**3. CSS Box Model**

The CSS Box Model is the layout foundation of every HTML element on a webpage.

Every element (like <div>, <p>, <button>, etc.) is treated as a **box** made up of four parts:

• Content

• Padding

• Border

• Margin

These layers define how elements are spaced, sized, and displayed in relation to each other.

The box model is crucial for layout and design.

┌─────────────────────────────────────┐

│             Margin                  │  ⬅ Outermost spacing

│ ┌─────────────────────────────────┐ │

│ │           Border               │ │

│ │ ┌─────────────────────────────┐ │

│ │ │         Padding            │ │

│ │ │ ┌─────────────────────────┐ │ │

│ │ │ │        Content          │ │ │

│ │ │ └─────────────────────────┘ │ │

│ │ └─────────────────────────────┘ │

│ └─────────────────────────────────┘ │

└─────────────────────────────────────┘

**Box and Spacing Properties**

**Property**

**What it does**

**Example**

width

Width of the element

width: 200px;

height

Height of the element

height: 100px;

padding

Space inside the box

padding: 10px;

margin

Space outside the box

margin: 20px;

border

Adds border

border: 2px solid black;

box-sizing

Includes border/padding in size

box-sizing: border-box;

• Content

This is the actual content inside the box — where text, images, buttons, etc. appear.

*Example:*

button {

    width: 200px;

    height: 100px;

}

• Padding

Padding is the space between the content and the border.

*Example:*

div {

    padding: 10px;

}

This increases space **inside** the box — it pushes the content **inward**.

You can control sides individually:

div {

   padding-top: 10px;

   padding-right: 5px;

   padding-bottom: 15px;

   padding-left: 5px;

}

Or use shorthand:

div {

    padding: 10px 5px 15px 5px;     /\* top right bottom left \*/

}

• Border

The edge of the box surrounding the padding and content. It goes around the padding and content.

*Example:*

div {

    border: 1px solid #000;

}

You can control:

div {

    border-width: 1px;

             border-style: solid;

             border-color: #000;

}

• Margin

Margin is the **space outside the border**, separating one box from another.

*Example:*

div {

    margin: 20px;

}

Like padding, you can control individual sides:

div {

    margin-top: 20px;

    margin-bottom: 30px;

}

HTML

<body>

  <div class="margin">

    Margin

    <div class="border">

      Border

      <div class="padding">

        Padding

        <div class="content">

          Content

        </div>

      </div>

    </div>

  </div>

</body>

CSS

body {

  background-color: #f5f5f5;

  font-family: Arial, sans-serif;

  text-align: center;

  padding-top: 50px;

}

.margin {

  background-color: #ffe4e1;

  width: 400px;

  margin: 0 auto;

  padding: 20px;

}

/\* Border Area \*/

.border {

  background-color: #add8e6;

  border: 5px solid #00008b;

  padding: 20px;

}

/\* Padding Area \*/

.padding {

  background-color: #90ee90;

  padding: 20px;

}

/\* Content Area \*/

.content {

  background-color: #ffffe0;

  padding: 15px;

  border: 1px dashed gray;

}

HTML

<div class="card">

  <h1>This is a card.</h1>

  <p>Lorem……</p>

</div>

CSS

.card {

  width: 200px;

  padding: 20px;

  margin: 30px;

  border: 2px solid darkblue;

  background-color: lightyellow;

}

**The Role of Box Sizing:    box-sizing**

Box-sizing specifies how the total width and height of an element are calculated.

By default, **padding and border** are added **outside** the content’s width/height.

*Example:*

.card {

    width: 200px;

    padding: 20px;

}

→ Total width = 200px + 20px (left) + 20px (right) = **240px**

**Solution:** {box-sizing: border-box }

This tells the browser to **include** padding and border within the set width/height.

.card {

    box-sizing: border-box;

}

HTML

<div class="profile-card">

  <h2>Student Name</h2>

  <p>Class: JSS2</p>

</div>

CSS

.profile-card {

  width: 250px;

  padding: 20px;

  margin: 30px auto;

  border: 2px dashed green;

  background-color: #f0fff0;

  box-sizing: border-box;

  text-align: center;

}

HTML

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <link rel="stylesheet" href="style.css">

