Understanding CSS

1. Introduction

- It is a style sheet language used to define the presentation and layout of web pages.
- It enhances HTML by allowing developers to separate content from design.
- Enabling more flexible and consistent formatting.
- It is a rule-based language used to apply styles (like colors, fonts, spacing) to HTML elements.
- The style definitions are normally saved in external .css files. With an external style sheet file.
- You can change the look of an entire website by changing just one file

2. CSS Syntax

- CSS works by associating rules with HTML elements.
- A rule consists of a selector and a declaration block.
- CSS syntax contains the following elements:
    - Selector: Targets the HTML element(s) to style (e.g., h1, .class, #id).
    - Property: Specifies the style attribute (e.g., color, margin).
    - Value: Defines the style (e.g., red, 10px)

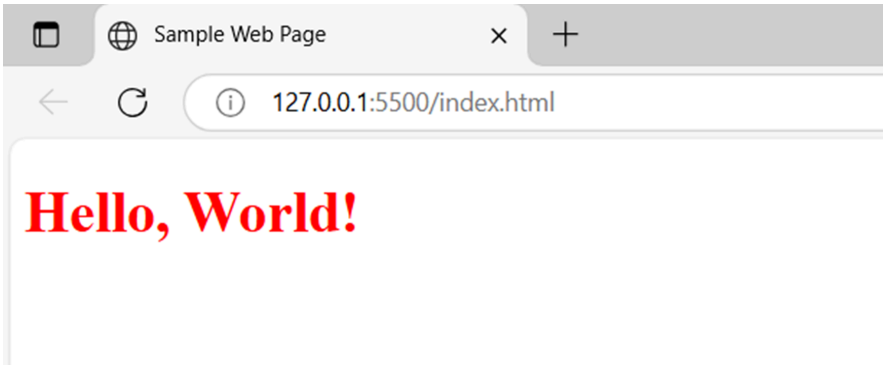
```
h1 {
        color: blue;
        font-size: 12px;
}
```

3. Types of CSS

- All the styles in a page will "cascade" into a new virtual style sheet by the following rules, where the first one has the highest priority

1. Inline CSS
    - Styles are applied directly to individual HTML elements.
    - Uses the style attribute.
    - Useful for quick, one-off styles or testing but is not recommended for large-scale projects due to its limitations in scalability and maintainability.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Web Page</title>
    </head>
    <body>
        <h1 style="color: red">Hello, World!</h1>
    </body>
</html>
```



Advantages

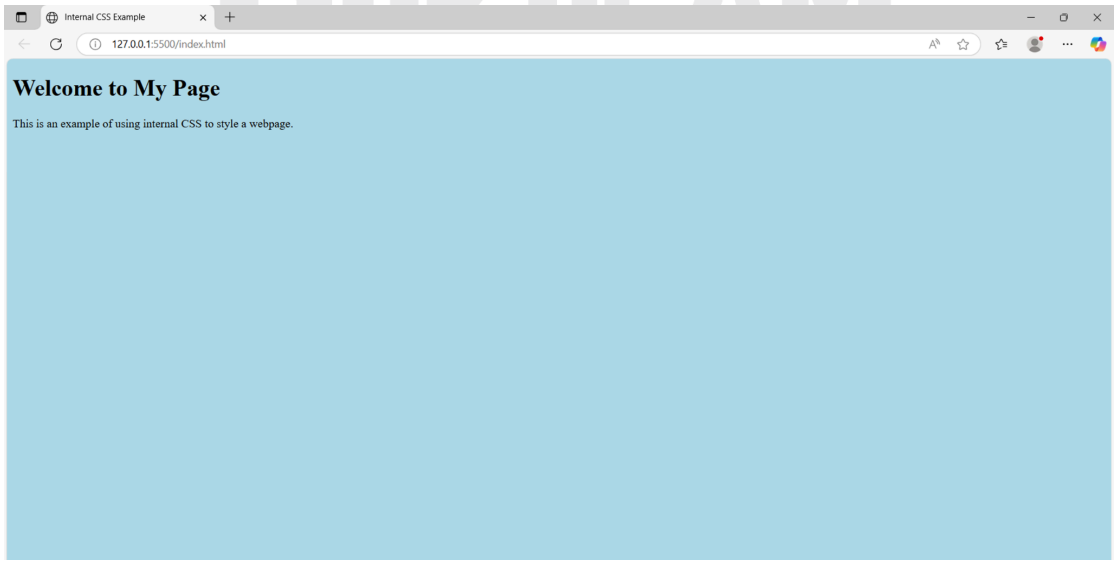
- Quick and Easy: Great for applying quick fixes or testing styles during development.
- Overrides Other Styles: Useful when you need to override external or internal styles for specific elements.
- No Additional File: Does not require a separate CSS file, which can simplify small projects.

Disadvantages

- Poor Maintainability: Inline CSS makes the HTML code messy and difficult to manage for larger projects.
- Limited Reusability: Styles cannot be reused, leading to redundancy and inefficiency.
- Performance Issues: Increases HTML file size and reduces performance, as styles are not cached like external CSS.
- Low Readability: Hard to read and debug when scattered throughout the HTML.

2. Internal CSS
    - Defining styles within the <style> tag in the <head> section of an HTML document.
    - These styles apply to all elements in the same HTML file, offering better organization compared to inline CSS while maintaining control within a single file.
    - Internal CSS is written inside a <style> tag within the <head> section of the HTML document.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Internal CSS Example</title>
        <style>
            body { background-color: lightblue;}
        </style>
    </head>
    <body>
        <h1>Welcome to My Page</h1>
        <p>This is an example of using internal CSS to style a webpage.
        </p>
    </body>
</html>
```



Advantages

- File Consolidation: Styles are included in the same file as the HTML, making it easy to share and distribute a single file.
- Scope: Styles affect only the document in which they are defined, avoiding unintentional application across multiple pages.
- Ease of Debugging: No need to switch between files to modify styles.

Disadvantages

- Limited Reusability: Styles cannot be reused across multiple files, leading to redundancy.
- Increased File Size: Mixing styles and content can make the HTML file larger and harder to maintain.
- Performance: External style sheets are cached, but internal CSS is processed each time the page loads, potentially slowing down rendering.

3. External CSS

- Defines CSS rules in a separate .css file
- Links it to one or more HTML documents.
- This approach separates content (HTML) from design (CSS),
- Improving maintainability and reusability across multiple web pages.
- To use an external CSS file, you link it to your HTML document using the <link> tag within the <head> section.

