Chapter 2: Beginner's Guide to CSS

Welcome to this beginner's guide to CSS (Cascading Style Sheets). This document will walk you through the fundamentals of CSS, from basics to more advanced topics like layouts, effects, and variables. We'll include explanations, purposes, code examples, a quiz, exercises, and a cheat sheet for quick reference. Let's dive in!

What is CSS and Why Do We Need It?

CSS stands for Cascading Style Sheets. It is a stylesheet language used to describe the presentation and formatting of a document written in HTML or XML. CSS separates the content (HTML) from the design (styles), making it easier to maintain and update websites.

Purpose and Usage:

- **Why we need it:** Without CSS, HTML pages would look plain and unstructured. CSS allows you to control layout, colors, fonts, and more, improving user experience, accessibility, and responsiveness across devices.
- **How to use it:** CSS is applied to HTML elements via selectors, properties, and values. It can be embedded in three ways (detailed later). It "cascades," meaning styles can inherit or override based on specificity and order.

Example: A basic HTML page without CSS looks plain. With CSS, you can style it.

```html
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
    <title>My Page</title>
    <style>
        body { background-color: lightblue; }  /* Simple CSS rule */
    </style>
</head>
<body>
    <h1>Hello, World!</h1>
</body>
</html>
```

This sets the background color of the body to light blue.

CSS Syntax: Selector, Property, and Value

Explanation: A CSS rule consists of a **selector**, which targets HTML elements to style, a **property**, which specifies what aspect of the element to style (e.g., color, font), and a **value**, which defines how to style it (e.g., red, 16px). Rules are written within curly braces `{}` and end with a semicolon `;`.

Purpose: This structure allows precise control over styling.

There are 3 Types - Common Selector Types:

```
/* Type 1: Tag selector */
p {
    color: navy;    /* the color of all paragraphs will be navy */
}

/* Type 2: Class selector via using (.) */
.text {
    font-family: Arial, sans-serif;
    /* The assigned element class font will be Arial, if not found, it will
sans-serif. */
}

/* Type 3: ID selector via using # */
#banner {
    padding: 20px;
    /* The assigned id element will have 20px space around its text. */
}
```

```
<!-- index.html -->
<!DOCTYPE html>
<html>
<head>
    <title>My Page</title>
</head>
<body>
    <p> This text in navy color </p>
    <p class='text' > This text in navy color and in an Arial font </p>
    <p id='banner' > This text in navy color and and have 20px padding
space around it </p>
    <h1 class='text' >This text in in an Arial font only </h1>
</body>
</html>
```

Color Values: RGB and HEX

Explanation: The `color` property sets the foreground color of an element, typically the text color. The `background-color` property sets the background color of an element. Colors can be specified in various formats, including RGB, HEX, and named colors.

Possible Values for `color` and `background-color` (in order):

- `red`: A named color, one of many predefined color names (e.g., blue, green).
- `#FF0000`: HEX format, a 6-digit code representing RGB values (red, green, blue) in hexadecimal. Shorthand: #F00.
- `rgb(255, 0, 0)`: RGB format, where each value (0-255) specifies red, green, and blue intensity.
- `rgba(255, 0, 0, 0.5)`: RGB with alpha channel (0-1) for transparency.

Purpose: To define colors for text, backgrounds, borders, etc., ensuring visual appeal and branding consistency.

Examples (4 Examples - Common Color Formats):

```
/* Example 1: Named color */
h1 {
    color: red; /* Named color, e.g., blue, green */
}

/* Example 2: RGB */
h2 {
    color: rgb(0, 128, 0); /* RGB values (0-255) for red, green, blue */
}

/* Example 3: HEX */
h3 {
    color: #0000FF; /* 6-digit HEX code for RGB, shorthand #00F */
}

/* Example 4: RGBA */
p {
    background-color: rgba(255, 165, 0, 0.5); /* RGB with alpha (0-1) for
transparency */
}
```

CSS Fonts: Font Size, Color, Face, Weight, Text Align, Line Height

You can change the element's fonts properties using CSS as follow:

Explanation and Possible Values:

- **Font Size (`font-size`)**: Sets the size of text. Possible values:
  - `16px`: Fixed size in pixels.
  - `1em`: Relative to the parent element’s font size.
  - `1rem`: Relative to the root element’s font size.
- **Font Color (`color`)**: Sets the text color. See **Color Values** section for values (red, #FF0000, rgb(255, 0, 0), rgba(255, 0, 0, 0.5)).
- **Font Face/Family (`font-family`)**: Specifies the typeface. Possible values:
  - `Arial, sans-serif`: Primary font with a fallback (sans-serif if Arial is unavailable).
  - `"Times New Roman", serif`: Serif font with fallback.
  - `"Courier New", monospace`: Monospace font with fallback.
- **Font Weight (`font-weight`)**: Sets the thickness of text. Possible values:
  - `normal`: Default weight (400).
  - `bold`: Bold weight (700).
  - `700`: Numeric weight (100-900 in increments of 100).
- **Text Align (`text-align`)**: Aligns text horizontally. Possible values:
  - `left`: Aligns text to the left.
  - `center`: Centers text.