    <title>Profile Card</title>

</head>

<body>

    <!-- Profile Card -->

    <div class="profile-card">

        <div class="profile-image">

            <img src="profile-photo.jpg" alt="Profile Picture">

        </div>

        <h1 class="profile-name">Sherif Abdul</h1>

        <p class="profile-location">Lagos, Nigeria</p>

        <p class="profile-bio">"Software Developer"</p>

        <div class="social-links">

            <a href="" target="\_blank" class="social-button">GitHub</a>

            <a href="" target="\_blank" class="social-button">Frontend Mentor</a>

            <a href="" target="\_blank" class="social-button">LinkedIn</a>

            <a href="" target="\_blank" lass="social-button">Twitter</a>

        </div>

    </div>

</body>

</html>

CSS

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    font-family: 'Arial', sans-serif;

}

body {

    background-color: #121212;

    display: flex;

    justify-content: center;

    align-items: center;

    min-height: 100vh;

    position: relative;

    overflow: hidden;

}

.profile-card {

    background-color: #1e1e1e;

    border-radius: 10px;

    padding: 30px;

    width: 100%;

    max-width: 350px;

    text-align: center;

}

.profile-image {

    width: 100px;

    height: 100px;

    border-radius: 50%;

    background-color: #333;

}

.profile-image img {

    width: 100%;

    height: 100%;

    object-fit: cover;

}

.profile-name {

    color: white;

    font-size: 24px;

    margin-bottom: 5px;

}

.profile-location {

    color: #b9e769;

    font-size: 14px;

    margin-bottom: 15px;

}

.profile-bio {

    color: #cccccc;

    font-size: 14px;

    margin-bottom: 25px;

}

.social-links {

    display: flex;

    flex-direction: column;

    gap: 10px;

}

.social-button {

    background-color: #333333;

    color: white;

    padding: 12px;

    border-radius: 5px;

    text-decoration: none;

    transition: background-color 0.3s;

    font-size: 14px;

}

.social-button:hover {

    background-color: #444444;

}

**Practical Exercise:** Implementing the Box Model

Using the box model properties to style a set of boxes in an HTML document.

Instructions:

Open your text editor and create a new HTML file named box\_model.html.

Write the following HTML code:

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <title>Box Model Example</title>

    <link rel="stylesheet" href="box\_styles.css">

</head>

<body>

    <div class="box">

        <p>Box 1</p>

    </div>

    <div class="box">

        <p>Box 2</p>

    </div>

    <div class="box">

        <p>Box 3</p>

    </div>

</body>

</html>

Create an external CSS file named box\_styles.css with the following content:

.box {

    width: 200px;

    padding: 20px;

    border: 2px solid #000;

    margin: 10px;

    background-color: #ccc;

    box-sizing: border-box;

}

.box p {

    text-align: center;

    margin: 0;

}

**4. CSS Layout Techniques**

**Introduction to CSS Layout**

CSS layout determines how elements are arranged on a webpage. Modern CSS offers several layout models

**• Display Property:** block, inline, inline-block and none

**• Positioning**

**• Flexbox Layouting** (1-dimensional layouts)

**• Grid Layouting**  (2-dimensional layouts)

And after looking at the above we then look at

**• Responsive Design** (adapts to screen sizes)

**1 Display Property**

The display property in CSS determines how an element is rendered in the document. It defines the type of box used for an element and affects layout behaviour.

**Common Values**

• display: block;

◦ Takes full width, starts on a new line.

Examples: <div>, <p>, <h1>-<h6>.

• display: inline;

◦ Occupies only necessary width, stays in line with text.

Examples: <span>, <a>, <strong>.

• display: inline-block;

◦ Combines inline and block: stays in line but allows width/height adjustments.

• display: none;

◦ Removes the element from the layout (not rendered).

• display: flex; & display: grid;

◦ Enables Flexbox and Grid layout models (discussed later).

Note: Flexbox and Grid are the backbone of modern responsive design.

*Examples:*

.block-element {

  display: inline;     /\* Now behaves like a <span> or <a> \*/

}

<div class="inline-div">This is now inline!</div>

<p class="inline-p">This paragraph is inline too.</p>

.inline-div, .inline-p {

  display: inline;

  background: lightblue;

}

.box {

  display: inline-block;

  width: 100px;

  height: 100px;

}

<span class="block-span">This span is now block-level!</span>

<a class="block-link" href="#">This link is a block now.</a>

.block-span, .block-link {

  display: block;

  background: lightcoral;

  margin: 5px 0;

}

<div class="box">Box 1</div>

<div class="box">Box 2</div>

.box {

  display: inline-block;

  width: 100px;

  height: 100px;

  background: lightgreen;

}

**Why Layout Techniques Matter**

Before Flexbox and Grid, layout used to rely on:

• Tables ❌

• Float hacks ❌

• Complicated position tricks ❌

Now, Flexbox and Grid make things easy for us to:

• Organize content

• Make responsive websites

• Build real-world layouts (navbars, galleries, forms, etc.)