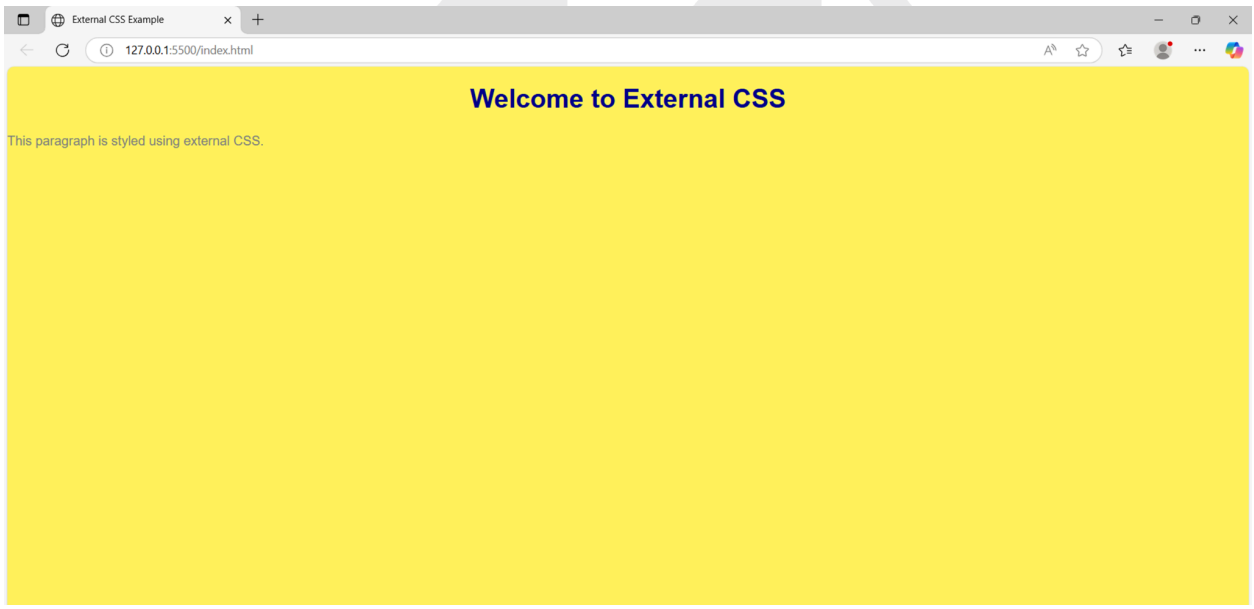
index.html

```
<!DOCTYPE html>
<html >
    <head>
        <title>External CSS Example</title>
        <link rel="stylesheet" href="styles.css" />
    </head>
    <body>
        <h1>Welcome to External CSS</h1>
        <p>This paragraph is styled using external CSS.
        </p>
    </body>
</html>
```


style.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #f9f9f9;
    margin: 0;
    padding: 0;
}
h1 {
```

```
    color: darkblue;
    text-align: center;
}
p {
    color: gray;
    font-size: 16px;
    line-height: 1.5;
}
```



Advantages

- Reusability: One CSS file can style multiple HTML pages, reducing redundancy.
- Better Maintainability: Changes made to the CSS file automatically reflect across all linked HTML files.
- Clean Code: Separates structure (HTML) from design (CSS), enhancing readability.

- Caching: Browsers cache external CSS files, improving performance for subsequent visits.

Disadvantages

- Dependency: The webpage relies on the external file, and styles won't load if the file is missing or inaccessible.
- Latency: Additional HTTP requests to fetch the CSS file may increase initial page load time.
- Complexity: Requires managing multiple files, which may be challenging for beginners.

4. CSS selectors

- CSS selectors are patterns or rules used to select and style specific HTML elements on a webpage.
- By targeting elements, attributes, classes, or IDs, selectors enable precise styling.

1. Type/ Element Selector

- Used to target all elements of a specific type, such as <p>, <h1>, , etc.
- It applies styles to every instance of the specified element on the webpage.
- The type selector targets elements by their tag name and applies styles to all occurrences of the specified tag in the document.
  Syntax: .className { property: value; }

2. Class Selector

- Used to target HTML elements that have a specific class attribute.
- Allows for more specific styling than type selectors and is often used for grouping styles that apply to multiple elements.
- A class can be applied to multiple elements.
- Elements can have multiple classes.
- Class selectors offer higher specificity compared to type selectors. Classes can be used on multiple elements, reducing code duplication.

  Syntax: .className { property: value; }

  css

  ```
  <style>
      .highlight {
          background-color: yellow;
      }
  </style>
  ```

  html

  ```
  <p class="highlight">This paragraph is highlighted.</p>
  ```

  Styling Elements with a Single Class

  css

  ```
  . button {
      background-color: blue;
      color: white;
      padding: 10px 20px;
      border-radius: 5px;
  }
  ```

html

```
<button class="button">Click Me</button>
```

Styling Multiple Elements with the Same Class

css

```
.highlight {
        background-color: yellow;
        font-weight: bold;
}
```

html

```
<p class="highlight">This text is highlighted.</p>
<p class="highlight">This text is also highlighted.</p>
```

Combining Multiple Classes

css

