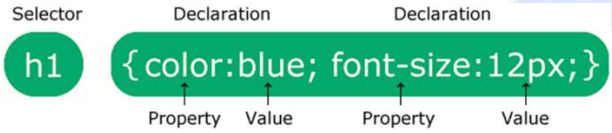
CSS NOTES

❖ What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

CSS Syntax



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

❖ The 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.

```
.center {
  text-align: center;
  color: red;
}
```

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

```
p.center {
  text-align: center;
  color: red;
}
```

HTML elements can also refer to more than one class.

```
<p class="center large">This paragraph refers to two classes.</p>
```

❖ The CSS Universal Selector

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

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

❖ The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

p {
  text-align: center;
  color: red;
}


h1, h2, p {
  text-align: center;
  color: red;
}
```

❖ Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

❖ External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

❖ Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

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

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

❖ CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date. Comments are ignored by browsers.

A CSS comment is placed inside the <style> element, and starts with /* and ends with */:

```
p {
  color: red;  /* Set text color to red */
}
```

❖ CSS Background Color

You can set the background color for HTML elements:

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

❖ CSS Text Color

You can set the color of text:

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

❖ CSS Border Color

You can set the color of borders:

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
```

```
p {
  border-left: 6px solid red;
}
```

```
p {
  border-bottom: 6px solid red;
}
```

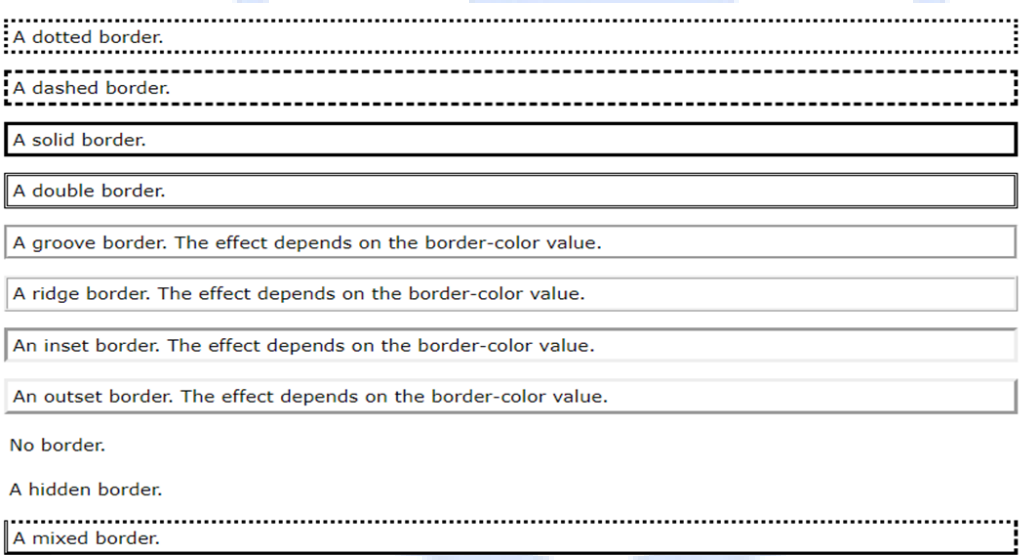
❖ CSS Border Style

The border-style property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border

- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border



❖ CSS Rounded Borders

The border-radius property is used to add rounded borders to an element:

```
p {
  border: 2px solid red;
  border-radius: 5px;
}
```

❖ CSS Color Values

In CSS, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values:

Same as color name "Tomato":

❖ RGB Value

In CSS, a color can be specified as an RGB value, using this formula:

rgb(red, green, blue)

Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

To display black, set all color parameters to 0, like this: rgb(0, 0, 0).

To display white, set all color parameters to 255, like this: rgb(255, 255, 255).

Experiment by mixing the RGB values below:

rgba(red, green, blue, alpha)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

❖ HEX Value

In CSS, a color can be specified using a hexadecimal value in the form:

#rrggbb

Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, #ff0000 is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

To display black, set all values to 00, like this: #000000. To display white, set all values to ff, like this: #ffffff.

- CSS Backgrounds

  The CSS background properties are used to add background effects for elements.