**1. INTRODUCTION TO CSS FLEXBOX  — One-Dimensional Layout**

**What is Flexbox?** *flexboxfroggy.com*

Flexbox also known as the Flexible Box Model is a one-dimensional layout model that provides a method to offer space distribution and it gives us alignment capabilities.

• Before Flexbox, we aligned elements using floats and tables. Flexbox has made life easier for people using CSS.

• CSS Grid is another option for alignment, display, etc.

Flexbox best for layouts in a single direction (row or column)

Flexbox helps you arrange things in one direction at a time —either horizontally (row) or vertically (column).

**Flexbox** is good when you are laying items in a straight line or side by side.

*Example with Flexbox:*

• All menu/nav items in a **row** (horizontally)

• Or stacking form fields in a **column** (vertically)

Think of Flexbox like a straight road — you move along it either left-right or up-down, but not both at the same time.

(Row direction →)      Item 1     Item 2     Item 3     Item 4

(↓ Column direction)

Item 1

Item 2

Item 3

Item 4

**Basic Flexbox Concepts**

**Flex Container:**

The flex container is the parent element that holds flex items where “display: flex;” is applied.

**Flex Items:**

Flex items are the **direct** children of the flex container.

*Example:*

  <div class="flex-container">

      1. <div>Flex Item 1</div>

      2. <div>Flex Item 2</div>

  </div>

**Setting Up Flexbox**

.container {

  display: flex;

}

This turns all direct children into flex items.

**Properties of Flex Containers and Flex Items**

FLEX CONTAINER

• Flex-direction

• Flex-wrap

• Flex-flow

• Justify-content

• Align-items

• Align-content

• gap

FLEX ITEM

• order

• Flex-grow

• Flex-shrink

• Flex-basis

• Align-self

**Main Flexbox Properties**

**Property**

**Applied To**

**Purpose**

display: flex

Parent

Enables flexbox layout

flex-direction

Parent

Set layout direction: row, column

justify-content

Parent

Align items horizontally

align-items

Parent

Align items vertically

flex-wrap

Parent

Allow items to wrap to next line

flex

Child

Set how items grow/shrink inside flex

**Flex Direction**

Defines the direction in which the flex items are placed in the flex container.

row (default), row-reverse, column, column-reverse.

.container {

  flex-direction: row;

}

**Justify Content**

Aligns flex items along the main axis.

flex-start, flex-end, center, space-between, space-around.

.container {

  justify-content: center;

}

**Align Items**

Aligns flex items along the cross axis.

flex-start, flex-end, center, baseline, stretch.

.container {

  align-items: center;

}

**Align Self**

Aligns individual flex items along the cross axis.

auto, flex-start, flex-end, center, baseline, stretch.

.container {

  align-self: flex-end;

}

*Example:*

**HTML**

<div class="container">

  <div class="item one">One</div>

  <div class="item two">Two</div>

  <div class="item three">Three</div>

  <div class="item four">Four</div>

  <div class="item five">Five</div>

</div>

**CSS**

.container {

  display: flex;

  justify-content: space-between;

  align-items: center;

  background-color: lightblue;

  padding: 20px;

}

.item {

  background-color: white;

  padding: 10px 20px;

  border: 1px solid #ccc;

}

**Common justify-content values:**

• flex-start — items aligned to the left

• center — items in the middle

• space-between — even spacing between items

• space-around — space around each item

• space-evenly — equal space between and around

**2. INTRODUCTION TO CSS GRID — Two-Dimensional Layout**

**What is CSS Grid?**

CSS Grid is a powerful layout system in CSS that allows you to create complex web layouts with ease. It offers a two-dimensional grid-based layout system, meaning you can control both columns and rows, making it a more robust option compared to other layout methods.

Best for building entire page layouts, grids, or anything with rows and columns.

**Comparison with Other Layout Systems:**

**Flexbox**: Primarily one-dimensional, focusing either on columns or rows. Ideal for components in a single row or column.

**Grid** helps you arrange things **in two directions at the same time** — **both rows and columns** together.

**Grid** is good when you are building **complex layouts** like:

• Pages with multiple sections

• Product cards in rows and columns

• Photo galleries

(Row 1)  Item 1 | Item 2 | Item 3

(Row 2)  Item 4 | Item 5 | Item 6

(Row 3)  Item 7 | Item 8 | Item 9

Each **row** and **column** is controlled separately.