    - `right`: Aligns text to the right.
    - `justify`: Stretches text to align both edges.
- **Line Height (`line-height`)**: Sets the space between lines of text. Possible values:
    - `1.5`: Multiplier of font size (150%).
    - `20px`: Fixed height in pixels.
    - `normal`: Browser default (typically 1.2).

Purpose: Enhances readability and aesthetics by controlling text appearance.

Examples (4 Examples - Common Font Styling Combinations):

```
/* Example 1: Serif with large size */
p.style1 {
    font-family: "Georgia", serif; /* Primary font with fallback */
    font-size: 18px; /* Fixed size in pixels */
    color: #2F4F4F; /* HEX color */
    font-weight: normal; /* Default weight (400) */
    text-align: left; /* Align text to the left */
    line-height: 1.4; /* 140% of font size */
}

/* Example 2: Sans-serif with center alignment */
p.style2 {
    font-family: "Helvetica", sans-serif; /* Sans-serif font with fallback
*/
    font-size: 16px; /* Fixed size in pixels */
    color: rgb(139, 69, 19); /* RGB color */
    font-weight: bold; /* Bold weight (700) */
    text-align: center; /* Centers text */
    line-height: 1.6; /* 160% of font size */
}

/* Example 3: Monospace with right alignment */
p.style3 {
    font-family: "Courier New", monospace; /* Monospace font with fallback
*/
    font-size: 14px; /* Fixed size in pixels */
    color: #8B008B; /* HEX color */
    font-weight: 500; /* Numeric weight (100-900) */
    text-align: right; /* Align text to the right */
    line-height: 1.8; /* 180% of font size */
}

/* Example 4: Justified sans-serif */
p.style4 {
    font-family: "Verdana", sans-serif; /* Sans-serif font with fallback */
    font-size: 12px; /* Fixed size in pixels */
    color: hsl(120, 50%, 50%); /* HSL color */
    font-weight: 700; /* Numeric weight (bold) */
    text-align: justify; /* Stretches text to align both edges */
    line-height: 1.2; /* 120% of font size */
}
```

CSS Block and Inline HTML Elements

Explanation:

- **Block Elements**: Elements that take the full width of their parent and start on a new line. They are typically used for structural components like sections or paragraphs. Common block elements: `div`, `p`, `h1-h6`, `ul`, `table`.
- **Inline Elements**: Elements that only take the width of their content and stay inline with other elements, without starting a new line. They are used for text-level or small components. Common inline elements: `span`, `a`, `img`, `strong`, `em`.
- **CSS Interaction**:
    - **Display (`display`)**: Controls element rendering. Values:
        - `block`: Forces an element to behave as a block element.
        - `inline`: Forces an element to behave as an inline element.
        - `inline-block`: Inline flow but allows block properties like width/height.
        - `none`: Hides the element.

Purpose: Understanding block and inline elements helps control layout and apply appropriate CSS styling (e.g., margins work differently on inline elements).

Examples with Sample HTML (4 Examples - Block and Inline Behaviors):

```
<!DOCTYPE html>
<html>
<head>
    <title>Block and Inline Demo</title>
    <style>
        /* Example 1: Block element styling */
        .block1 {
            display: block; /* Full-width, new line */
            width: 200px; /* Fixed width in pixels */
            margin: 10px; /* Fixed margin in pixels */
            background-color: #f0f0f0; /* HEX color */
        }

        /* Example 2: Inline element styling */
        .inline1 {
            display: inline; /* Inline flow, no line break */
            padding: 5px; /* Fixed padding */
            background-color: #e0e0ff; /* HEX color */
        }

        /* Example 3: Inline-block element */
        .inline-block1 {
            display: inline-block; /* Inline with block properties */
            width: 100px; /* Fixed width in pixels */
            height: 50px; /* Fixed height in pixels */
            margin: 5px; /* Fixed margin */
            background-color: #ffcccb; /* HEX color */
        }

        /* Example 4: The element will be hidden */
        .none {
            display: none; /* Hides the element */
        }


    </style>
</head>
<body>
    <div class="block1">Block Div</div>
    <span class="inline1">Inline Span</span>
    <span class="inline-block1">Inline-Block Span</span>
    <div class="none">Hidden Div</div>
</body>
</html>
```

