Fundamentals of Tailwind CSS

Engage and Think

A tech firm is building a new web platform and must ensure the user interface is both visually appealing and responsive. The team is evaluating styling methods to streamline development and maintain design consistency.

What key considerations should influence the choice of an approach for structuring and managing styling in a large-scale web development project?

Learning Objectives

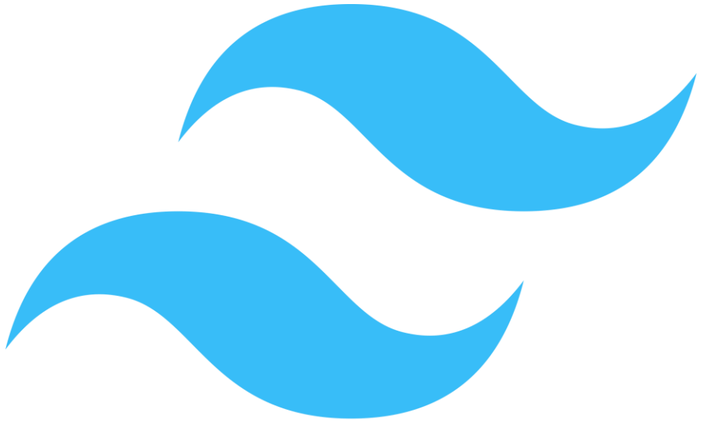
By the end of this lesson, you will be able to:

- Identify common use cases where Tailwind CSS improves web design and development
- Demonstrate how to set up Tailwind CSS using a CDN link in a sample HTML file
- Utilize Tailwind Flex utilities to align elements both vertically and horizontally
- Develop a custom design system and adjust text size with utility classes for optimal readability using Tailwind

Introduction to Tailwind

What Is Tailwind?

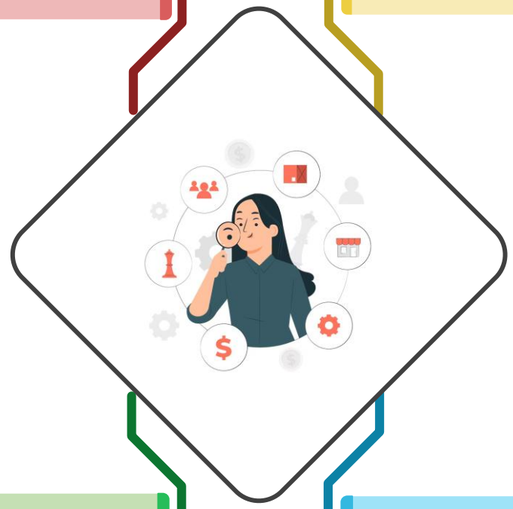
It is an open-source CSS framework that provides a utility-first approach to styling web applications.



Tailwind offers a collection of pre-defined CSS classes that allow developers to quickly build modern, responsive designs while also enabling extensive customization.

Tailwind: Characteristics

1. Supports easy customization through configuration files
2. Enables rapid prototyping with minimal custom CSS



3. Offers built-in responsiveness using mobile-first breakpoints
4. Promotes consistency by using a design system approach

Why Use Tailwind CSS?

Tailwind CSS is a modern, utility-first framework that empowers developers to build responsive, customizable, and scalable designs with ease.

Following are key reasons why Tailwind CSS is a preferred choice for styling web applications:

- Reduces file size in production by purging unused CSS classes
- Integrates seamlessly with frameworks like React, Vue, and Angular
- Enhances maintainability with utility class naming over custom CSS rules

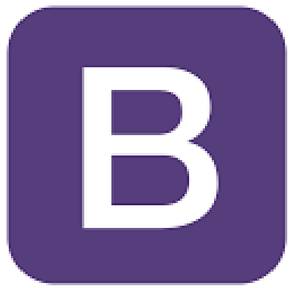
Tailwind: Use Cases

Following are key use cases where Tailwind UI can be effectively implemented to enhance the design and development of web applications:

- Speed up development with pre-defined utility classes
- Maintain consistency using configuration-driven design tokens
- Create responsive layouts directly in markup without switching files
- Minimize custom CSS by composing utilities
- Customize themes easily with Tailwind configuration
- Integrate seamlessly with modern frontend frameworks

