Sheryians Coding School

# Live Cohort

## Notes Day 5

# Day 2 : More About CSS

## Understanding Background Properties

CSS provides various background properties to enhance the visual appeal of web pages. These properties allow developers to control the background image, size, positioning, and repeating behavior.

### Background-size

Defines how a background image is scaled within an element.

- background-size: cover; → The image covers the entire element, maintaining its aspect ratio.
- background-size: contain; → The image scales to fit inside the element while maintaining its aspect ratio.
- background-size: 50% 50%; → The image is resized to 50% of the width and height of the element.

Example::

```
body {
  background-image: url('image.jpg');
  background-size: cover;
}
```

# Day 2 : More About CSS

## Background-Image

Used to set an image as the background of an element.

Example:

```
div {
  background-image: url('example.jpg');
  background-size: cover;
}
```

# Day 2 : More About CSS

## Background-Repeat

Controls whether the background image repeats and how.

- background-repeat: repeat; (default) → Image repeats both horizontally and vertically.
- background-repeat: no-repeat; → Image does not repeat.
- background-repeat: repeat-x; → Image repeats horizontally.
- background-repeat: repeat-y; → Image repeats vertically.

Example:

```
div {
  background-image: url('pattern.png');
  background-repeat: no-repeat;
}
```

# Day 2 : More About CSS

## Background-Position

Defines the starting position of a background image.

- background-position: center; → Centers the image.
- background-position: top left; → Positions the image at the top-left.
- background-position: 50% 50%; → Places the image in the middle.

Example:

```
div {
  background-image: url('example.jpg');
  background-position: center;
}
```

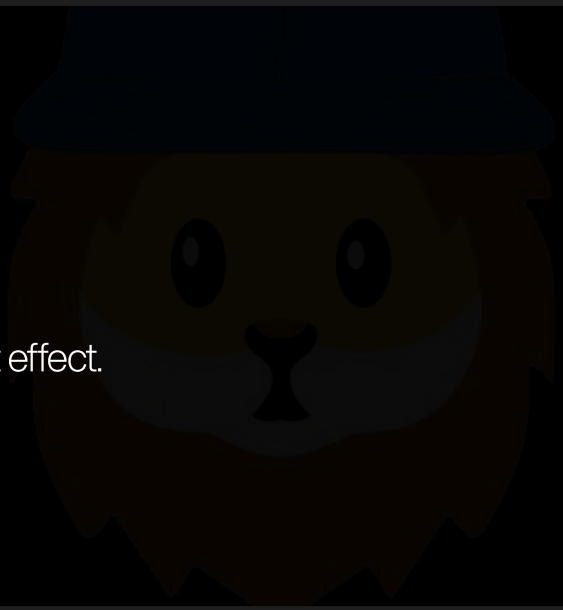
# Day 2 : More About CSS

## Linear-Gradient

Creates a smooth transition of colors in a straight line.

Example:

```
div {
  background: linear-gradient(to right, red, blue);
}
```



## Radial-Gradient

Creates a circular gradient effect.

Example:

```
div {
  background: radial-gradient(circle, red, blue);
}
```

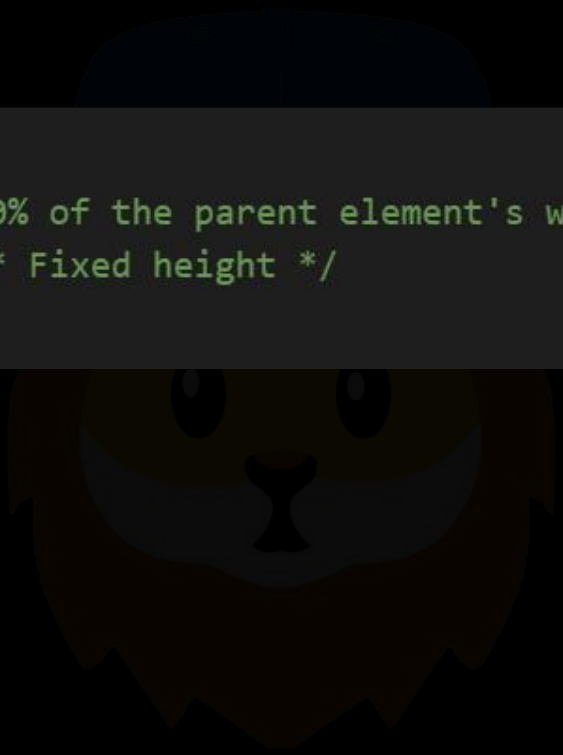
# Day 2 : More About CSS

## Px Vs %

- px (Pixels): A fixed unit of measurement. Useful when you need precise control over an element's size.
- % (Percentage): Relative to the parent container. Useful for responsive designs.