**Common Grid Properties**

**Property**

**Purpose**

display: grid

Activates grid layout

grid-template-columns

Define number and size of columns

grid-template-rows

Define number and size of rows

grid-gap / gap

Space between items

grid-column / grid-row

Span items across columns or rows

place-items

Center items horizontally and vertically

**Setting Up a Grid**

Creating Columns. You can also create rows and columns together.

Apply display: grid to the container.

.grid-container {

  display: grid;

  grid-template-columns: 1fr 1fr 1fr;

  gap: 10px;

}

• 1fr = 1 fraction of available space

• gap = spacing between grid items

**Fractional Units (fr), Pixels (px), Percentages (%), and Auto:**

**fr**: Fractional unit of available space.

**px**: Fixed size.

**%**: Percentage of the parent container.

**auto**: Adjusts size based on content.

**Using grid-template-columns and grid-template-rows:**

.container {

  display: grid;

  grid-template-columns: 1fr 1fr 1fr;

  grid-template-rows: 100px 1fr 50px;

}

**200px 1fr 200px**: Creates three columns with the middle column taking up the remaining space.

**100px 1fr 50px**: Creates three rows with the middle row expanding to fill the remaining space.

**Placing Grid Items -On a child element**

grid-column and grid-row Properties:

.item {

  grid-column: 1 / 3;

  grid-row: 2 / 4;

}

**1 / 3**: Starts at column line 1 and ends before column line 3.

**2 / 4**: Starts at row line 2 and ends before row line 4.

**grid-gap, row-gap, and column-gap:**

.container {

  grid-gap: 10px;

  row-gap: 15px;

  column-gap: 20px;

}

**Example:**

<div class="grid-container">

  <div class="box">1</div>

  <div class="box">2</div>

  <div class="box">3</div>

</div>

.grid-container {

  display: grid;

  grid-template-columns: repeat(3, 1fr);

  gap: 10px;

  background-color: #eee;

  padding: 20px;

}

.box {

  background-color: teal;

  color: white;

  padding: 30px;

  text-align: center;

  font-size: 20px;

}

**Example:** Basic Grid with Explicit Rows & Columns

Html

<h1 class="title">My Photo Gallery</h1>

<div class="container">

  <div class="photo">1</div>

  <div class="photo">2</div>

  <div class="photo">3</div>

  <div class="photo">4</div>

  <div class="photo">5</div>

  <div class="photo">6</div>

</div>

CSS

body {

  font-family: Arial, sans-serif;

  text-align: center;

  background-color: #f4f4f4;

  padding: 20px;

}

.title {

  font-size: 2rem;

  margin-bottom: 30px;

}

.gallery {

  display: grid;

  grid-template-columns: repeat(3, 1fr); /\* 3 equal columns \*/

  gap: 20px;

  max-width: 800px;

  margin: auto;

}

.photo {

  background-color: #4ce19e;

  color: white;

  font-size: 2rem;

  padding: 50px 0;

  border-radius: 10px;

  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

}

<div class="container">

  <div class="photo">1</div>

  <div class="photo">2</div>

  <div class="photo">3</div>

  <div class="photo">4</div>

  <div class="photo">5</div>

  <div class="photo">6</div>

</div>

.container {

  display: grid;

  grid-template-columns: 100px 200px 1fr; /\* 3 columns: fixed, fixed, flexible \*/

  grid-template-rows: 50px 100px; /\* 2 explicit rows \*/

  gap: 10px; /\* Spacing between items \*/

}

.item {

  background: coral;

  padding: 10px;

}

.container {

  display: grid;

  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

  grid-auto-rows: 200px;

  gap: 15px;

}

**Extras:**

Grid Item Properties

Placement

Property

Values

Description

grid-column-start

1, span 2, area-name

Start position in columns

grid-column-end

3, span 2

End position in columns

grid-row-start

Line number or name

Start position in rows

grid-row-end

Line number or name

End position in rows

grid-column

1 / 3 or 1 / span 2

Shorthand for column start/end

grid-row

2 / 4

Shorthand for row start/end

grid-area

header or 1 / 1 / 3 / 3 (row-start/col-start/row-end/col-end)

Super shorthand or area name

**Responsive Grid Layouts with repeat, auto-fit, and auto-fill:**

.container {

  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));

}

**auto-fit**: Fits as many items as possible in the container.

**auto-fill**: Fills the container with items, even if they don't fit perfectly.

**Responsive Image Gallery:**

.gallery {

  display: grid;

  grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));

  gap: 10px;

}

.gallery img {

  width: 100%;

  height: auto;

}