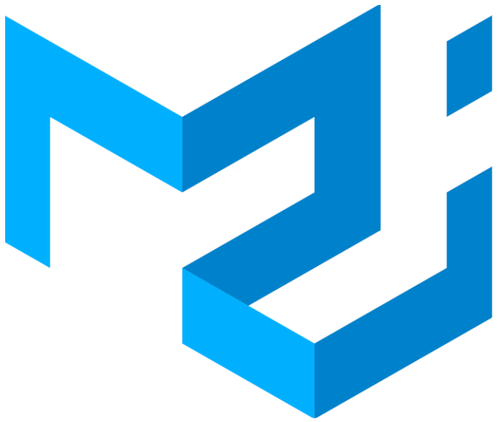
Overview of Other CSS Frameworks

Here is a brief overview of other popular CSS frameworks:



Bootstrap

Widely used



Material UI

Quite complex for beginners

Foundation

Limited community and adoption

Setting Up Tailwind CSS

There are multiple ways to set up Tailwind CSS based on your project needs, ranging from simple CDN usage to advanced build tool integrations.

- **CDN setup**: Use a script tag in HTML for quick and easy Tailwind access without installation
- **Tailwind CLI setup**: Install Tailwind using Node.js to enable customization and manual CSS builds
- **Build tool setup**: Integrate Tailwind with tools like Vite or Webpack for full control in production projects

Assisted Practice

Setting Up Tailwind Using a CDN Link **Duration: 10 Min.**

Problem statement:
You have been assigned a task to get the Tailwind UI CSS integrated within an html page without using node or any other package manager. Theme is to solely rely on inclusion of a CDN link within the head tag of the html file.

Outcome:
By the end of this demo, you will be able to include Tailwind UI using a CDN link, apply a basic h1 tag, and verify that Tailwind is working correctly in your project.

Note: Refer to the demo document for detailed steps:
01_Setting_Up_Tailwind_Using_a_CDN_Link

Assisted Practice: Guidelines

Steps to be followed:

1. Create an HTML file
2. Integrate Tailwind via CDN

Quick Check

Your team is building a modular dashboard with Tailwind CSS. As the project scales, class duplication is increasing, making design consistency hard to manage. What is the best way to improve maintainability?

A. Replace all Tailwind classes with inline styles for better visibility

B. Create reusable UI components that use consistent Tailwind utility class patterns

C. Use global CSS files for each module with hardcoded color codes

D. Minimize the use of Tailwind utilities and adopt framework-specific styling only

Key Features of Tailwind

Key Features of Tailwind CSS

Tailwind CSS offers key features that improve flexibility, speed, and consistency in modern web development. It includes:

01 **Responsive web design**: Ensures the UI adapts seamlessly to mobile and tablet screens

02 **Utilities**: Provides classes for margin, padding, sizing, and flexbox layout

03 **Customization**: Allows easy adjustments to colors, sizes, and layouts

Responsive Web Design

It is a development approach that enables websites to dynamically adjust their layout, content, and functionality based on the screen size, resolution, and device type.

Company retreats

Incredible accommodation for your team

Looking to take your team away on a retreat to enjoy awesome food and take in some sunshine? We have a list of places to do just that.

View: Mobile

Company retreats

Incredible accommodation for your team

Looking to take your team away on a retreat to enjoy awesome food and take in some sunshine? We have a list of places to do just that.

View: Laptop or a larger screen

This ensures a seamless user experience across mobile phones, tablets, laptops, and desktop computers by using flexible grids, media queries, and scalable images.

Source: https://tailwindcss.com/docs/responsive-design

Responsive Web Design

Below is a list of essential breakpoints that help achieve responsiveness:

Breakpoint prefix	Minimum width	CSS
sm	40rem (640px)	@media (width >= 40rem) { ... }
md	48rem (768px)	@media (width >= 48rem) { ... }
lg	64rem (1024px)	@media (width >= 64rem) { ... }
xl	80rem (1280px)	@media (width >= 80rem) { ... }

Utilities

Tailwind CSS provides a utility-first approach, meaning it offers a wide range of predefined utility classes that allow developers to style elements directly in the markup without writing custom CSS.

Below are some key utility categories in Tailwind CSS:

1. Apply predefined text and background color classes
2. Adjust font size, weight, spacing, and text alignment
3. Modify element width, height, and aspect ratio
4. Design responsive layouts with flexible and grid-based structures

Note:

Tailwind CSS includes over 30+ typography related utility classes for controlling font size, weight, letter spacing, line height, and text alignment.