Example:

```
div {
  width: 50%; /* 50% of the parent element's width */
  height: 200px; /* Fixed height */
}
```



# Day 2 : More About CSS

## Working With Positional Properties

CSS position properties define how an element is positioned within its container.

## Position: Absolute

Removes the element from the normal document flow and positions it relative to the nearest positioned ancestor (or the viewport if no positioned ancestor exists).

Example:

```
div {
  position: absolute;
  top: 50px;
  left: 100px;
}
```

# Day 2 : More About CSS

## Position: Relative

Positions an element relative to its normal position.

Example:

```
div {
  position: relative;
  top: 20px;
  left: 30px;
}
```

## Transform: Translate

Moves an element without affecting surrounding elements.

Example:

```
div {
  transform: translate(50px, 100px);
}
```

# Day 2 : More About CSS

## Introduction To Flexbox For Alignment & Structure

Flexbox is a powerful layout system in CSS that helps in aligning and distributing elements efficiently.

### 1. display: flex

Defines a flex container, making its child elements flexible.

**Example:**

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

### 2. flex-direction

Specifies the direction of the flex items.

- `row` (default) → Items align horizontally.
- `column` → Items align vertically.

**Example:**

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

# Day 2 : More About CSS

## Introduction To Flexbox For Alignment & Structure

### 3. flex-wrap

Controls whether flex items wrap onto multiple lines.

- `nowrap` (default) → Items stay on a single line.
- `wrap` → Items wrap to the next line if needed.

**Example:**

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

### 4. flex-shrink

Determines how much a flex item can shrink if needed.

**Example:**

```
.item {
  flex-shrink: 2; /* Shrinks twice as fast as others */
}
```

# Day 2 : More About CSS

## Introduction To Flexbox For Alignment & Structure

### 5. justify-content

Aligns flex items along the main axis.

- `flex-start` → Items start from the beginning.
- `center` → Items are centered.
- `flex-end` → Items align at the end.
- `space-between` → Equal space between items.
- `space-around` → Equal space around items.

**Example:**

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

# Day 2 : More About CSS

## Introduction To Flexbox For Alignment & Structure

### 6. align-items

Aligns flex items along the cross-axis.

- `flex-start` → Items align at the top.
- `center` → Items align in the middle.
- `flex-end` → Items align at the bottom.