```
.text-large {
        font-size: 20px;
}
.text-bold {
        font-weight: bold;
}
```

html

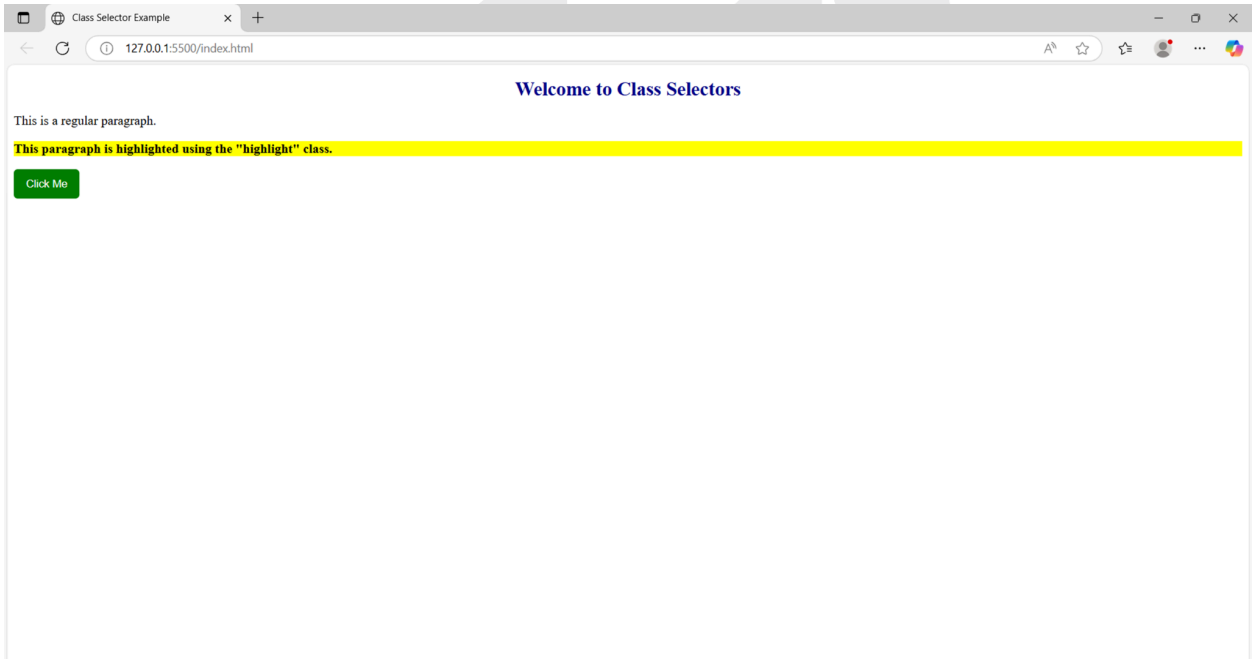
```
<p class="text-large text-bold">This text is large and
bold.</p>
```

index.html

```
<style>
        .title {
                font-size: 24px;
                color: darkblue;
                text-align: center;
         }
          .highlight {
                background-color: yellow;
                font-weight: bold;
         }
         .button {
                background-color: green;
                color: white;
                padding: 10px 15px;
                border: none;
                border-radius: 5px;
                cursor: pointer;
          }
          .button:hover {
                background-color: darkgreen;
         }
</style>
```

style.css

```
<body>
        <h1 class="title">Welcome to Class Selectors</h1>
        <p>This is a regular paragraph.</p>
        <p class="highlight">
                This paragraph is highlighted using   the "highlight" class.
        </p>
    <button class="button">Click Me</button>
</body>
```



3. Id Selector

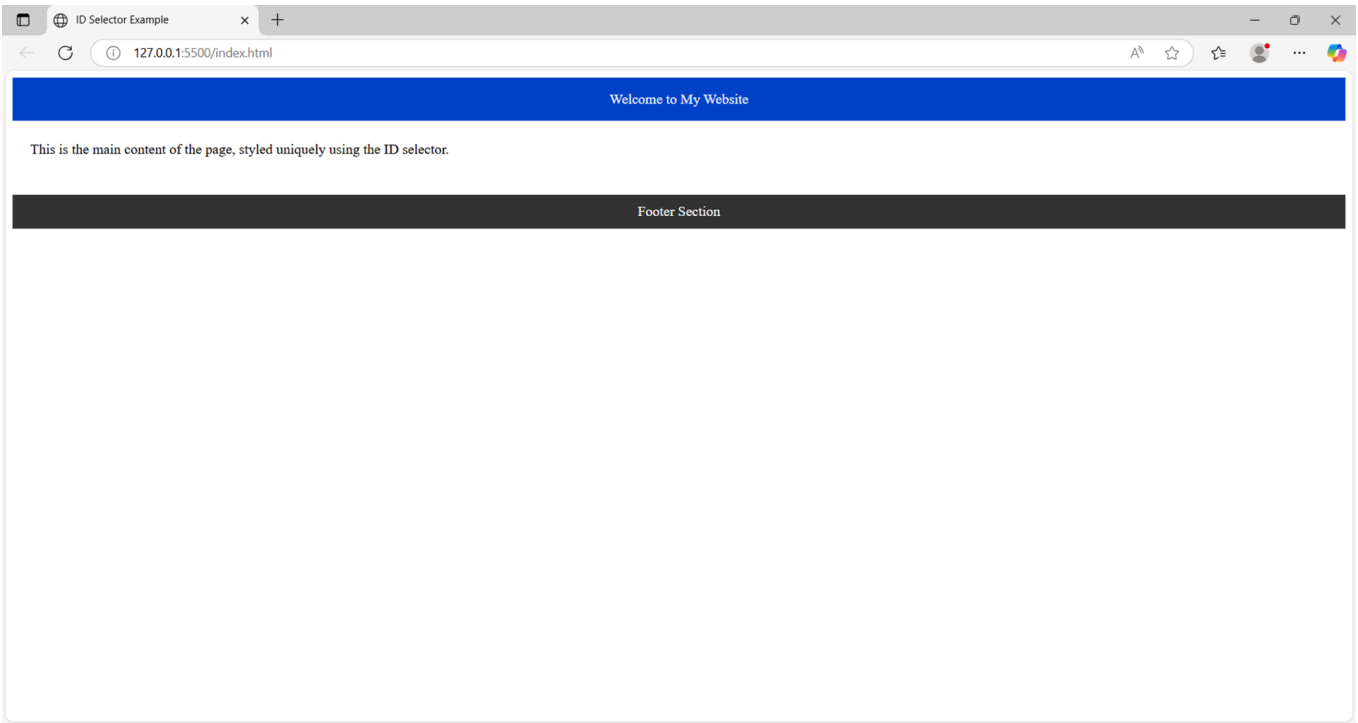
- Used to target a specific element with a unique id attribute.
- Highly specific and applies styles to one element per page.
- Targets a single, unique element by its id.
- Offers higher specificity than class or type selectors and is useful for unique elements, such as headers, footers, or specific sections.
- Best suited for styling one-off elements, offering precision and high specificity.
- However, its overuse can lead to challenges in CSS maintenance and scalability.

Syntax: #idName { property: value; }

index.html

```
<style>
        #header {
                background-color: #0044cc;
                color: white;
                text-align: center;
                padding: 15px;
        }
        #footer {
                background-color: #333;
                color: white;
                text-align: center;
                padding: 10px;
                margin-top: 20px;
        }
        #main {
                padding: 20px;
                font-size: 16px;
                line-height: 1.5;
        }
</style>
```