CSS Block Model: Margin, Padding, Border, Border-Radius, Width, Height

Now, any block element can have these properties to style it. **Explanation and Possible Values**:

- **Margin (`margin`, `margin-top`, `margin-right`, `margin-bottom`, `margin-left`)**: Sets the outer space around an element. Values for all:
  - `10px 20px 30px 40px`: Sets top, right, bottom, left margins (shorthand for `margin`).
  - `10px 20px`: Top/bottom, left/right.
  - `10px`: All sides.
  - `auto`: Centers block elements horizontally (for `margin-left`, `margin-right`).
  - Individual properties (e.g., `margin-top: 10px`) set specific sides.
- **Padding (`padding`, `padding-top`, `padding-right`, `padding-bottom`, `padding-left`)**: Sets the inner space between content and border. Values:
  - `10px 20px 30px 40px`: Top, right, bottom, left (shorthand for `padding`).
  - `10px 20px`: Top/bottom, left/right.
  - `10px`: All sides.
  - Individual properties (e.g., `padding-top: 10px`) set specific sides.
- **Border (`border`)**: Defines an element's outline with width, style, and color. Values:
  - `1px solid black`: Width, style (solid), color.
  - `2px dashed blue`: Dashed style.
  - `3px dotted red`: Dotted style.
- **Border-Radius (`border-radius`)**: Rounds corners of an element. Values:
  - `10px`: Rounds all corners equally.
  - `5px 10px`: Top-left/bottom-right, top-right/bottom-left.
- **Width (`width`)**: Sets the content width. Values:
  - `100px`: Fixed width in pixels.
  - `50%`: Percentage of parent width.
  - `auto`: Default, based on content or parent.
- **Height (`height`)**: Sets the content height. Values:
  - `100px`: Fixed height in pixels.

    - `auto`: Based on content.
- **Box-Sizing (`box-sizing`)**: Defines how width/height are calculated. Values:
    - `content-box`: Width/height applies to content only (default).
    - `border-box`: Includes padding and border in width/height.
- **Box-Shadow (`box-shadow`)**: Adds a shadow effect. Values:
    - `5px 5px 10px gray`: Offset-x, offset-y, blur, color (standard shadow).
    - `inset 0 0 8px blue`: Inset shadow inside element.
    - `0 0 20px rgba(128, 0, 128, 0.4)`: Glow effect with no offset.

Block Elements: Elements that take full width and start on a new line (e.g., div, p, h1-h6, ul, table). **Inline Elements**: Elements that only take the width of their content and stay inline (e.g., span, a, img, strong, em).

Purpose: Controls spacing and sizing for layout.

Examples (4 Examples - Common Box Model Styles):

```
/* Example 1: Standard box */
div.box1 {
    width: 200px; /* Fixed width in pixels */
    height: 100px; /* Fixed height in pixels */
    padding: 10px; /* Same padding all sides */
    border: 2px solid black; /* Width, solid style, named color */
    margin: 20px; /* Same margin all sides */
    border-radius: 15px; /* Rounds all corners equally */
    box-sizing: border-box; /* Includes padding and border in width/height
*/
}

/* Example 2: Dashed border with centered margin */
div.box2 {
    width: 200px; /* Fixed width in pixels */
    height: 100px; /* Fixed height in pixels */
    padding: 15px; /* Same padding all sides */
    border: 3px dashed blue; /* Width, dashed style, named color */
    margin: 10px auto; /* Top/bottom margin, auto centers horizontally */
    border-radius: 10px; /* Rounds all corners equally */
}

/* Example 3: Minimal padding */
div.box3 {
    width: 180px; /* Fixed width in pixels */
    height: 120px; /* Fixed height in pixels */
    padding: 5px; /* Same padding all sides */
    border: 1px solid green; /* Width, solid style, named color */
    margin: 15px; /* Same margin all sides */
    border-radius: 20px; /* Rounds all corners equally */
}

/* Example 4: Double border */
div.box4 {
    width: 220px; /* Fixed width in pixels */
    height: 80px; /* Fixed height in pixels */
    padding: 20px; /* Same padding all sides */
    border: 4px double red; /* Width, double style, named color */
    margin: 25px; /* Same margin all sides */
    border-radius: 5px; /* Rounds all corners equally */
}
```

