CSS pseudo-elements. The first letter and first line are styled differently, and extra content is added before and after the text.
</p>
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>
</body>
```



Insert Emoji (Smiley) Using Keyboard in VS CODE
Windows key + . (period)
or
Windows key + ; (semicolon)

Display Property

The display property is one of the most important CSS properties because it **controls how elements are displayed and positioned** on a web page.

It tells the browser **how an element should behave in the document layout** whether it should appear inline, as a block, as a flexible box, or be hidden entirely.

Display Type	Description	Behavior / Use Case	Visual Behavior
inline	Displays elements **in a line**, without starting on a new line.	Does **not accept width/height**. Common for , <a>, .	Elements sit **side by side** in a single line.
block	Displays element as a **block**, starting on a new line.	Takes **full width** available and allows **width/height** to be set.	Each element appears **on a new line**.
inline-block	Combines features of **inline** and **block**.	Appears **inline**, but allows **width and height**.	Boxes are **side by side**, but **size-controllable**.
none	**Hides** the element completely (removed from layout).	Element takes **no space** on the page.	The element is **invisible** and **does not occupy space**.
flex	Displays element as a **flex container**.	Allows flexible alignment and distribution of child elements.	Items are **arranged in a row** (or column) with flexible spacing.
grid	Displays element as a **grid container**.	Divides layout into **rows and columns** for advanced control.	Elements are placed in a **grid layout** (rows and columns).

Example:

```
<head>
<style>
.box {
  background-color: lightblue;
  border: 2px solid navy;
  padding: 10px;
  margin: 5px;
  text-align: center;
}
/* INLINE - appear side by side */
.inline div {
  display: inline;
  background-color: yellow;
  border: 1px solid orange;
  padding: 5px;
}
/* BLOCK - start on a new line */
.block span {
  display: block;
```

```
  background-color: lightcoral;
  border: 1px solid red;
  color: white;
  padding: 10px;
  margin-bottom: 5px;
}
/* INLINE-BLOCK - side by side + size controllable */
.inline-block div {
  display: inline-block;
  background-color: lightgreen;
  border: 2px solid green;
  width: 120px;
  height: 60px;
  line-height: 60px;
  margin: 5px;
}
/* NONE - hidden element */
.none p {
  display: none;
}
</style>
</head>
<body>
<h2>CSS Display Property Example</h2>
<!-- INLINE -->
<h3>display: inline</h3>
<div class="inline box">
  <div>Inline 1</div>
  <div>Inline 2</div>
  <div>Inline 3</div>
</div>
<p> These elements appear <strong>side by side</strong>; width and height cannot be set individually.</p>
<!-- BLOCK -->
<h3>display: block</h3>
<div class="block box">
  <span>Block 1</span>
  <span>Block 2</span>
  <span>Block 3</span>
</div>
<p> Each element starts on a <strong>new line</strong> and takes full width.</p>
<!-- INLINE-BLOCK -->
<h3>display: inline-block</h3>
<div class="inline-block box">
  <div>Box 1</div>
  <div>Box 2</div>
  <div>Box 3</div>
</div>
```

```
<p> These elements sit <strong>side by side</strong> but allow <strong>width and height</strong>
adjustments.</p>
<!-- NONE -->
<h3>display: none</h3>
<div class="none box">
  <p>This text is hidden using display:none;</p>
</div>
<p> The hidden text does <strong>not take up any space</strong> on the page.</p>
</body>
```

CSS Display Property Example

display: inline

Inline 1 Inline 2 Inline 3

These elements appear **side by side**; width and height cannot be set individually.

display: block

Block 1

Block 2

Block 3

Each element starts on a **new line** and takes full width.

display: inline-block

Box 1 Box 2 Box 3

These elements sit **side by side** but allow **width and height** adjustments.

display: none

The hidden text does **not take up any space** on the page.

Flexbox property

- ✓ **Flexbox** (short for *Flexible Box Layout*) is a **modern CSS layout model** that provides an efficient way to align, distribute, and space items inside a container even when their size is unknown or dynamic.
- ✓ It's designed to make it easier to create **responsive layouts**, where elements adjust automatically to different screen sizes and available space.

Property	Description	Possible Values	Syntax / Example
display: flex	Defines a flex container to arrange items flexibly.	flex, inline-flex	div { display: flex; }
flex-direction	Defines the direction of flex items.	row (default), row-reverse, column, column-reverse	flex-direction: row;
justify-content	Aligns items horizontally (along the main axis).	flex-start, flex-end, center, space-between, space-around, space-evenly	justify-content: space-between;
align-items	Aligns items vertically (along the cross axis).	stretch (default), flex-start, flex-end, center	align-items: center;
flex-wrap	Determines whether flex items wrap onto multiple lines.	nowrap (default), wrap, wrap-reverse	flex-wrap: wrap;
gap	Defines the space between flex items.	Any CSS length unit (px, em, %)	gap: 15px;
order	Specifies the display order of flex items.	Integer values (0 default, can be positive or negative)	order: 2;
flex *(shorthand)*	Sets how an item grows, shrinks, and defines its base size.	flex: grow shrink basis; e.g., flex: 1 0 100px;	flex: 1; or flex: 1 0 200px;

Flex Example

