# Cascading Style Sheets (CSS)

Adding Style and Structure to the Web

*CSS allows developers to design visually engaging websites by separating style from content. This module introduces you to the core concepts, rules, and practices of CSS used in modern web design.*

By: Gino Fernando

## What is CSS?

- CSS stands for Cascading Style Sheets.
- It is a style language used to describe how HTML elements should be displayed on screen, paper, or in other media.
- CSS controls layout, colors, fonts, spacing, and much more.
- It allows a single stylesheet to control the appearance of multiple web pages.

## Why Learn CSS?

- Without CSS, web pages look plain and unstyled.
- CSS enables responsive, interactive, and modern design.
- Enhances user experience (UX) and accessibility.
- Required for layout systems like Flexbox and Grid.
- Supported by all modern browsers and essential for front-end development.

## Ways to Add CSS

- Inline CSS
- Internal CSS
- External CSS

## Inline CSS

- Inside the style attribute of an HTML tag

*Example: <p style="color:red;">Text</p>*

## Internal CSS

- Placed in a <style> block within the <head> tag

*Example:*

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

## External CSS

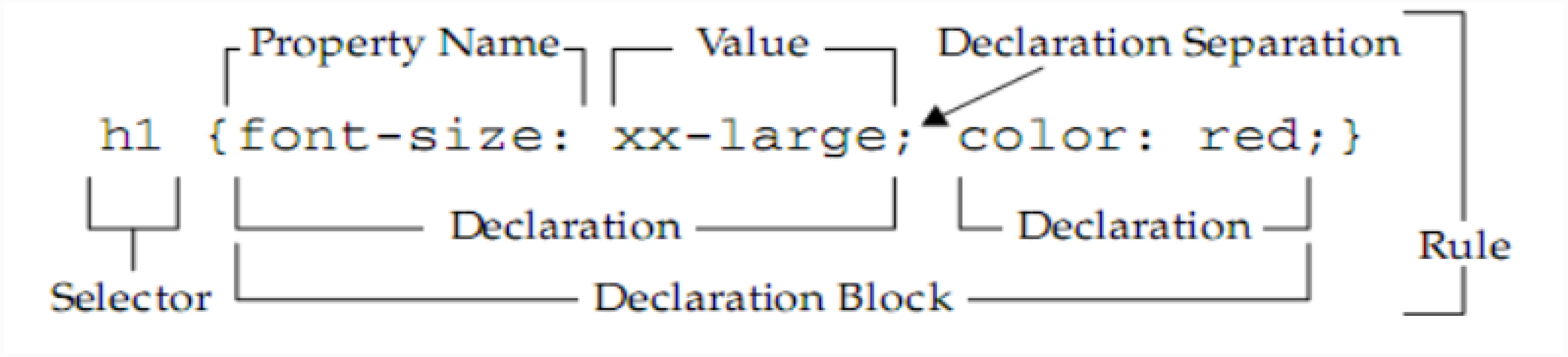
- Linked via a separate .css file
- Best practice: Clean, maintainable, reusable

*Example:*

```
<link rel="stylesheet" href="styles.css">
```

# CSS Syntax and Structure

- A CSS rule consists of selectors and declarations:



Parts:

Selector: Targets the HTML element (h1)

Property: Style attribute (e.g., color)

Value: The value of the property (e.g., navy)