```
<body>
    <div id="header">Welcome to My Website</div>
    <div id="main">
        This is the main content of the page,  styled uniquely
        using the ID selector.
    </div>
    <div id="footer">Footer Section</div>
</body>
```



4. Grouping of Selectors

- Allows you to apply the same styles to multiple elements without repeating the style rules.
- This is done by separating the selectors with a comma (,).

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

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Web Page</title>
        <link rel="stylesheet" href="styles.css">
    </head>
    <body>
        <h1>Grouping Selectors Example</h1>
        <h2>Main Heading</h2>
        <h3>Subheading</h3>
        <p>This is a paragraph.</p>
        <div>This is a div element.</div>
    </body>
</html>
```

style.css

```
h1,h2,p,div {
    font-family: Arial, sans-serif;
    color: blue;
    margin: 10px,0;
}
```

Benefits of Grouping Selectors

- Reduces Redundancy: Avoids repeating the same style rules for multiple selectors.
- Simplifies Code: Makes CSS more readable and easier to maintain.

- Improves Performance: A single rule is applied to multiple elements, leading to less repetitive CSS parsing.

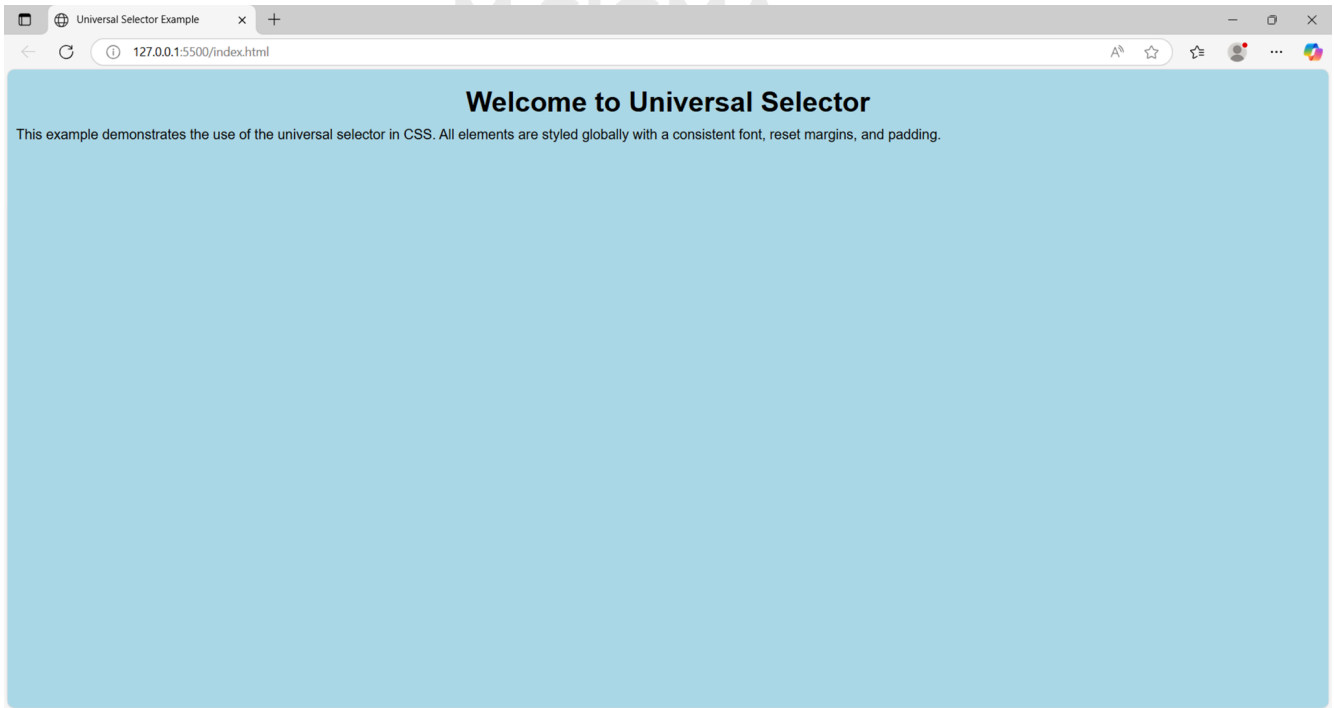
5. Universal Selectors

- The universal selector (*) is a basic CSS selector
- Targets all elements on a webpage, regardless of their type, class, or ID.
- It is often used to apply general styles across the entire document.
- The universal selector targets every element within the document and is useful for applying consistent styles like resetting margins, padding, or box-sizing.

  Syntax: { property: value; }

```
<style>
    * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
        font-family: Arial, sans-serif;
    }
    body {
        background-color: lightblue;
    }
    h1 {
        text-align: center;
        margin-top: 20px;
```

```
        }
        p {
            margin: 10px;
            text-align: justify;
        }
    </style>
    <body>
        <h1>Welcome to Universal Selector</h1>
        <p>
            This example demonstrates the use of the universal selector in CSS. All elements are styled globally with a consistent font, reset margins, and padding.
        </p>
    </body>
```



6. Pseudo Classes

- Pseudo-classes in CSS are keywords added to selectors that specify a special state of the selected elements.
- They are typically used to style elements based on their state, position, or interaction.

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

Commonly used pseudo-classes

- :hover - Applies styles when the user hovers over an element.

```
a:hover {
    color: red;
}
```

- :focus - Applies styles to an element when it is focused (e.g., clicked or tabbed into).

```
input:focus {
    border: 2px solid blue;
}
```

- :active - Applies styles to an element when it is being clicked.

```
button:active {
    background-color: green;
}
```