```
<head>
<style>
.container {
  display: flex;              /* Creates a flex container. Change values and check effects */
  flex-direction: row;        /* Arranges items in a row (default). Change values and check effects */
  justify-content: space-between; /* Spaces items horizontally. Change values and check effects */
  align-items: center;        /* Aligns items vertically in the center. Change values and check effects */
  flex-wrap: wrap;            /* Wraps items to next line if space runs out */
  gap: 15px;                  /* Adds space between items */
  background-color: lightgray;
  padding: 15px;
}
.item {
  background-color: steelblue;
  color: white;
  padding: 20px;
  text-align: center;
  border-radius: 5px;
  width: 200px;
```

```
}
.item:nth-child(2) {
  order: 1;            /* This will move the item to appear after others. Change values and check effects(POSITIVE,NEGATIVE,0) */
/*  flex: 1 1 100px;  */     /* Remove comments from this line to check effect. flex-grow: 1, flex-shrink: 1, flex-basis: 100px */
  background-color: orange;
}
</style>
</head>
<body>
<div class="container">
  <div class="item">Item 1</div>
  <div class="item">Item 2 (Order 1)</div>
  <div class="item">Item 3</div>
</div>
</body>
```

Property	Description	In This Example
display: flex	Turns the container into a flexbox container.	.container becomes a flex container, and its child .item elements become flex items.
flex-direction: row	Arranges flex items horizontally (left to right).	All .item boxes appear side by side.
justify-content: space-between	Distributes items evenly with space between them.	The first and last item stick to edges; middle items get even gaps.
align-items: center	Aligns items vertically along the cross-axis.	Centers items vertically within the container's height.
flex-wrap: wrap	Allows items to wrap to a new line if there's not enough space.	If you resize the browser, items will move to the next row.
gap: 15px	Adds space between flex items.	Creates uniform spacing (15px) between all .items.
order	Defines the visual order of items (default = 0).	order: 1 makes Item 2 appear **after** Items 1 and 3.
flex: 1 1 100px	Shorthand for flex-grow, flex-shrink, and flex-basis.	Item 2 starts at 100px wide, can grow and shrink with available space.

Normal Flow: Item 1 | Item 2 | Item 3
After order: 1 on Item 2: It moves after the other two → Item 1 | Item 3 | Item 2
Resize the browser window: Item 2 (with flex: 1 1 100px) will stretch or shrink smoothly as the container grows or shrinks.

Grid Layout Properties

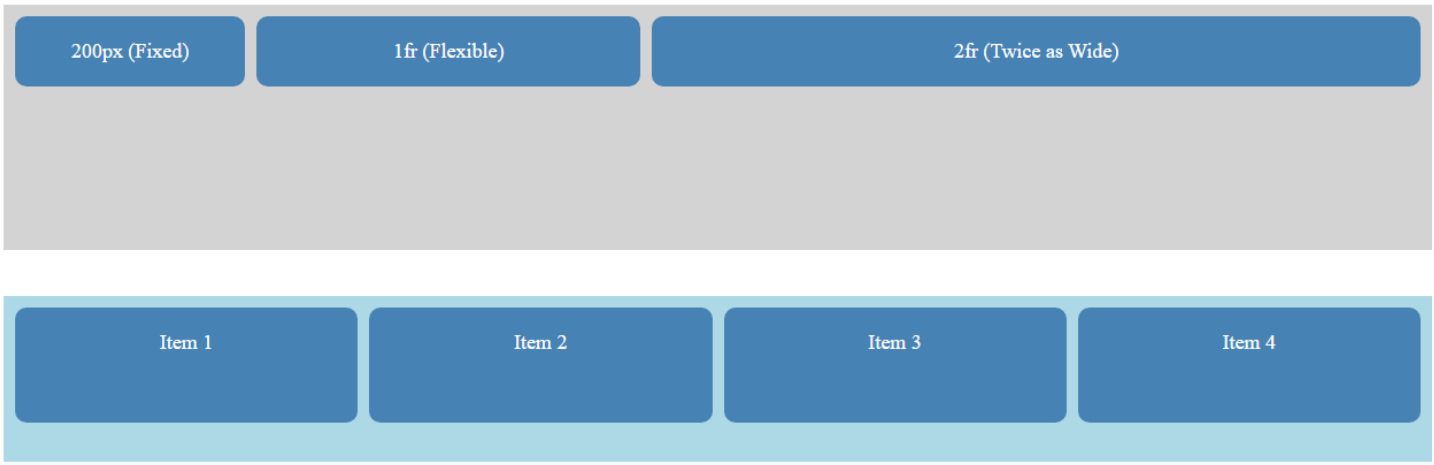
- ✓ **CSS Grid Layout** is a **two-dimensional layout system**. It allows you to arrange content in **rows and columns** easily.
- ✓ It's perfect for designing **web page structures**, **galleries**, **dashboards**, or **complex layouts** without relying on floats or flexbox hacks.

Property	Description	Possible Values	Default Value	Syntax / Example
grid-template-columns	Defines the number and width of columns in a grid layout.	Fixed units (px, em, %)	none	grid-template-columns: 200px 1fr 2fr; The **fr** unit means **"fraction of the available space."**
grid-template-rows	Defines the number and height of rows in a grid layout.	Fixed units (px, em, %),	none	grid-template-rows: 100px auto 100px;
gap (grid-gap)	Defines the space between rows and columns.	Any CSS length (px, em, %)	0	gap: 10px; or gap: 20px 40px;
grid-column	Specifies the horizontal position of an item within the grid columns (start / end lines).	<start-line> / <end-line>	auto	grid-column: 1 / 3; (spans across 2 columns)
grid-row	Specifies the vertical position of an item within the grid rows (start / end lines).	<start-line> / <end-line>	auto	grid-row: 1 / 2; (spans across 1 row)

Example:

```
<head>
<style>
/* ===== Example 1: Fixed + Flexible Columns ===== */
.grid1 {
  display: grid;
  grid-template-columns: 200px 1fr 2fr; /* 3 columns: fixed + flexible */
  grid-template-rows: 1fr 2fr; /* 2 flexible rows (1:2 ratio) */
  gap: 10px;
  background-color: lightgray;
  padding: 10px;
  margin-bottom: 40px;
}
/* ===== Example 2: Equal Columns ===== */
```

```
.grid2 {
  display: grid;
  grid-template-columns: repeat(4, 1fr); /* 4 equal columns */
  grid-template-rows: 100px;
  height:150px;
  gap: 10px;
  background-color: lightblue;
  padding: 10px;
}
.item {
  background-color: steelblue;
  color: white;
  text-align: center;
  padding: 20px;
  border-radius: 10px;
  font-size: 18px;
}
</style>
</head>
<body>
<div class="grid1">
  <div class="item">200px (Fixed)</div>
  <div class="item">1fr (Flexible)</div>
  <div class="item">2fr (Twice as Wide)</div>
</div>
<div class="grid2">
  <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>
```



CSS Positioning Properties

Property	Description	Possible Values	Default Value	Syntax / Example
position	Defines how an element is positioned in the document.	static, relative, absolute, fixed, sticky	**static**	position: absolute;
top	Distance between element and top edge of container.	Any CSS length (px, %, auto)	**auto**	top: 20px;
right	Distance between element and right edge of container.	Any CSS length (px, %, auto)	**auto**	right: 10px;
bottom	Distance between element and bottom edge of container.	Any CSS length (px, %, auto)	**auto**	bottom: 15px;
left	Distance between element and left edge of container.	Any CSS length (px, %, auto)	**auto**	left: 30px;
z-index	Controls stack order of overlapping elements.	Integer (auto, positive or negative values)	**auto**	z-index: 2; (Higher = on top)

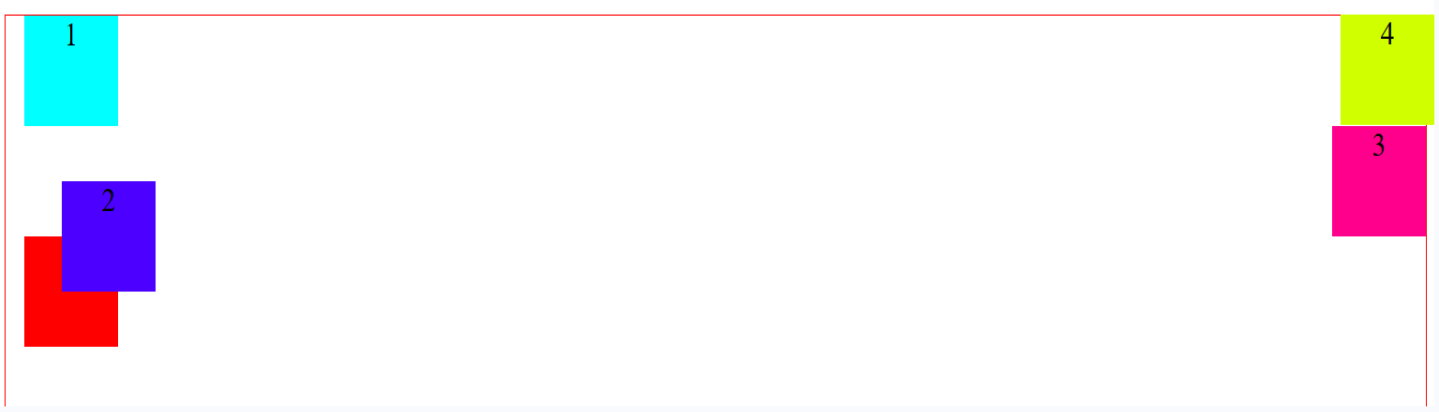
- ✓ **static: Default, element follows normal document flow.**
- ✓ **relative: Moved relative to its normal position.**