- CSS background-color

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

- CSS background-image

  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.

```
body {
  background-image: url("paper.gif");
}
```

❖ CSS 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, like this:

```
body {
  background-image: url(“COW.PNG");
  background-repeat: no-repeat;
}
```

❖ CSS background-position

The background-position property is used to specify the position of the background image.

```
body {
    background-image: url("img_tree.png");
    background-repeat: no-repeat;
```

```
  background-position: right top;
}
```

❖ CSS background-attachment

The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):

```
   body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
/*OR*/
  background-attachment: scroll;

}
```

❖ CSS Margins

The CSS margin properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

margin-top

margin-right

margin-bottom

margin-left

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}

p {
  margin: 25px 50px 75px 100px;
}

p {
  margin: 25px 50px;
}

p {
  margin: 25px;
}
```

❖ The CSS Box Model

In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins. The image below illustrates the box model:

❖ CSS fonts

CSS fonts are used to control the appearance of text on a web page. Understanding CSS font properties allows you to specify various aspects of text styling, including font family, size, weight, style, and more. Here's an explanation of the key CSS font properties:

1. font-family

The font-family property specifies the typeface that should be used for the text. You can list multiple font families as a fallback system, ensuring that if the first choice isn't available, the browser will try the next one.

body { font-family: "Arial", "Helvetica", sans-serif; }

Generic Font Families

In CSS there are five generic font families:

Serif fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.

Sans-serif fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.

Monospace fonts - here all the letters have the same fixed width. They create a mechanical look. Cursive fonts imitate human handwriting. Fantasy fonts are decorative/playful fonts.

CSS fonts are used to control the appearance of text on a web page. Understanding CSS font properties allows you to specify various aspects of text styling, including font family, size, weight, style, and more. Here's an explanation of the key CSS font properties:

1. font-family

The font-family property specifies the typeface that should be used for the text. You can list multiple font families as a fallback system, ensuring that if the first choice isn't available, the browser will try the next one.

css

body { font-family: "Arial", "Helvetica", sans-serif; }

In this example, the browser will use Arial if available. If not, it will try Helvetica, and if neither is available, it will use a generic sans-serif font.

2. font-size

The font-size property sets the size of the text. It can be defined using different units like pixels (px), em, rem, percentages, and more.

css

p { font-size: 16px; }

3. font-weight

The font-weight property sets the thickness of the font. It can take numeric values (100 to 900) or keywords (normal, bold, bolder, lighter).

css

h1 { font-weight: bold; } h2 { font-weight: 700; }

4. font-style

The font-style property specifies whether the text is normal, italic, or oblique.

css

em { font-style: italic; }

5. line-height

The line-height property sets the amount of space between lines of text.

css

p { line-height: 1.5; }

6. letter-spacing

The letter-spacing property controls the space between characters.

css

p { letter-spacing: 0.05em; }

7. text-transform

Though not directly a font property, text-transform can affect the appearance of text by changing the case.

❖ Using Font Awesome

Font Awesome is a popular icon library that provides a wide range of icons.

Step 1: Include Font Awesome

Add the Font Awesome CSS file to the <head> section of your HTML document:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Font Awesome Icons</title>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">

</head> <body>

<!-- Content goes here -->

</body>

</html>

Step 2: Add Icons

Use the or tag with the appropriate classes to add icons to your page:

<body> <h1>Using Font Awesome Icons</h1>   </body>

2. Using Inline SVG Icons

You can also use inline SVG icons, which provide great flexibility and customization.

Step 1: Add SVG Code

Include the SVG code directly within your HTML document:

html

<body>

<h1>Using Inline SVG Icons</h1>

<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-camera" viewBox="0 0 16 16">

<path d="M10.5 5a.5.5 0 0 1 .5.5v.5h1a.5.5 0 0 1 0 1h-1v1a.5.5 0 0 1-1 0v-1H9a.5.5 0 0 1 0-1h1V5.5a.5.5 0 0 1 .5-.5zm-3-2a1.5 1.5 0 0 1 1.493 1.356L9 4h1a1 1 0 0 1 .117 1.993L10 6H6a1 1 0 0 1-.117-1.993L6 4h1a1.5 1.5 0 0 1 .5-.356L7.5 3z"/>

<path d="M4.5 3A1.5 1.5 0 0 0 3 4.5v7A1.5 1.5 0 0 0 4.5 13h7A1.5 1.5 0 0 0 13 11.5v-7A1.5 1.5 0 0 0 11.5 3h-7zM4 4.5a.5.5 0 0 1 .5-.5h7a.5.5 0 0 1 .5.5v7a.5.5 0 0 1-.5.5h-7a.5.5 0 0 1-.5-.5v-7z"/>

</svg>

</body>

❖ The display property in CSS

The display property in CSS is a fundamental part of layout design, determining how elements are displayed on the web page. It controls the type of box an element generates, influencing the flow and positioning of elements within the document. Here are the primary values of the display property and their effects:

1. display: block

Description: The element generates a block-level box.

Characteristics:

Starts on a new line.

Takes up the full width available (if width is not set).

Height and width properties can be set.

Margins and padding are respected.

css