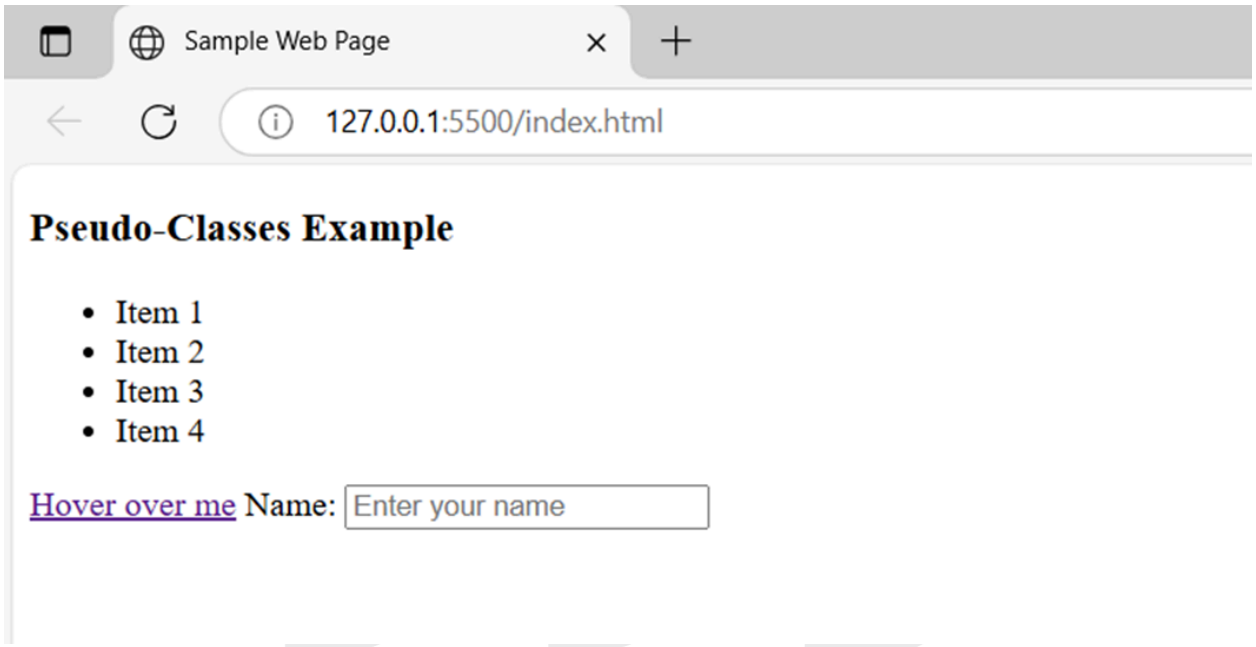
index.html

```
<head>
    <title>Sample Web Page</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h3>Pseudo-Classes Example</h3>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
        <li>Item 4</li>
    </ul>
    <a href="#">Hover over me</a>
    <label for="name">Name:</label>
    <input type="text" id="name" name="name"
    placeholder="Enter your name">
</body>
```

style.css

```
a:hover {
    color: red;
}
```

```
input:focus {
        border: 2px solid blue;
}
```



Advantages of Pseudo-Classes

- Dynamic Styling: Reacts to user interactions (e.g., :hover, :focus).
- Target Specific States: Styles based on element states or positions.
- Efficient: Reduces the need for JavaScript for simple dynamic behaviors.

6. Cascade in CSS

- Cascade refers to the priority system CSS uses to determine which styles to apply when multiple conflicting rules are present for the same element.
- When multiple factors are considered, the same engine determines which style applies in the following order.

- Inline style
- !important rules
- ID selectors
- Element and pseudo-elements
- Source order

7. CSS Property-value Forms

- In CSS, properties are categorized based on their functions, and they control different aspects of the styling of web elements.
- Below are the major categories of properties with examples of their usage in property-value pairs.
- Property values in CSS can appear in many forms.
- CSS provides a wide variety of value types to define the styles for elements.

a) Property-values

1. Keywords

- Predefined values that CSS provides.
- Example: none, auto, inherit, initial.

display: block;

position: absolute;

2. Length Values

- Specify the size of elements.
- Can be absolute or relative units.
- Absolute units: px, cm, mm, in, pt, pc.

- Relative units: em, rem, %, vw, vh, vmin, vmax, ch, ex.

  width: 50px;

  Font-size: 1.5em;

3. Color Values

- Define colors in various formats:
- Named colors: red, blue, green.
- Hexadecimal: #ff5733.
- RGB/RGBA: rgb(255, 87, 51), rgba(0, 0, 0, 0.7).
- HSL/HSLA: hsl(20, 100%, 60%), hsla(20, 100%, 60%, 0.6).

  background-color: #ff5733;

  color: rgba(0, 0, 0, 0.7);

4. Percentage Values

- Often used for dimensions, positioning, and gradients.

  width: 80%;

  background-size: 100% 50%;

5. Time Values

- Used for animations or transitions. Units: s (seconds), ms (milliseconds).

  transition-duration: 0.3s;

6. Numeric Values

- Plain numbers, often used without units in specific contexts.

z-index: 10;

opacity: 0.8;

7. URL Values

- For linking external resources like images or fonts.

background-image: url('background.jpg');

b) Font Properties

1. Font Family

- Specifies the typeface for the text.
- Accepts a comma-separated list of font names (fallbacks).
- Generic family options: serif, sans-serif, monospace, cursive, fantasy.

font-family: 'Arial', 'Helvetica', sans-serif;

2. Font Size

- Defines the size of the text.
- Accepts length units (px, em, rem, %, etc.) or keywords (small, medium, large, etc.).

font-size: 16px;

font-size: 1.2em;

3. Font Weight

- Determines the thickness of the text.
- Accepts numeric values (100 to 900) or keywords (normal, bold, lighter, bolder).

font-weight: bold;

font-weight: 400; / Normal weight */*

4. Font Style

- Specifies whether text is normal, italic, or oblique.

font-style: normal;

font-style: italic;

font-style: oblique;

5. Font variant

- Controls the display of small caps or other typographic variants.

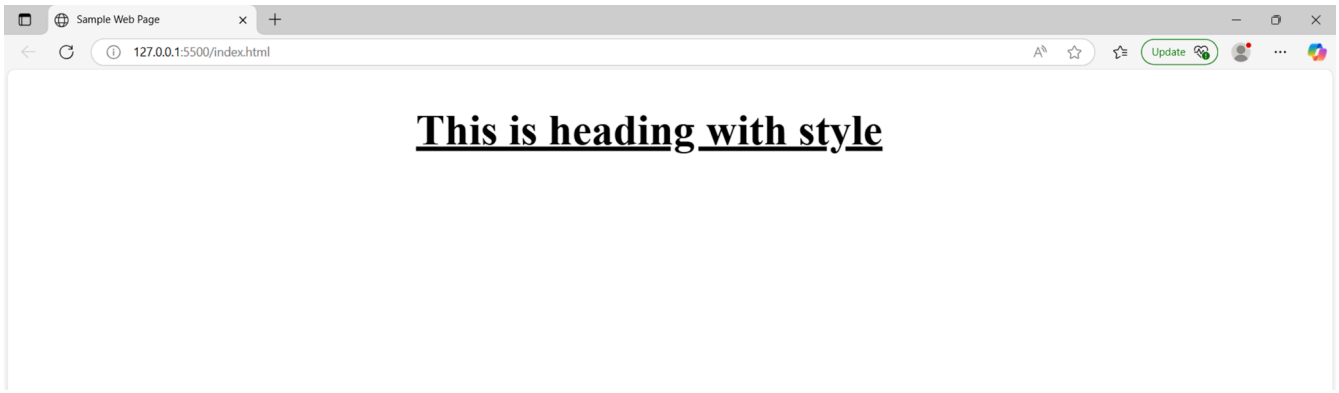
font-variant: small-caps;