CSS Background: Attachment, Color, Image, Position, Repeat, Origin, Size

To set the background, you can use the background properties as follow

Explanation and Possible Values:

- **Background-Color (`background-color`)**: Sets the background color of an element. Values:
  - `#f0f0f0`: HEX color.
  - `rgb(255, 255, 255)`: RGB color.
  - `transparent`: No color (default).
- **Background-Image (`background-image`)**: Sets a background image or gradient. Values:
  - `url('image.jpg')`: Image file URL.
  - `linear-gradient(to right, blue, white)`: Linear gradient from left to right.
- **Background-Repeat (`background-repeat`)**: Controls image repetition. Values:
  - `repeat`: Repeats in both directions.
  - `repeat-x`: Repeats horizontally.
  - `repeat-y`: Repeats vertically.
  - `no-repeat`: No repetition.
- **Background-Position (`background-position`)**: Sets the image's starting position. Values:
  - `center`: Centers the image.
  - `top left`: Positions at top-left corner.
  - `10px 20px`: Custom position (x, y).
- **Background-Size (`background-size`)**: Sets the image size. Values:
  - `cover`: Scales to cover element, may crop.
  - `contain`: Scales to fit without cropping.
  - `100px`: Fixed size (width or width/height).
- **Background-Attachment (`background-attachment`)**: Controls image behavior on scroll. Values:
  - `scroll`: Moves with the element (default).
  - `fixed`: Stays fixed relative to viewport.
- **Background-Origin (`background-origin`)**: Sets where the image starts. Values:
  - `padding-box`: Starts from padding edge (default).
  - `border-box`: Starts from border edge.
  - `content-box`: Starts from content edge.

Purpose: Add visual interest to element backgrounds.

Examples (4 Examples - Common Background Styles):

```
/* Example 1: Solid color */
div.bg1 {
    background-color: #f0f0f0; /* HEX color */
}

/* Example 2: Image with cover */
div.bg2 {
    background-image: url('image.jpg'); /* Image file URL */
    background-position: center; /* Centers the image */
    background-repeat: no-repeat; /* No repetition */
    background-size: cover; /* Scales to cover element, may crop */
}

/* Example 3: Linear gradient */
div.bg3 {
    background: linear-gradient(to right, blue, white); /* Linear gradient 
from left to right */
}

/* Example 4: Repeating pattern */
div.bg4 {
    background-image: url('pattern.png'); /* Image file URL */
    background-repeat: repeat-x; /* Repeats horizontally */
    background-position: top; /* Positions at top */
}
```

CSS Embedding: External, Internal, Inline

Explanation: CSS can be applied to HTML in three ways, each with different use cases and priorities. There are no specific CSS properties here, but the methods are:

- **External**: Links an external `.css` file using `<link>`.
- **Internal**: Embeds CSS within a `<style>` tag in the HTML `<head>`.
- **Inline**: Applies CSS directly to an element using the `style` attribute.
- **Priority**: Determines which styles apply when there are conflicts. Possible values (in order of precedence):
    - Inline: Highest priority, applied directly on the element.
    - ID: Overrides class and tag selectors due to higher specificity.
    - Class: Overrides tag selectors.
    - Tag: Overrides inherited styles.
    - Inheritance: Styles inherited from parent elements (lowest priority).

Purpose: Flexibility in managing styles—external for site-wide, internal for page-specific, inline for quick overrides.

Examples (3 Examples - All Embedding Methods):

```
<!-- Example 1: External CSS (styles.css) -->
<link rel="stylesheet" href="styles.css">
<!-- styles.css -->
body { background-color: #E6E6FA; } /* HEX color for background */
```

```
<!-- Example 2: Internal CSS -->
<head>
    <style>
        h1 {
            color: teal; /* Named color */
            font-size: 24px; /* Fixed size in pixels */
        }
    </style>
</head>
```

```
<!-- Example 3: Inline CSS -->
<div style="border: 2px solid maroon; padding: 10px;">Content</div> <!--
Solid border, fixed padding -->
```

CSS Selectors: By Tag, ID, Class; Combining; Inheritance

Explanation and Possible Values:

- **Tag Selector**: Targets elements by their HTML tag name (e.g., `p` for all paragraphs). Value: Any valid HTML tag (e.g., `p`, `div`, `h1`).
- **ID Selector**: Targets a single element with a unique `id` attribute. Value: `#id` (e.g., `#hero`).
- **Class Selector**: Targets elements with a specific `class` attribute. Value: `.class` (e.g., `.info`).
- **Combined Selectors**:
  - **Multiple**: Targets multiple selectors. Value: Comma-separated list (e.g., `h1, h2`).
  - **Descendant**: Targets elements inside another. Value: Space-separated (e.g., `div p`).
  - **Child**: Targets direct children. Value: `>` separator (e.g., `div > p`).
  - **Adjacent Sibling (`+`)**: Targets the next sibling element immediately following another. Value: `+` separator (e.g., `h2 + p` styles a `<p>` immediately after an `<h2>`).
  - **General Sibling (`~`)**: Targets all sibling elements following another. Value: `~` separator (e.g., `h2 ~ p` styles all `<p>` elements after an `<h2>` in the same parent).
- **Inheritance**: Not a selector but a behavior where children inherit properties (e.g., `color`, `font-family`) from parents unless overridden.

Purpose: To apply styles precisely to specific elements or groups.

Examples (4 Examples - Common Selector Types, Including Sibling Combinators):

```
/* Example 1: Tag selector */
div {
    background-color: #F0FFF0; /* HEX color for background */
}

/* Example 2: ID selector */
#hero {
    font-size: 28px; /* Fixed size in pixels */
    color: navy; /* Named color */
}

/* Example 3: Adjacent sibling selector */
h2 + p {
    color: #4682B4; /* HEX color */
    font-style: italic; /* Italic text style */
     font-weight: 700;
}

/* Example 4: General sibling selector */
h2 ~ p {
    color: #228B22; /* HEX color */
    margin-top: 10px; /* Fixed margin in pixels */
}

/* Example 5: Style element that have the following order */
/* Style all <li> that it inside a <ul> that is inside a <div
class='color>. Note that is possible some elements in between like: */
/* div div ul li  will still work*/
div.color ul li{
  color:red;
}

/* Style all <li> that it inside a <ul> that is inside a <div
class='color>. Note that is must be direct, no elements in between like: */
/* div div ul li  will not work*/
div.color > ul > li{
  color:red;
}
```

CSS Selector Attributes

Explanation: Attribute selectors target elements based on their HTML attributes or attribute values.

Possible Values (in order):

- `[type="text"]`: Matches elements with an exact attribute value (e.g., `type="text"`).
- `[href^="https"]`: Matches attributes starting with a value (e.g., `href` starting with "https").
- `[alt*="logo"]`: Matches attributes containing a substring (e.g., `alt` containing "logo").
- `[href$=".pdf"]`: Matches attributes ending with a value (e.g., `href` ending with ".pdf").

Purpose: Style elements with specific attribute values.

Examples (4 Examples - Common Attribute Selectors):

```
/* Example 1: Exact match */
input[type="text"] {
    border: 1px solid blue; /* Solid border, named color */
}
```

```
/* Example 2: Starts with */
a[href^="https"] {
    color: green; /* Named color */
}

/* Example 3: Contains */
img[alt*="logo"] {
    border-radius: 10px; /* Rounds corners equally */
}

/* Example 4: Ends with */
a[href$=".pdf"] {
    font-weight: bold; /* Bold weight (700) */
}
```

Pseudo-Classes and Pseudo-Elements

Explanation:

- **Pseudo-Classes**: Style elements based on their state or position. Common values:
    - `:link`: Styles unvisited links.
    - `:visited`: Styles visited links.
    - `:hover`: Styles elements on mouse hover.
    - `:active`: Styles elements during interaction (e.g., click).
- **Pseudo-Elements**: Style specific parts of an element. Common values:
    - `::before`: Inserts content before an element’s content.
    - `::first-letter`: Styles the first letter of an element.

Purpose: Dynamic styling without JavaScript.

Examples (4 Examples - Common Pseudo-Classes/Elements):

```
/* Example 1: Link states */
a:link {
    color: #0000FF; /* HEX color for unvisited links */
}
a:visited {
    color: #800080; /* HEX color for visited links */
}

/* Example 2: Hover effect */
button:hover {
    background-color: #FFD700; /* HEX color on mouse hover */
}

/* Example 3: First-child */
li:first-child {
    font-size: 18px; /* Fixed size in pixels */
    color: teal; /* Named color */
}

/* Example 4: Pseudo-element for first letter */
p::first-letter {
    font-size: 2em; /* Relative to parent font size */
    color: red; /* Named color */
}
```

CSS Units and Measurement

Explanation: CSS units define the size or length of properties like `width`, `font-size`, `margin`, etc. Units can be absolute (fixed, e.g., pixels) or relative (based on other elements, viewport, or font size).