```
div { display: block; }
```

2. display: inline

Description: The element generates an inline-level box.

Characteristics:

Does not start on a new line.

Takes up only as much width as necessary.

Height and width properties have no effect.

Margins and padding are applied but do not affect the layout as in block elements.

css

```
span { display: inline; }
```

3. display: inline-block

Description: The element generates a block-level box that flows with inline content.

Characteristics:

Does not start on a new line.

Takes up only as much width as necessary.

Height and width properties can be set.

Margins and padding are respected.

css

button { display: inline-block; }

4. display: none

Description: The element is not displayed and does not take up any space in the layout.

Characteristics:

Element is completely removed from the document flow.

No space is allocated for it.

css.hidden { display: none; }

❖ CSS Position's:

1. position: static

Description: This is the default value. Elements are positioned according to the normal document flow.

Characteristics:

- Not affected by the top, bottom, left, or right properties.

css

.static-element { position: static; }

2. position: relative

Description: The element is positioned relative to its normal position.

Characteristics:

- Can be moved using the top, bottom, left, and right properties without affecting other elements' positions.
- Leaves space in the document where it would normally be.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

css.relative-element { position: relative; top: 10px; /* Moves the element 10px down */ left: 20px; /* Moves the element 20px to the right */ }

3. position: absolute

Description: The element is positioned relative to its nearest positioned ancestor (an ancestor with a position other than static). If no such ancestor exists, it is positioned relative to the initial containing block (usually the document body).

Characteristics:

- Removed from the normal document flow.
- Can overlap other elements.
- Positioned using the top, bottom, left, and right properties.

css

.absolute-element { position: absolute; top: 50px; /* Moves the element 50px from the top of the positioned ancestor */ right: 30px; /* Moves the element 30px from the right of the positioned ancestor */ }

4. position: fixed

Description: The element is positioned relative to the browser window, regardless of scrolling.

Characteristics:

- Stays in the same position even when the page is scrolled.
- Positioned using the top, bottom, left, and right properties.

css

.fixed-element { position: fixed; top: 0; /* Sticks the element to the top of the viewport */ left: 0; /* Sticks the element to the left of the viewport */ }

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

5. position: sticky

Description: The element is treated as relative until it crosses a specified threshold (defined by top, bottom, left, or right). Then it is treated as fixed.

Characteristics:

- Useful for creating elements that stick to the viewport (like headers) when scrolling past them.

css

.sticky-element { position: sticky; top: 0; /* Sticks the element to the top of the viewport when scrolling past it */ }

6. z-index

Description: The z-index property specifies the stack order of positioned elements (elements with position values other than static). An element with a higher z-index is always in front of an element with a lower z-index.

css

.high-z-index { position: absolute; z-index: 10; /* Higher stack order */ }
.low-z-index { position: absolute; z-index: 1; /* Lower stack order */ }

❖ CSS Flexbox

It is one dimentional layout method for arranging items in row.

Flexbox, short for the Flexible Box Layout, is a CSS layout module designed to help you create flexible and efficient layouts. It simplifies the alignment and distribution of space among items in a container, even when their size is unknown or dynamic. Flexbox is particularly well-suited for responsive web design and one-dimensional layouts (either row or column).

Key Concepts and Terminology

Flex Container: The parent element that holds flex items. It is defined by setting the display property to flex or inline-flex.

Flex Items: The child elements of a flex container.

display: Specifies the element as a flex container.

display: flex; (block-level flex container)

display: inline-flex; (inline-level flex container)

flex-direction: Defines the direction of the main axis (row or column).on this it desides main and cross axis of content

row (default): Items are placed horizontally. Main axis horixontal left to right

row-reverse: Items are placed horizontally in reverse order. . Main axis horixontal right to left

column: Items are placed vertically. Main axis vertically top to bottom

column-reverse: Items are placed vertically in reverse order. Main axis vertically bottom to top

justify-content: Aligns flex items along the main axis. Teels how the browser distribute space between or around main axis

flex-start (default): Items are packed toward the start.

flex-end: Items are packed toward the end.

center: Items are centered.

space-between: Items are evenly distributed with space between them.

space-around: Items are evenly distributed with space around them.

space-evenly: Items are evenly distributed with equal space around them.

flex-wrap: Determines whether flex items are forced into a single line or can wrap onto multiple lines.

nowrap (default): All items will be on one line.

wrap: Items will wrap onto multiple lines.

wrap-reverse: Items will wrap onto multiple lines in reverse order.

align-items: Aligns flex items along the cross axis.

stretch (default): Items stretch to fill the container.

flex-start: Items are aligned toward the start.

flex-end: Items are aligned toward the end.

center: Items are centered.

align-content: Aligns the flex lines (when there are multiple lines) along the cross axis.

stretch (default): Lines stretch to fill the container.

flex-start: Lines are packed toward the start.

flex-end: Lines are packed toward the end.

center: Lines are centered.

space-between: Lines are evenly distributed with space between them.

space-around: Lines are evenly distributed with space around them.