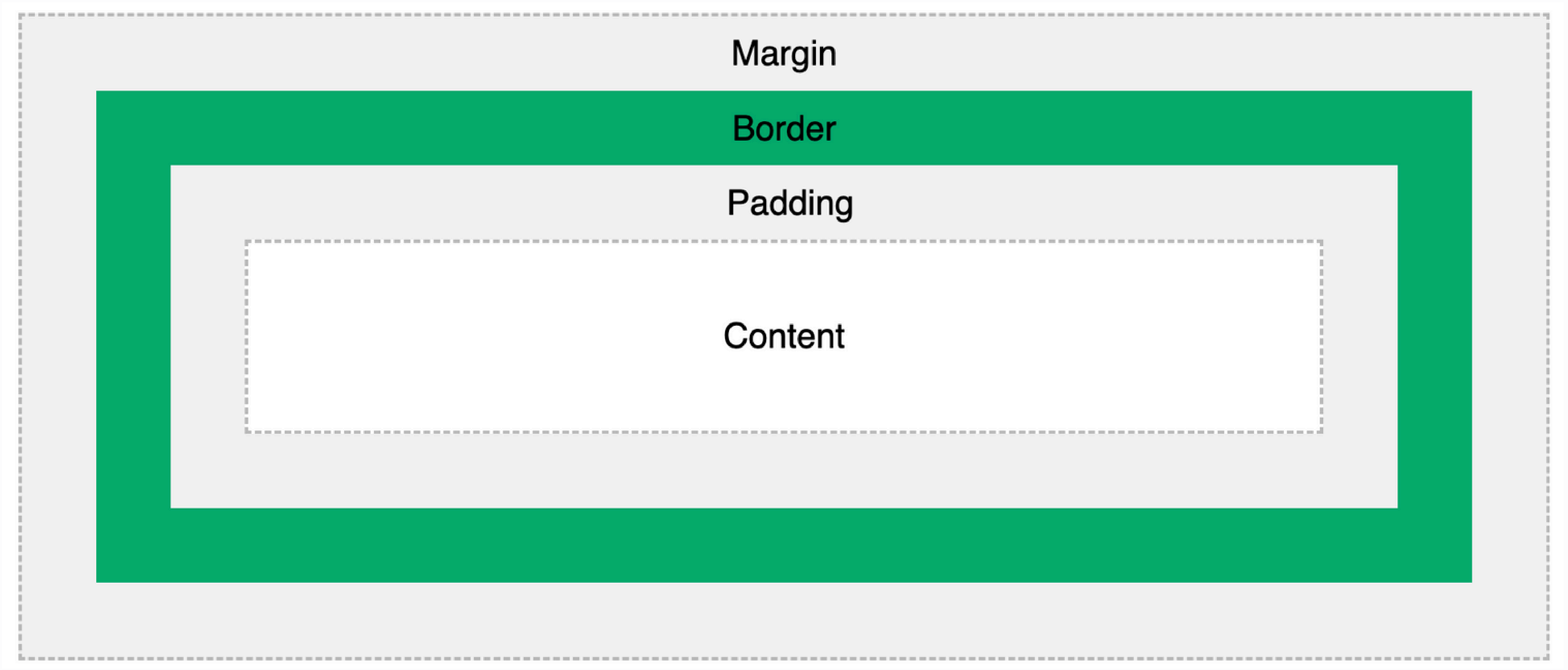
## Types of CSS Selectors

| Selector Type | Example | Description |
|---|---|---|
| Universal | * {} | Targets all elements |
| Element | p {} | Targets all <p> elements |
| Class | .title {} | Targets elements with class="title" |
| ID | #header {} | Targets the element with id="header" |
| Group | h1, p {} | Targets both h1 and p |
| Descendant | div p {} | Targets p inside any div |
| Child | div > p {} | Targets p directly inside div |
| Attribute | input[type="text"] | Targets input with type="text" |
| Pseudo-class | a:hover {} | Applies when a user hovers over a link |
| Pseudo-element | p::first-line {} | Styles only the first line of a p |

## The CSS Box Model

Every HTML element is a rectangular box made up of:

- Content: The actual text or image
- Padding: Space around the content
- Border: Surrounds the padding
- Margin: Space outside the border



Visual: Think of nested boxes from inside out:
content → padding → border → margin

## Margins and Padding

| Property | Description |
|---|---|
| margin | Outer spacing around the element |
| padding | Inner spacing inside the element |

Shorthand Usage:

*/* Top, Right, Bottom, Left */*
*margin: 10px 20px 10px 0;*
*padding: 5px 10px;*

## Flexbox Introduction

Flexbox is a modern layout system for arranging items in rows or columns.
Used for navbars, cards, and adaptive layouts.