- ✓ **absolute: Positioned relative to the nearest positioned ancestor.**
- ✓ **fixed: Stays fixed in place even when scrolling.**
- ✓ **z-index: Controls which element appears on top of others.**

Example to understand position

```
<html>
<head>
  <style>
.container{
  margin-top: 50px;
  height: 800px;
  border: 1px solid red;
  position: relative;
}
.container1{
  margin-top: 30px;
  height: 1000px;
  border: 2px solid black;
}
```

```
.test {
    margin-left: 20px;
    height: 100px;
    width : 100px;
    text-align: center;
    font-size:30px;
}
#test1{
    background-color: aqua;
}
#test2{
    background-color: rgb(76, 0, 255);
    position: relative;
    top: 50px;
    left:40px;
    z-index: 1;
}
#test3{
    background-color: rgb(255, 0, 140);
    position: absolute;
    top: 100px;
    right: 0px;
}
#test4{
    background-color: rgb(208, 255, 0);
    position: fixed;
    top: 50px;
    right: 0;
}
#test5{
    background-color: rgb(255, 0, 0);
    position: sticky;
    top: 0px;
}
    </style>
</head>
<body>
    <div class="container">
        <div class="test" id="test1">1</div>
        <div class="test" id="test2">2</div>
```

```
    <div class="test" id="test3">3</div>
    <div class="test" id="test4">4</div>
    <div class="test" id="test5">5</div>
  </div>
  <div class="container1"></div>
</body>
</html>
```



Selector	Position Type	Behavior / Explanation
#test1	static (default)	No positioning applied follows normal document flow.
#test2	relative	Moved **50px down** and **40px right** *from its normal position*. The original space is still reserved.
#test3	absolute	Placed **100px from top** and **0px from right** of the **nearest positioned ancestor** (.container has position: relative, so that's the reference).
#test4	fixed	Positioned **50px from top and right of the viewport**, stays **fixed even when scrolling**.
#test5	sticky	Acts like a normal element until the page is scrolled then it **sticks to the top (0px)** of its container.