Possible Values (in order):

- **Absolute Units**:
  - `px`: Pixels, a fixed unit relative to screen resolution (e.g., 16px is a common font size).
  - `cm`: Centimeters, used for print styles (e.g., 1cm).
  - `mm`: Millimeters, also for print (e.g., 10mm).
- **Relative Units**:
  - `em`: Relative to the parent element's font size (e.g., `1em` = parent's font size).
  - `rem`: Relative to the root (`html`) element's font size (e.g., `1rem` = 16px if root is 16px).
  - `%`: Percentage of the parent element's size (e.g., `50%` width is half the parent's width).
  - `vw`: Viewport width, 1% of the viewport's width (e.g., `50vw` is half the viewport width).
  - `vh`: Viewport height, 1% of the viewport's height (e.g., `100vh` is full viewport height).

Purpose: Provide flexibility in sizing elements, enabling responsive and scalable designs.

CSS: Styling Tables

Explanation and Possible Values:

- **Border-Collapse (`border-collapse`)**: Controls whether table cell borders merge. Values:
  - `collapse`: Merges adjacent borders into a single border.
  - `separate`: Keeps borders separate (default).
- **Border-Spacing (`border-spacing`)**: Sets the space between cells when `border-collapse: separate`. Values:
  - `5px`: Uniform spacing.
  - `5px 10px`: Horizontal, vertical spacing.
- **Cell Styling (`th, td`)**: Styles table cells. Common properties:
  - `border: 1px solid`: Sets cell borders.
  - `padding: 8px`: Adds padding inside cells.
  - `text-align: left`: Aligns cell content (left, center, right).

Purpose: Make tables readable and visually appealing.

Examples (4 Examples - Common Table Styles):

```
/* Example 1: Collapsed borders with zebra stripes */
table.style1 {
    border-collapse: collapse; /* Merges adjacent borders */
    width: 100%; /* Full width of parent */
}
table.style1 th, table.style1 td {
    border: 1px solid black; /* Width, solid style, named color */
    padding: 8px; /* Fixed padding */
    text-align: left; /* Aligns content to the left */
}
table.style1 tr:nth-child(even) {
    background-color: #f2f2f2; /* HEX color for even rows */
}

/* Example 2: Separate borders */
table.style2 {
    border-collapse: separate; /* Keeps borders separate */
    border-spacing: 5px; /* Uniform spacing between cells */
    width: 100%; /* Full width of parent */
}
table.style2 td {
    border: 2px solid navy; /* Width, solid style, named color */
    text-align: center; /* Centers content */
}

/* Example 3: Minimalist with hover */
table.style3 {
    width: 100%; /* Full width of parent */
}
table.style3 td {
    border-bottom: 1px solid gray; /* Bottom border only */
    padding: 5px; /* Fixed padding */
}
table.style3 tr:hover {
    background-color: #FFFACD; /* HEX color on hover */
}

/* Example 4: Colored headers */
table.style4 {
    border-collapse: collapse; /* Merges adjacent borders */
}
table.style4 th {
    background-color: #4682B4; /* HEX color for header background */
    color: white; /* Named color for text */
    padding: 12px; /* Fixed padding */
}
table.style4 td {
    border: 1px dashed blue; /* Width, dashed style, named color */
}
```

Category	Property	Attribute	Possible Values with Explanation

Category	**Property**	**Attribute**	**Possible Values with Explanation**
Syntax & Basics	CSS Rule	selector { property: value; }	`selector { property: value; }`: Targets element (selector), sets style (property: value).
	Colors	color	`red`: Named color. `#FF0000`: HEX (6-digit RGB). `rgb(255, 0, 0)`: RGB values (0-255). `rgba(255, 0, 0, 0.5)`: RGB with alpha (transparency).
		background-color	`#f0f0f0`: HEX color. `rgb(255, 255, 255)`: RGB color. `transparent`: No color.
	Fonts	font-family	`Arial, sans-serif`: Font with fallback. `"Times New Roman", serif`: Serif font. `"Courier New", monospace`: Monospace font.
		font-size	`16px`: Pixel size. `1em`: Relative to parent. `1rem`: Relative to root.
		font-weight	`normal`: Default weight. `bold`: Bold text. `700`: Numeric weight (100-900).
		text-align	`left`: Left-aligned. `center`: Centered. `right`: Right-aligned. `justify`: Justified.
		line-height	`1.5`: 150% of font size. `20px`: Fixed height. `normal`: Default spacing.
Units & Measurement	Units	font-size, width, margin, etc.	`px`: Pixels, fixed unit. `cm`: Centimeters, for print. `mm`: Millimeters, for print. `em`: Relative to parent font size. `rem`: Relative to root font size. `%`: Percentage of parent size. `vw`: 1% of viewport width. `vh`: 1% of viewport height.
Block & Inline	Block Elements		div, p, h1-h6, ul, table: Full-width, new line.
	Inline Elements		span, a, img, strong, em: Inline, no line break, width based on content.