Container Properties:

- display: flex
- flex-direction
- justify-content
- align-items

```
.container {
  display: flex;
  justify-content: space-between;
  align-items: center;
}
```

## What is Box-Sizing?

By default, CSS adds padding and border outside the width and height you set.

```
.box {
  width: 100px;
  padding: 20px;
  border: 5px solid black;
}
```

Actual width = 100 + 20 + 20 + 5 + 5 = 150px

## Use box-sizing: border-box

With CSS3, we can fix this using:

```
* {
  box-sizing: border-box;
}
```

Now, the width includes padding and border.
If you say width is 100px, it stays 100px — even with padding or borders.
This makes layout simpler and more accurate.

# Introduction to Flexbox

Flexbox is a new way to arrange elements in a row or column — easily and responsively.

Flexbox is smart — it adjusts automatically for screen sizes!

```
.container {
  display: flex;
}
```

## Flex Direction (Row or Column)

Flexbox arranges items in a row by default. You can change this with flex-direction.

```
.container {
  display: flex;
  flex-direction: row;
}

.container-column {
  display: flex;
  flex-direction: column;
}
```

| **Value** | **What it does** |
|---|---|
| row | Horizontal (left to right) |
| column | Vertical (top to bottom) |

## Justify Content (Main Axis)

justify-content controls how items are spaced horizontally (when in a row).

```
.container {
  display: flex;
  justify-content: center;
}
```

| Value | What It Does |
|---|---|
| flex-start | All items go to the left |
| flex-end | All items go to the right |
| center | All items are centered |
| space-between | Space between items only |
| space-around | Space around each item |

## Align Items (Cross Axis)

align-items controls how items line up vertically (in a horizontal row).

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

| Value | What It Does |
|---|---|
| stretch | Stretches items to fill the height |
| center | Aligns items to the middle vertically |
| flex-start | Aligns to the top |
| flex-end | Aligns to the bottom |

## Flex Item Properties

Each item inside a flex container can also control how much space it takes.

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

| **Property** | **What It Does** |
|---|---|
| flex-grow | How much the item grows (like filling space) |
| flex-shrink | How much it can shrink if needed |
| flex-basis | The item’s starting size before growing |
| align-self | Aligns one item differently from others |

## Common Flexbox Layout – Horizontal Menu

Goal: Items stay in a row and space out evenly.

```
.menu {
  display: flex;
  justify-content: space-between;
}
```

*Visual:*

*| Home | About | Services | Contact |*

## Flexbox – Centering Anything

Flexbox makes centering very easy — both horizontally and vertically.

```
.center-box {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 300px;
}
```

Use this for:

- Login forms
- Splash screens
- Loading indicators

# Flex Wrap (Multiline Rows)

By default, Flexbox puts all items on one line. Use flex-wrap to make items wrap to next line.

```
.container {
  display: flex;
  flex-wrap: wrap;
}
```

Now if items are too many, they move to the next row.

Good for:

- Image galleries
- Button groups
- Responsive cards

## CSS Fonts and Text Styling

- CSS provides powerful control over text appearance:

Common Properties:

- font-family: Sets the font type (Arial, Georgia, etc.)
- font-size: Sets the size (e.g., 16px, 1.2em, 120%)
- font-style: Options: normal, italic, oblique
- font-weight: Options: normal, bold, lighter, numeric (100-900)

## Example:

```
p {
    font-family: 'Verdana', sans-serif;
    font-size: 14px;
    font-style: italic;
    font-weight: bold;
}
```

*Unit Types for font-size:*

*px, em, rem, %, pt, cm, in*

# CSS Text Formatting Properties

| Property | Description |
|---|---|
| color | Sets the color of text |
| text-align | Aligns text (left, right, center, justify) |
| text-decoration | Adds decorations (underline, line-through, etc.) |
| text-transform | Controls capitalization (uppercase, lowercase) |
| letter-spacing | Sets space between letters |
| line-height | Sets vertical spacing between lines of text |
| text-indent | Indents the first line of text |