font-variant: normal;

c) Text alignment and Decoration

- Align text and apply decorations such as underlines.
- Common Properties:
  - text-align: Aligns text (left, center, right, or justify).
  - text-decoration: Adds underline, overline, or strikethrough.
  - line-height: Adjusts spacing between lines.

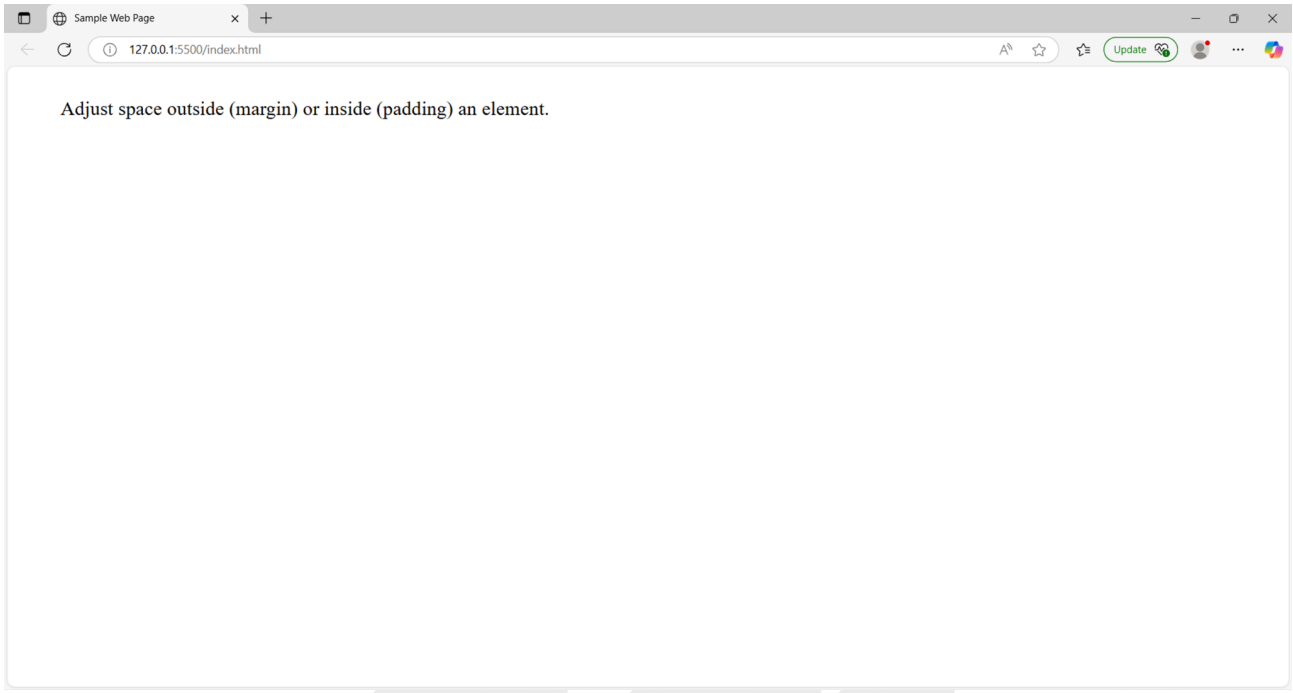
```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Sample Web Page</title>
        <style>
            h1 {
                text-align: center;
                text-decoration: underline;
                line-height: 1.5;
            }
        </style>
    </head>
    <body>
        <h1>This is heading with style</h1>
    </body>
</html>
```



d) Margin and Padding

- Adjust space outside (margin) or inside (padding) an element.
- Common Properties:
    - margin: Sets the outer space.
    - padding: Sets the inner space.
    - margin-top, margin-right, margin-bottom,
    - margin-left, padding-left: Specify sides.