```
<!DOCTYPE html>
<html>
<head>
<style>
 .container
 { display: flex; flex-direction: row; justify-content: space-around; align-items: center; height: 200px; border: 2px solid black; }
.item { background-color: lightblue; padding: 20px; margin: 10px; }
 </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>
```

❖ CSS :hover Selector

```
/* unvisited link */
a:link {
  color: green;
}

/* visited link */
a:visited {
  color: green;
}
```

```
/* mouse over link */
a:hover {
  color: red;
}

/* selected link */
a:active {
  color: yellow;
}
```

❖ CSS, a transition

Enables you to define the transition between 2 states of an element

In CSS, a transition allows you to change property values smoothly (over a given duration) instead of having them change abruptly. This feature is useful for animating changes in CSS properties such as color, size, position, and more, providing a more polished and interactive user experience.

Ex there is box div wanted to change in circle form when hover

```
div{
Height:200px; width:200px;
}
.box { width: 100px;
 height: 100px;
 background-color: blue;
transition:2s;
 }
 .box:hover {
 background-color: red;
 width: 200px;
}
```

- **Transition shortend**

transition: background-color 0.5s ease, width 0.5s ease;

Property name :background color;

Duration:2s;

Timimg function:ease in out;

Delay:0.2s

transition-property: The name of the CSS property to which the transition is applied.

transition-duration: The duration of the transition.

transition-timing-function: The timing function that defines the speed curve of the transition. Check on MDN

transition-delay: The delay before the transition starts

```
<!DOCTYPE html>
 <html>
 <head>
<style>
 .box { width: 100px;
 height: 100px;
 background-color: blue;
transition: margin-top 0.5s ease-in-out 0.2s, }
 .box:hover
{margin-top:10em;


}
```

```
</style>
</head>
 <body>
 <div class="box"></div>
 </body>
</html>
```

❖ Css transform

The property let you rotate,scale, skew,translate an elment.

1. rotate(45deg): Rotates the box 45 degrees clockwise.

- Transform:rotate(45deg)

```
.box{
        height: 200px;
        width: 200px;
        border: 1px solid black;
        background-color: red;
       }

      div:hover{
        transform: rotate(30deg);
        background-color: yellow;
       }

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

2. Scale.: Used to scalup or scale down element with referance of x axis and y axis

- Transform:SCALE(0.5);
- Transform:SCALE(0.5, 2);

```
.box{
    height: 200px;
    width: 200px;
    border: 1px solid black;
    background-color: red;
   }

  div:hover{
    transform: scale(5);
    background-color: yellow;
   }
<body>
<div class="box"></div>
</body>
```

3. Translate .: Used to move element with referance of x axis and y axis

- Transform:translate (50px, 50px);
- Transform:translateX(20px);
- Transform:translateY(20px);

```
.box{
    height: 200px;
    width: 200px;
```

```
        border: 1px solid black;
        background-color: red;
       }

       div:hover{
        transform: translate(50px);
        background-color: yellow;
       }
<body>
<div class="box"></div>
</body>
```