## The color Property in Depth

- Sets the text color for an element
- Accepts color names, hex codes, rgb(), and hsl()

Example:

```
p { color: red; }
span { color: #ff0000; }
h1 { color: rgb(255, 0, 0); }
```

# The color Property in Depth

| Format | Example | Notes |
|---|---|---|
| **Named** | red | Built-in color name |
| **Hex** | #ff0000 | Common format for exact control |
| **RGB** | rgb(255, 0, 0) | Red/Green/Blue in 0–255 |
| **RGBA** | rgba(255, 0, 0, 0.5) | Includes alpha (opacity 0–1) |
| **HSL** | hsl(0, 100%, 50%) | Hue, Saturation, Lightness |
| **HSLA** | hsla(0, 100%, 50%, 0.6) | HSL with opacity |

## CSS Text Shadow and Effects

- Text Shadow: Adds shadow behind text.

Example:

```
h1 {
    text-shadow: 2px 2px 4px #999999;
}
```

Explanation:
2px 2px = horizontal and vertical offset
4px = blur radius
#999999 = shadow color

# CSS Background Properties

| Property | Description |
|---|---|
| background-color | Sets background color |
| background-image | Sets image as background |
| background-repeat | Repeat image (default) or stop repetition |
| background-position | Position image (e.g., center top, right bottom) |
| background-size | Scale image (cover, contain, or specific size) |
| background-attachment | scroll or fixed |
| | |

## Background Shorthand Property

Instead of writing many background properties, you can use a shorthand:

```
body {
  background: #ffffff url('img_tree.png') no-repeat right top;
}
```

**Order of values:**

1. background-color
2. background-image
3. background-repeat
4. background-attachment
5. background-position

Shorthand saves space and improves code readability.

## Borders in CSS

| Property | Description |
|---|---|
| border-width | Sets thickness of border |
| border-style | Solid, dashed, dotted, etc. |
| border-color | Sets the color of the border |
| border | Shorthand for all three |

### Shorthand Usage:

*border-top: 1px solid black;*

*border-left: 3px dotted red;*

## Display Property

| **Value** | **Meaning** |
|---|---|
| block | Element takes full width (e.g., div) |
| inline | Element fits content size (e.g., span) |
| inline-block | Inline but allows box styling |
| none | Hides the element |
| flex | Flexible layout container |

# CSS Positioning Basics

| **Value** | **Description** |
|---|---|
| static | Default. Normal document flow |
| relative | Positioned relative to normal position |
| absolute | Positioned relative to nearest positioned ancestor |
| fixed | Stays fixed in viewport (even on scroll) |
| sticky | Scrolls with content until a threshold, then sticks |

## CSS Float Property

The float property is used to position elements to the left or right and allow text or inline elements to wrap around them.
Values: left, right, none, inherit

- Allows text to wrap around an image or div.
- Often used in multi-column layouts or sidebars.

```
img {
  float: right;
  margin: 10px;
}
```

## The clear Property

When elements follow a floated element, they may wrap around it unintentionally. The clear property prevents this.
Values: left, right, both, none

```
div.clearfix {
  clear: both;
}
```

*Use clear after floating elements to avoid layout issues.*

## Overflow Property

Controls what happens when content overflows an element's box.

Values: visible, hidden, scroll, auto

- visible shows overflow.
- hidden clips it.
- scroll adds scrollbars.
- auto adds scrollbars only if needed.

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

# z-index and Stacking Context

z-index controls the stacking order of overlapping elements. Higher z-index values appear on top. Only works with positioned elements (relative, absolute, fixed, sticky).

```
.box1 {
  position: absolute;
  z-index: 1;
}
.box2 {
  position: absolute;
  z-index: 10;
}
```