Category	Property	Attribute	Possible Values with Explanation
	Display	display	`block`: Full-width, new line. `inline`: Inline flow. `inline-block`: Inline with block properties. `none`: Hides element.
CSS Variables	Custom Property	--name	`--main-color: blue`: Define variable. `var(--main-color)`: Use variable in property.
Embedding	CSS Inclusion	External	`<link rel="stylesheet" href="file.css">`: Links external CSS file.
		Internal	`<style>...</style>`: CSS in `<head>`.
		Inline	`style="color: red;"`: CSS on element. Priority: Inline > ID > Class > Tag > Inheritance.
Selectors	Basic Selectors	Tag	`p {}`: Targets all `<p>` elements.
		ID	`#id {}`: Targets unique element with `id="id"`.
		Class	`.class {}`: Targets elements with `class="class"`.
	Combined Selectors	Multiple	`h1, h2 {}`: Targets multiple selectors.
		Descendant	`div p {}`: Targets `<p>` inside `<div>`.
		Child	`div > p {}`: Targets direct `<p>` children of `<div>`.
		Adjacent Sibling	`h2 + p {}`: Targets `<p>` immediately following `<h2>`.
		General Sibling	`h2 ~ p {}`: Targets all `<p>` elements after `<h2>` in same parent.
	Attribute Selectors	[attr]	`[type="text"]`: Exact match. `[href^="https"]`: Starts with. `[alt*="logo"]`: Contains. `[href$=".pdf"]`: Ends with.
	Pseudo-Classes	:pseudo	`:link`: Unvisited links. `:visited`: Visited links. `:hover`: On mouse over. `:active`: On click.
	Pseudo-Elements	::pseudo	`::before`: Content before element. `::first-letter`: Styles first letter.

Category	Property	Attribute	Possible Values with Explanation
Box Model	Spacing	margin	`10px 20px 30px 40px`: Top, right, bottom, left. `10px 20px`: Top/bottom, left/right. `10px`: All sides. `auto`: Centers block.
		margin-top	`10px`: Top margin. `auto`: Centers vertically (if applicable).
		margin-right	`20px`: Right margin. `auto`: Centers horizontally (with width).
		margin-bottom	`30px`: Bottom margin.
		margin-left	`40px`: Left margin. `auto`: Centers horizontally (with width).
		padding	`10px 20px 30px 40px`: Top, right, bottom, left. `10px 20px`: Top/bottom, left/right. `10px`: All sides.
		padding-top	`10px`: Top padding.
		padding-right	`20px`: Right padding.
		padding-bottom	`30px`: Bottom padding.
		padding-left	`40px`: Left padding.
	Border	border	`1px solid black`: Width, style, color. `2px dashed blue`: Dashed border. `3px dotted red`: Dotted border.
		border-radius	`10px`: Rounds all corners. `5px 10px`: Top-left/bottom-right, top-right/bottom-left.
	Size	width	`100px`: Fixed width. `50%`: Percentage of parent. `auto`: Default width.
		height	`100px`: Fixed height. `auto`: Content-based height.
		box-sizing	`content-box`: Default (content only). `border-box`: Includes padding/border.