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Sample Web Page</title>
        <style>
            div {
                margin: 20px;
                padding: 15px;
            }
        </style>
    </head>
    <body>
        <div>
            Adjust space outside (margin) or inside
            (padding) an element.
        </div>
    </body>
</html>
```

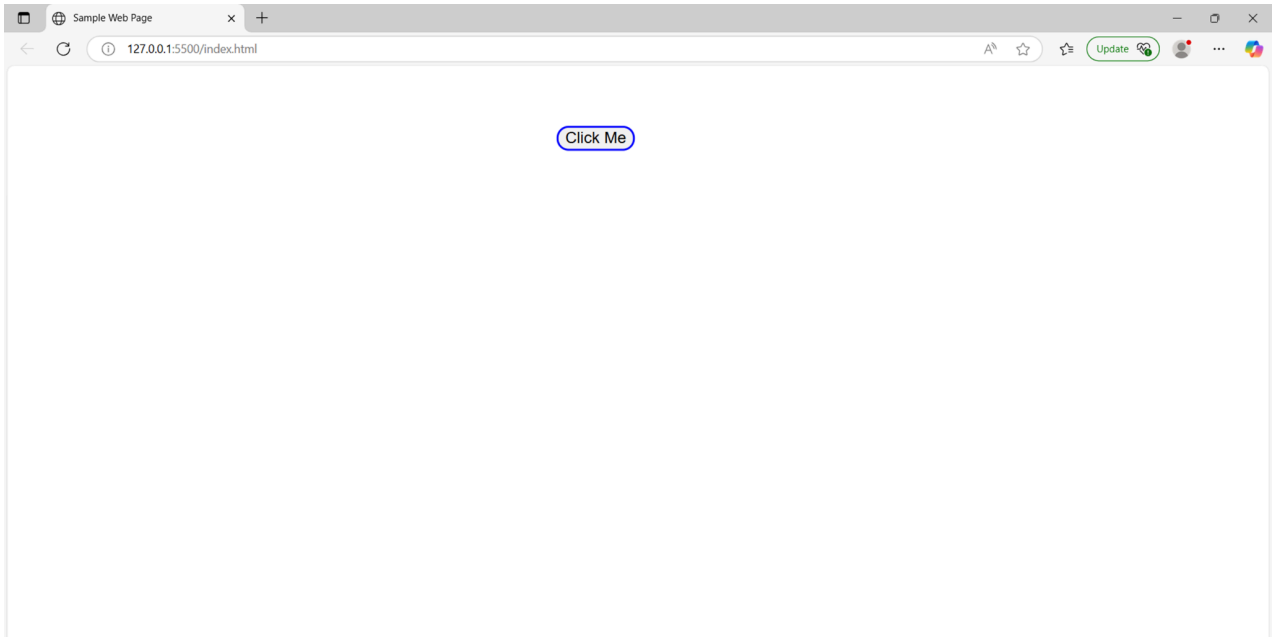


e) CSS Borders

- Control the appearance of element borders.
- Common Properties:
    - border: Shorthand for setting border width, style, and color.
    - border-radius: Rounds corners.
    - border-width: Specifies the width of the border as thin, medium, thick, or using units like px, em, rem, %, etc. Specify widths for each side in the order: top | right | bottom | left.
    - border-style: Defines the style of the border around an element, such as solid, dashed, dotted, etc.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Web Page</title>
```

```
        <style>
            button {
                border: 2px solid blue;
                border-radius: 10px;
            }
        </style>
    </head>
    <body>
        <div style="text-align: center; margin-top: 50px;">
            <button>Click Me</button>
        </div>
    </body>
</html>
```

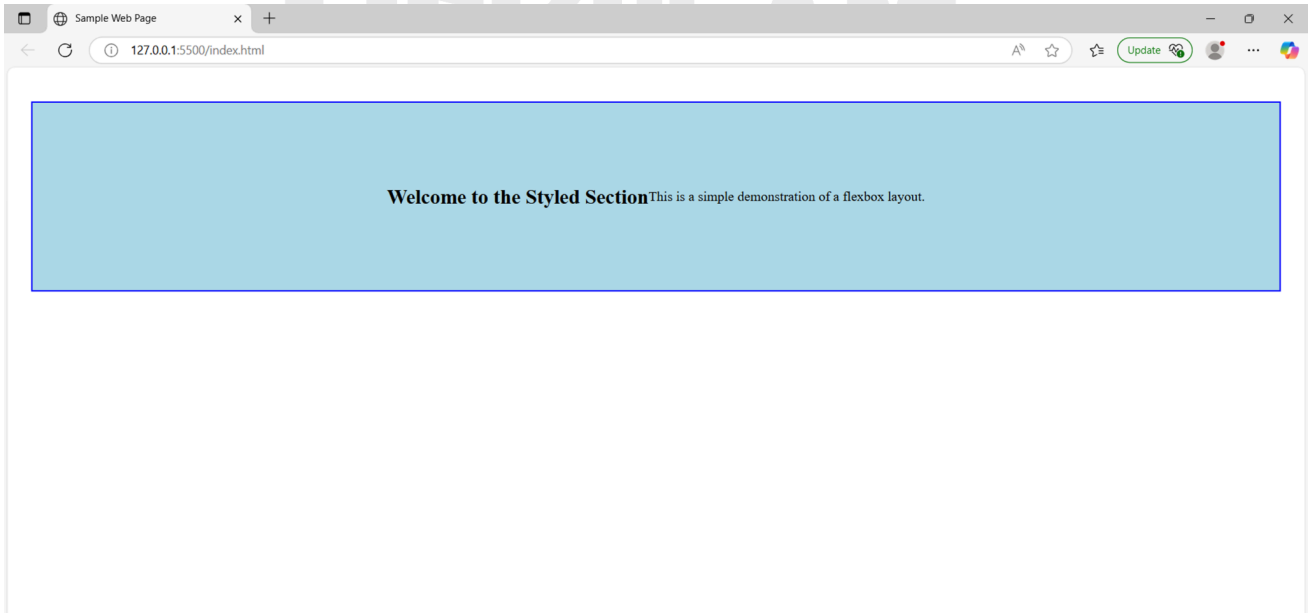


f) Display and Positions

- Control how elements are displayed and positioned on the page.
- Common Properties:
    - display: Defines the display type (block, inline, flex, etc.).
    - position: Specifies element positioning (static, relative, absolute, fixed).
    - top, left: Adjust the element's position.
    - The display: flex creates a flex container.
    - justify-content: center and align-items: center center the content horizontally and vertically.
    - top: 20px moves the section 20px down relative to its normal position.
    - background-color: lightblue adds a light blue background to the section.

```
<head>
    <title>Sample Web Page</title>
    <style>
        section {
            display: flex
            position: relative
            top: 20px;
            justify-content: center;
            align-items: center;
            height: 200px;
            background-color: lightblue;
```

```
            border: 2px solid blue;
            margin: 20px;
            padding: 10px;
        }
    </style>
</head>
<body>
    <section>
        <h2>Welcome to the Styled Section</h2>
        <p>
            This is a simple demonstration of a flexbox layout.
        </p>
    </section>
</body>
```



8. CSS Colors

- CSS provides multiple ways to specify and apply colors to HTML elements.
- Colors are essential in web design and can be set for backgrounds, text, borders, and other properties.

1. Named Colors

- CSS has 140 predefined color names, such as red, blue, green, gold, coral, etc.

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

2. Hexadecimal Notation

- A six-digit or three-digit code representing RGB values.

```
h1 { color: #ff5733;}
h2 { color: #f53; }
```

3. RGB (Red, Green, Blue)

- Specifies colors using RGB values ranging from 0 to 255.

```
div {
        background-color: rgb(255, 87, 51);
}
```

4. Named Colors

- RGBA color values are an extension of RGB color values with an alpha channel that specifies the opacity of a color.

```
section { background-color: rgba(255, 87, 51, 0.5); }
```

5. HSLA (Hue, Saturation, Lightness, Alpha)

- Extends HSL with an alpha channel for transparency.
- An HSL color value is specified as:
- hsl(hue, saturation, lightness)
    - Hue is a degree on the color wheel (0 to 360):
        - 0 (or 360) is red
        - 120 is green
        - 240 is blue
    - Saturation is a percentage value: 100% is full color.
    - Lightness is also a percentage: 0% is dark (black) and 100% is white.

```
p { color: hsla(30, 100%, 50%, 0.7); /* Semi-transparent orange */ }
```

6. Current Color Keyword

- The currentColor keyword is like a variable that holds the current value of the color property of an element.
- This keyword can be useful for consistency across a page.

```
button {
    border: 2px solid currentColor;
}
```

7. Transparent Keyword

- The transparent keyword is used to make a color transparent, often for background colors.