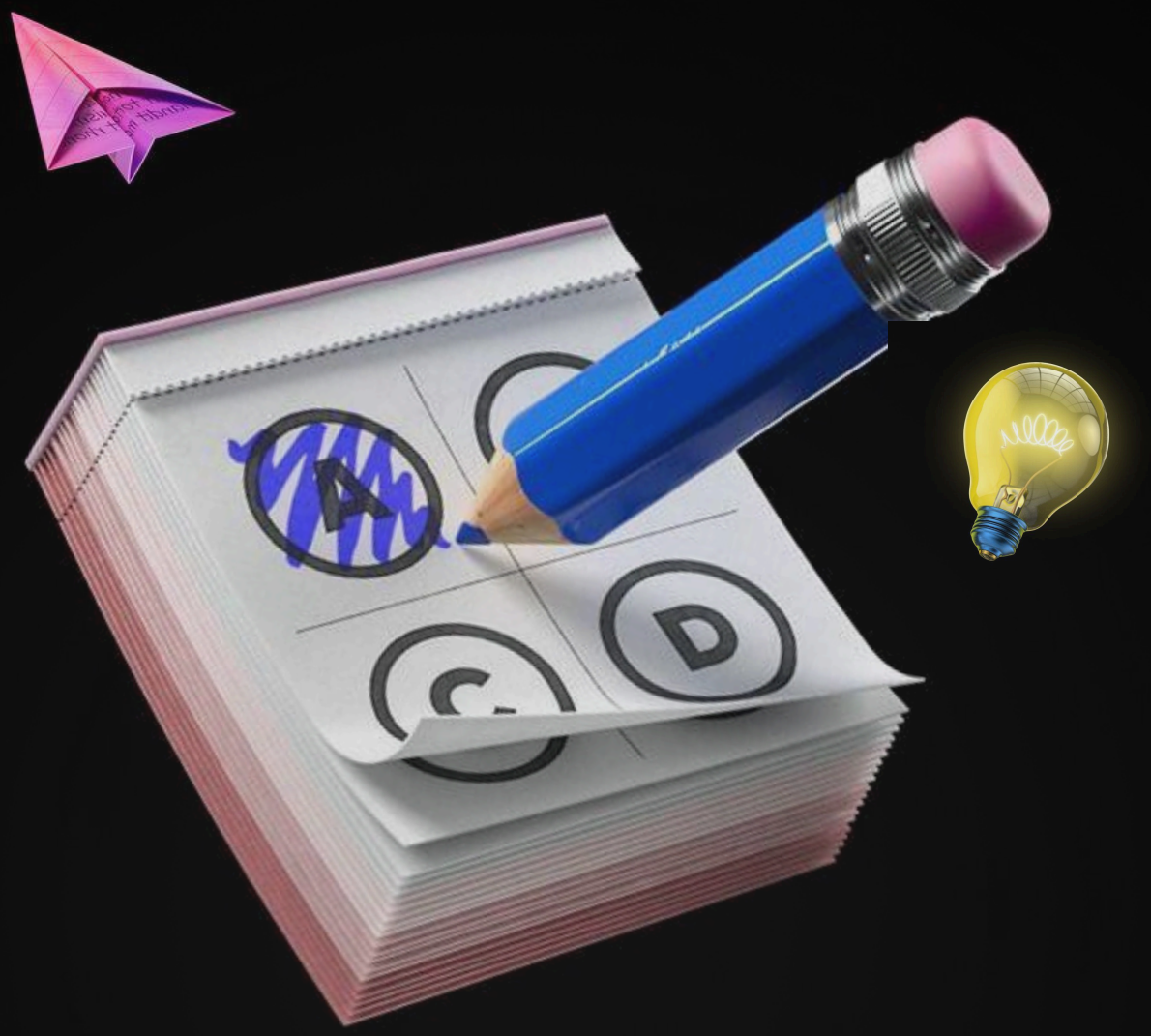
**Example:**

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

Sheryians Coding School

# Project Exercise 4

## Two-Column Layout with Flexbox



# Project Exercise 4: Two-Column Layout with Flexbox

## Delicious Coffee

Home About Products Contact

**Explore More**

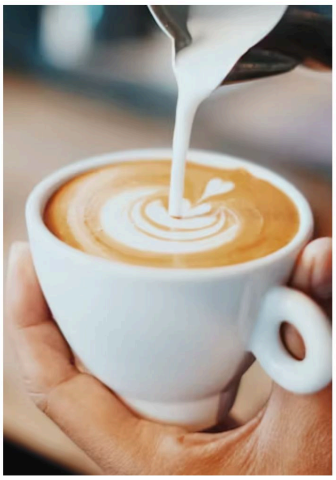
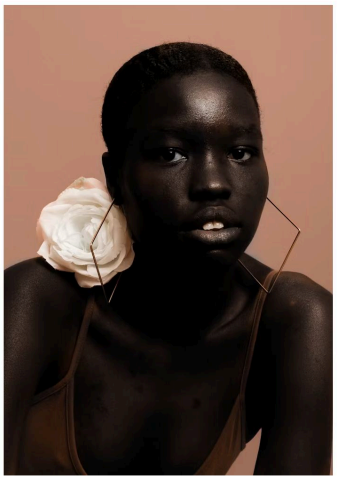
Our Story

Coffee Beans

Brewing Guides

Visit Us

### Welcome to Delicious Coffee!


We are passionate about bringing you the most delicious and ethically sourced coffee beans from around the world. Our coffee is roasted to perfection to bring out the unique flavors and aromas of each bean origin.

### Our Featured Coffees

- **Ethiopian Yirgacheffe:** Bright and floral with citrus notes.
- **Sumatran Mandheling:** Earthy and full-bodied with chocolate undertones.
- **Colombian Supremo:** Well-balanced and smooth with caramel sweetness.

Learn more about our brewing process, our commitment to sustainability, and our team of coffee experts on our About Us page.

Thank you for choosing Delicious Coffee!

# Project Exercise 3: Styling The Page

What TO Do:

- HTML First, Simple Structure: Create basic HTML with divs (header, nav, sidebar, main, footer) and placeholder text.
- Link CSS: Connect styles.css to your HTML in <head>.
- display Exploration: Try display: block, inline, inline-block, none, flex, grid on elements. See the changes.
- Flexbox for 2 Columns: Use display: flex on body. Experiment with flex-direction, justify-content, align-items. flex-grow: 1 for main content.
- Use Inspect Element: Crucial! Use browser tools to see CSS and experiment live.
- One Property at a Time: Change one CSS rule, then check the browser.
- Read Solution Comments: Understand why the solution code works.

Ask Questions: Get help when stuck!

What NOT to Do:

- Don't Start Complex: Focus on layout basics first, styling later.
- Don't Overwhelm Yourself: Focus on display, Flexbox, positioning, overflow for now.
- Don't Skip Inspect Element: It's essential for learning.
- Don't Fear Experimenting: Breaking things is part of learning.
- Don't Just Copy Solution: Try yourself first, then understand the solution.