3. Skew

- Transform: Skew (20deg);
- Transform: Skew (-20deg);
- Transform: Skew (90deg);

```
  .box{
        height: 200px;
        width: 200px;
        border: 1px solid black;
        background-color: red;
       }

       div:hover{
        transform: translate(50px);
        background-color: yellow;
       }
<body>
<div class="box"></div>
</body>
```

3. Box Shadow

Box-shadow:2px 2px 10px green;

2px: X offset; 2px: Y offset; 10px=blur radius green=color

❖ Media Query

Help to create responsive website

```
@media(width:400px){
Div{
Background-color:red;
}
}

@media(min-width:400px){
Div{
Background-color:red;
}
}

@media(max-width:400px){
Div{
Background-color:red;
}
}
```

❖ A descendant selector in CSS

A descendant selector in CSS is used to select elements that are descendants of another element. In CSS, a descendant is any element that is nested within another element, whether it is a direct child or a more deeply nested child.

Here's how a descendant selector works:

```
<div class="container">
    <p>This is a paragraph.</p>
    <div class="nested">
        <p>This is a nested paragraph.</p>
    </div>
</div>

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

Explanation :

.container is the ancestor element.

p is the descendant element.

The space between .container and p indicates that p is a descendant (at any level) of .container.

Result

In the example above, the CSS rule will apply to both paragraphs:

This is a paragraph.

This is a nested paragraph.

Both paragraphs will have their text color changed to blue because they are both descendants of the element.

- **Key Points**

Direct or Indirect Descendants: The descendant selector targets all descendants, whether they are direct children, grandchildren, or deeper nested elements.

Whitespace: The space between the ancestor and descendant selectors is crucial and indicates the relationship.

❖ Adjacent sibling combinators

Adjacent sibling combinators in CSS allow you to select an element that is immediately preceded by a specific element. This means it targets the immediate sibling that comes right after another element.

Syntax:-

element1 + element2 { /* styles */ }

Ex:-

<h1>Title</h1>

This paragraph follows the title.

This paragraph is another paragraph and is not selected by the adjacent sibling combinator.

Css:- h1 + p { color: blue; }

Result

The first following the <h1> will have its text color changed to blue.

The second will not be affected because it does not immediately follow an <h1>

- **Key Points**

Immediate Sibling: The adjacent sibling combinator only selects the element that directly follows the specified element. It does not apply to elements that are further away.

Single Target: Only the first immediate sibling is targeted. Subsequent siblings are not affected.

❖ The child combinator

The child combinator in CSS is used to select elements that are direct children of a specified element. This means it targets only the immediate children, not any deeper nested descendants

Syntax:- parent > child { /* styles */ }

parent is the parent element.

child is the direct child element to be styled.

The > sign is the child combinator.

Ex:-

```
<div class="container">
    <p>This is a paragraph.</p>
    <div class="nested">
    <p>This is a nested paragraph.</p>
    </div>
</div>
```

CSS:- .container > p { color: blue; }

The This is a paragraph. will have its text color changed to blue because it is a direct child of the .

The This is a nested paragraph. inside will not be affected because it is not a direct child of the .

Direct Child Only: The child combinator targets only the immediate children of the specified parent element. It does not apply to grandchildren or deeper descendants.

Specificity: Using the child combinator increases specificity, ensuring that styles are applied only to elements that are direct children, not more deeply nested elements.

❖ attribute selector

An attribute selector in CSS allows you to select elements based on the presence or value of their attributes. This can be useful for styling elements dynamically without needing to add specific classes or IDs

Syntax:- element[attribute] { /* styles */ }

<input type="text" placeholder="Enter your name">

<input type="email" placeholder="Enter your email">

<input type="submit" value="Submit">

Input

{ padding: 10px; margin: 5px; } /* Style only text input elements */

input[type="text"]

{ border: 2px solid blue; } /* Style only email input elements */

input[type="email"]

{ border: 2px solid green; } /* Style only submit button */

input[type="submit"]

{ background-color: yellow; border: none; padding: 10px 20px; }

❖ CSS Styling Instructions

1. **Main List Items**:
    - Set the font size to 20px.
    - Set the font weight to bold.
    - Set the text color to #333.
2. **Sub List Items**:
    - Set the font size to 18px.
    - Set the text color to #666.
    - Add a left margin of 20px.
3. **Sub Sub List Items**:
    - Set the font size to 16px.
    - Set the text color to #999.
    - Add a left margin of 40px.