Category	Property	Attribute	Possible Values with Explanation
	Shadow	box-shadow	`5px 5px 10px gray`: Standard shadow (x, y, blur, color). `inset 0 0 8px blue`: Inset shadow. `0 0 20px rgba(128, 0, 128, 0.4)`: Glow effect.
Background	Background	background-color	`#f0f0f0`: HEX color. `rgb(255, 255, 255)`: RGB color. `transparent`: No color.
		background-image	`url('image.jpg')`: Image URL. `linear-gradient(to right, blue, white)`: Gradient.
		background-repeat	`repeat`: Repeat both axes. `repeat-x`: Repeat horizontally. `repeat-y`: Repeat vertically. `no-repeat`: No repeat.
		background-position	`center`: Center image. `top left`: Position at top-left. `10px 20px`: Custom position.
		background-size	`cover`: Scale to cover. `contain`: Scale to fit. `100px`: Fixed size.
		background-attachment	`scroll`: Scrolls with page. `fixed`: Fixed to viewport.
		background-origin	`padding-box`: Starts from padding edge. `border-box`: Starts from border edge. `content-box`: Starts from content edge.
Borders & Shadows	Border	border	See Box Model for border values.
	Shadow	box-shadow	See Box Model for shadow values.
Overflow	Overflow	overflow	`visible`: Shows overflow. `hidden`: Clips overflow. `scroll`: Adds scrollbars. `auto`: Scrollbars if needed.
Tables	Table Styling	border-collapse	`collapse`: Merge borders. `separate`: Separate borders.
		border-spacing	`5px`: Space between cells (separate only). `5px 10px`: Horizontal, vertical spacing.
		th, td	`border: 1px solid; padding: 8px;`: Style cells. `text-align: left`: Align content.

Category	Property	Attribute	Possible Values with Explanation
Float & Position	Float	float	`left`: Float left. `right`: Float right.
		clear	`both`: Clears floats on both sides.
	Position	position	`static`: Default, normal flow. `relative`: Relative to normal position. `absolute`: Relative to positioned ancestor. `fixed`: Fixed to viewport. `sticky`: Sticks on scroll.
		top, left, bottom, right	`10px`: Fixed offset. `50%`: Percentage offset.
Layouts	Flexbox	display	`flex`: Enables flexbox layout.
		flex-direction	`row`: Horizontal (left to right). `column`: Vertical (top to bottom). `row-reverse`: Horizontal, reversed. `column-reverse`: Vertical, reversed.
		flex-wrap	`nowrap`: No wrapping. `wrap`: Wraps to next line. `wrap-reverse`: Wraps in reverse order.
		justify-content	`center`: Centers items. `space-between`: Spaces items, no end gaps. `space-around`: Equal gaps around items.
		align-items	`center`: Centers vertically. `stretch`: Stretches to fill. `baseline`: Aligns baselines.
		flex-basis	`200px`: Base size of item. `auto`: Based on content or width/height.
		flex-grow	`0`: No growth. `1`: Grows proportionally to fill space.
		flex	`1`: Grow/shrink proportionally. `0 0 200px`: No grow/shrink, fixed basis.
	Grid	display	`grid`: Enables grid layout.
		grid-template-columns	`1fr 1fr`: Equal columns. `200px auto`: Fixed and auto. `repeat(3, 1fr)`: Repeat columns.

Category	Property	Attribute	Possible Values with Explanation
		grid-template-rows	`100px 100px`: Fixed row heights. `auto`: Content-based height. `repeat(2, 1fr)`: Equal rows.
		gap	`10px`: Uniform gap. `10px 20px`: Row gap, column gap.
		grid-column	`1 / 3`: Spans from column 1 to 3. `2 / span 2`: Starts at 2, spans 2 columns.
		grid-row	`1 / 2`: Spans first row. `2 / span 2`: Starts at 2, spans 2 rows.
		align-self	`center`: Centers item vertically. `start`: Aligns to top. `end`: Aligns to bottom.
		justify-self	`center`: Centers item horizontally. `start`: Aligns to left. `end`: Aligns to right.
		grid-area	`header`: Assigns item to named area. `1 / 1 / 2 / 3`: Row-start/column-start/row-end/column-end.
Responsive	Viewport	meta	`<meta name="viewport" content="width=device-width, initial-scale=1">`: Responsive viewport.
	Media Queries	@media	`@media (max-width: 600px)`: Mobile screens (≤600px). `@media (max-width: 768px)`: Tablet screens (≤768px). `@media (min-width: 1024px)`: Desktop screens (≥1024px).
Effects	Transform	transform	`translate(50px, 0)`: Move element. `rotate(45deg)`: Rotate. `scale(2)`: Resize. `skew(20deg)`: Slant.
	Filters	filter	`blur(5px)`: Blur effect. `hue-rotate(90deg)`: Change hue. `brightness(120%)`: Adjust brightness. `grayscale(100%)`: Remove color.
	Transitions	transition	`all 1s ease`: Transition all properties over 1s. `background-color 0.5s linear`: Specific property transition.

Category	Property	Attribute	Possible Values with Explanation
	Animations	@keyframes, animation	`@keyframes name { from {} to {} }`: Define animation. `animation: name 2s infinite`: Apply animation.