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

9. CSS Background

CSS provides powerful properties for styling the background of elements, allowing customization using colors, images, gradients, and more.

1. Background Color

- Specifies the background color of an element.

element {

background-color: color;

}

2. Background Image

- Sets an image as the background.

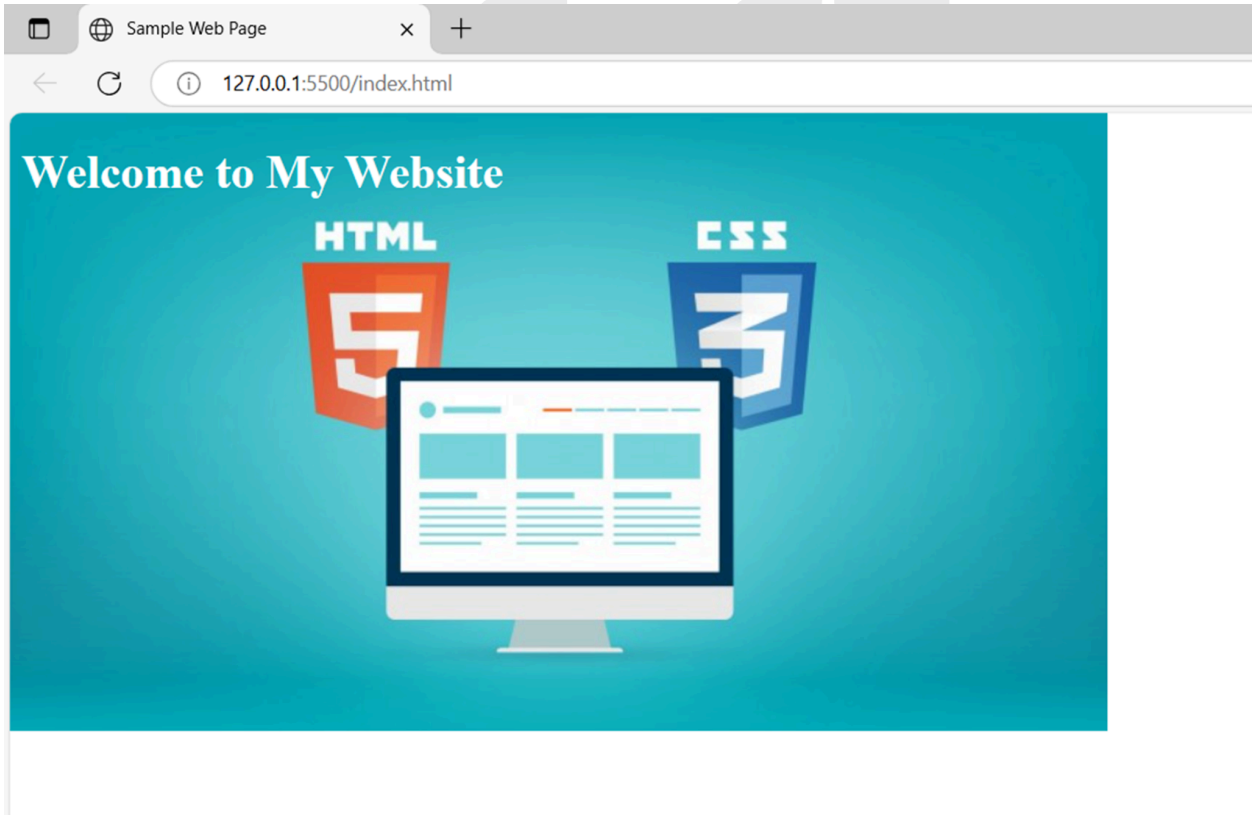
element {

background-image: url('image.jpg');

}

```
<!DOCTYPE html>
<html>
    <head>
        <title>Sample Web Page</title>
        <style>
            body {
            background-image: url('background.jpg');
                background-repeat: no-repeat;
            }
            h1 {
```

```
            color: white;
        }
    </style>
</head>
<body>
    <h1>Welcome to My Website</h1>
</body>
</html>
```



10. Box Model

- The CSS Box Model is a fundamental concept that describes how every HTML element is represented as a rectangular box.
- It determines how the element's content, padding, borders, and margins interact to define its size and spacing in the layout

Components of the CSS Box Model

1. Content
    - The innermost part of the box where text, images, or other content is displayed.
    - The width and height properties control the size of the content.

      width: 200px;

      height: 100px;

2. Padding
    - The innermost part of the box where text, images, or other content is displayed.
    - The width and height properties control the size of the content.
    - The space between the content and the border.
    - Padding adds extra space inside the element but increases the overall size of the box.
    - This can be set individually for each side (padding-top, padding-right, padding-bottom, padding-left) or all sides at once.

padding: 10px; / Uniform padding */*

padding: 10px 20px; / Vertical | Horizontal */*

3. Border

- A layer that wraps around the padding and content.
- Borders can have styles (solid, dashed, dotted, etc.), widths, and colors. Like padding,
- Borders also increase the overall size of the box.

border: 2px solid black;

4. Margin

- The outermost space that separates the element from neighboring elements.
- Does not affect the element's size but controls spacing outside the box.
- Can collapse with adjacent margins (margin collapsing).

margin: 20px;

margin: 10px 15px 20px 5px;

Eg: index.html

```
<head>
    <title>CSS Box Model Example</title>
    <style>
        .box {
            width: 200px;
            height: 100px;
            padding: 10px 20px;
```

```
            border: 2px solid black;
            margin: 20px;
            margin: 10px 15px 20px 5px;
            background-color: lightblue;
            box-sizing: border-box;
        }
    </style>
</head>
<body>
    <div class="box">
        <p>This is a box with custom styles.</p>
    </div>
</body>
```