Utilities

Below is the list of important utility classes:

Utility	Syntax	Explanation
Margin	`m-<0-96>, mt, mb, ms, me, mx, my`	Tailwind CSS margin utilities include: m-{0-96} (all sides), mt (top), mb (bottom), ms (start), me (end), mx (horizontal), my (vertical).
Padding	`p-<0-96>, pt, pb, ps, pe, px, py`	Tailwind CSS padding utilities include: p-{0-96} (all sides), pt (top), pb (bottom), ps (start), pe (end), px (horizontal), py (vertical).
Width	`w-<0-100>`	Tailwind CSS width utilities include: w-{0-100} (percentage-based width), w-auto (auto width), w-px (1px), w-full (100%), w-screen (viewport width), w-min (min-content), w-max (max-content), w-fit (fit-content).
Flex	`flex flex-row`	Tailwind CSS flex utilities include: flex (enables flexbox), flex-row / flex-col (direction), flex-wrap / flex-nowrap (wrapping), flex-{1, auto, initial, none} (flex-grow/shrink basis).

Customization

Customization allows users to modify and extend default styles while maintaining the core framework's benefits. It provides multiple methods to add custom styles efficiently.

Methods of customization are as follows:

NPM configuration

Modifies the **tailwind.config.js** file to define custom styles, themes, and utility classes

CDN approach

Uses the **:root** selector inside a **<style>** tag and applies styles inline using the style attribute

Custom CSS classes

Define styles within **:root** and reuse them by creating custom CSS classes

Assisted Practice

Demonstrating Custom Themes in Tailwind CSS **Duration: 10 Min.**

Problem statement:
Demonstrate text styling using Tailwind CSS by applying font sizes, colors, and alignment, and integrate custom CSS variables for compatibility with utility classes.

Outcome:
By the end of this demo, you will be able to apply different text sizes using Tailwind's font sizing utilities, use text color utilities to modify text appearance, align text using Tailwind's alignment classes, and verify that the styles are applied correctly in the project.

Note: Refer to the demo document for detailed steps:
02_Demonstrating_Custom_Themes_in_Tailwind_CSS

Assisted Practice: Guidelines

Steps to be followed:

1. Open Visual Studio Code and create index.html
2. Add custom theme related code

Overview of Typography

Typography

It follows a utility-based approach for styling text elements. Below are some most commonly used typography utilities:

- **Text sizing:** Control the font size of an element
- **Colour-shade system:** Control the text color of an element
- **Text alignment:** Control the alignment of text

Text Sizing

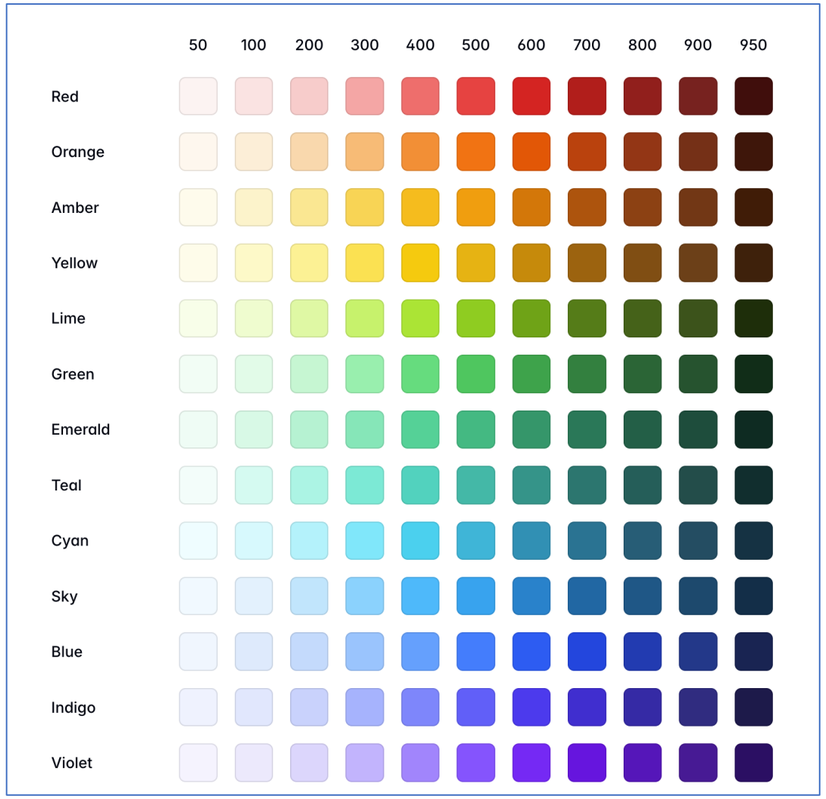
Below is a list of important text sizing classes along with their syntax and explanation:

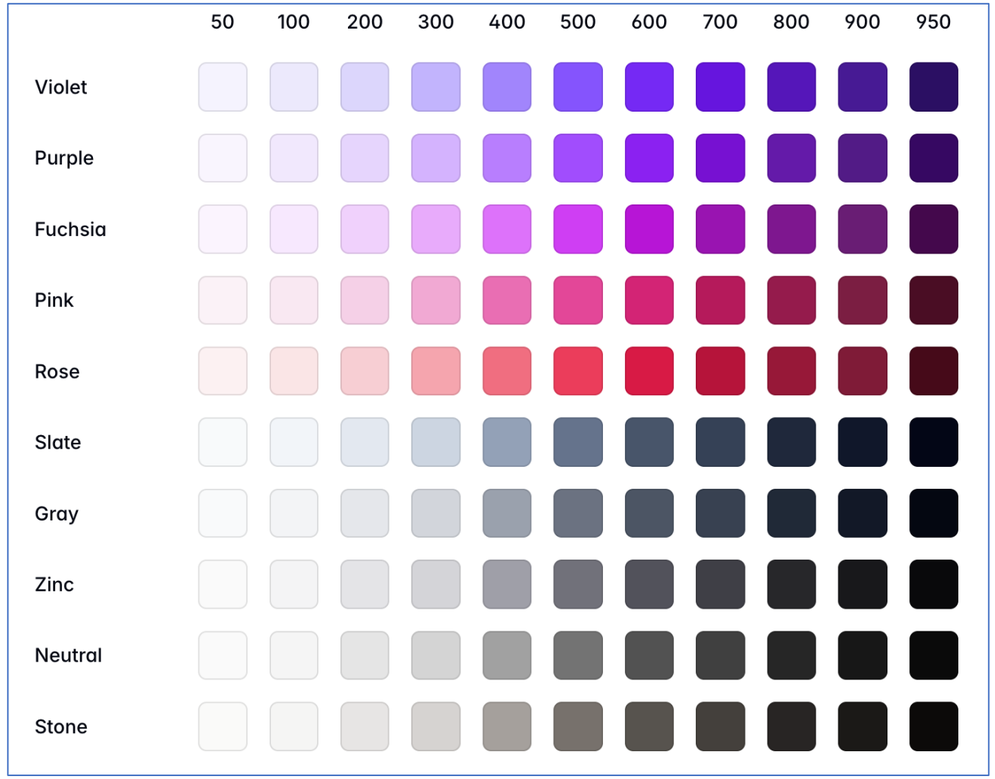
Class name	Syntax	Explanation
xs	`text-xs`	Extra small text size (12px / 0.75rem)
sm	`text-sm`	Small text size (14px / 0.875rem)
base	`text-base`	Default text size (16px / 1rem)
lg	`text-lg`	Large text size (18px / 1.125rem)
xl	`text-xl`	Extra large text size (20px / 1.25rem)
text-2xl	`text-<2-9>xl`	Scales up to 9xl (96px / 6rem max)

Colour-Shade System

It represents a structured approach in defining colors with varying intensity levels, helping maintain consistency in design.

Following is a colour-shade description chart:





Source: https://tailwindcss.com/docs

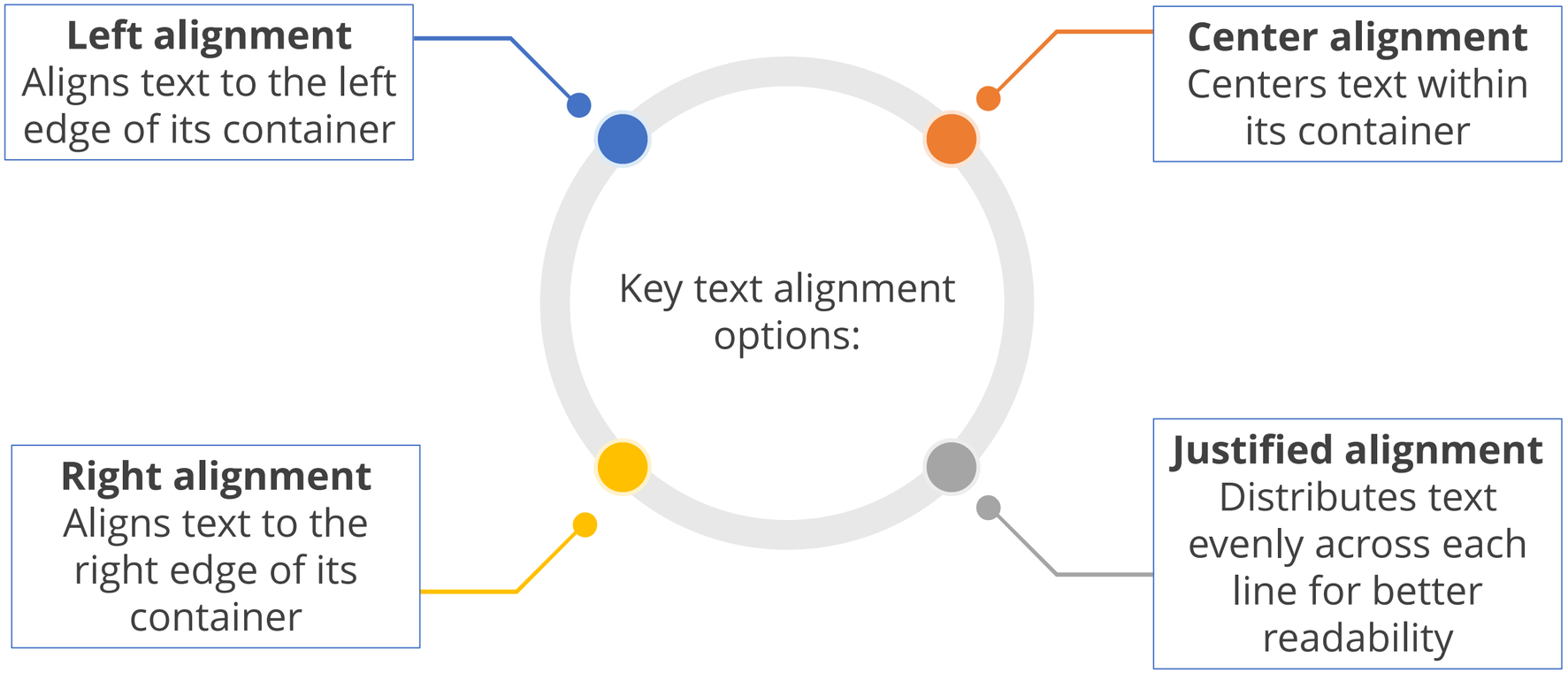
Colour-Shade System: List

Below is a list of important text color classes along with their syntax and explanation:

Class name	Syntax	Explanation
text-red-50	`text-[colorName]-[number: 50-950]`	Creates the lightest shade of a color, using a large amount of white. It may not be visible on a white background.
text-red-500	`text-[colorName]-[number: 50-950]`	Creates a balanced shade of a color, offering optimal contrast and visibility
text-red-950	`text-[colorName]-[number: 50-950]`	Creates the darkest shade of a color by adding a significant amount of black. It may not be visible on a dark background.
text-blue-500	`text-[colorName]-[number: 50-950]`	Creates a default blue shade, ensuring a well-balanced color tone
text-green-500	`text-[colorName]-[number: 50-950]`	Creates a default green shade, ensuring a well-balanced color tone
text-yellow-500	`text-[colorName]-[number: 50-950]`	Creates a default yellow shade, ensuring a well-balanced color tone

Text Alignment

It enables precise control over text positioning within its container, enhancing readability and ensuring a consistent design layout.



Text Alignment

Here are some essential text alignment classes:

Class Name	Explanation
`text-left`	Left align the text of an element
`text-right`	Right align the text of an element
`text-center`	Center align the text of an element
`text-justify`	Justify the text of an element

Assisted Practice

Demonstrating Typography Using Tailwind CSS **Duration: 10 Min.**

Problem statement:
Demonstrate text styling using Tailwind CSS to enhance readability, improve design consistency, and create visually appealing web pages with minimal effort.

Outcome:
By the end of this demo, you will be able to apply different text sizes using Tailwind's font sizing utilities, use text color utilities to modify text appearance, align text using Tailwind's alignment classes, and verify that the styles are applied correctly in your project.

Note: Refer to the demo document for detailed steps:
03_Demonstrating_Typography_Using_Tailwind_CSS

Assisted Practice: Guidelines

Steps to be followed:

1. Create an HTML file
2. Add typography-related code

Quick Check

A website’s navigation bar should be hidden on mobile but visible on larger screens. How should this be handled in Tailwind?

A. hidden md:block

B. block md:hidden

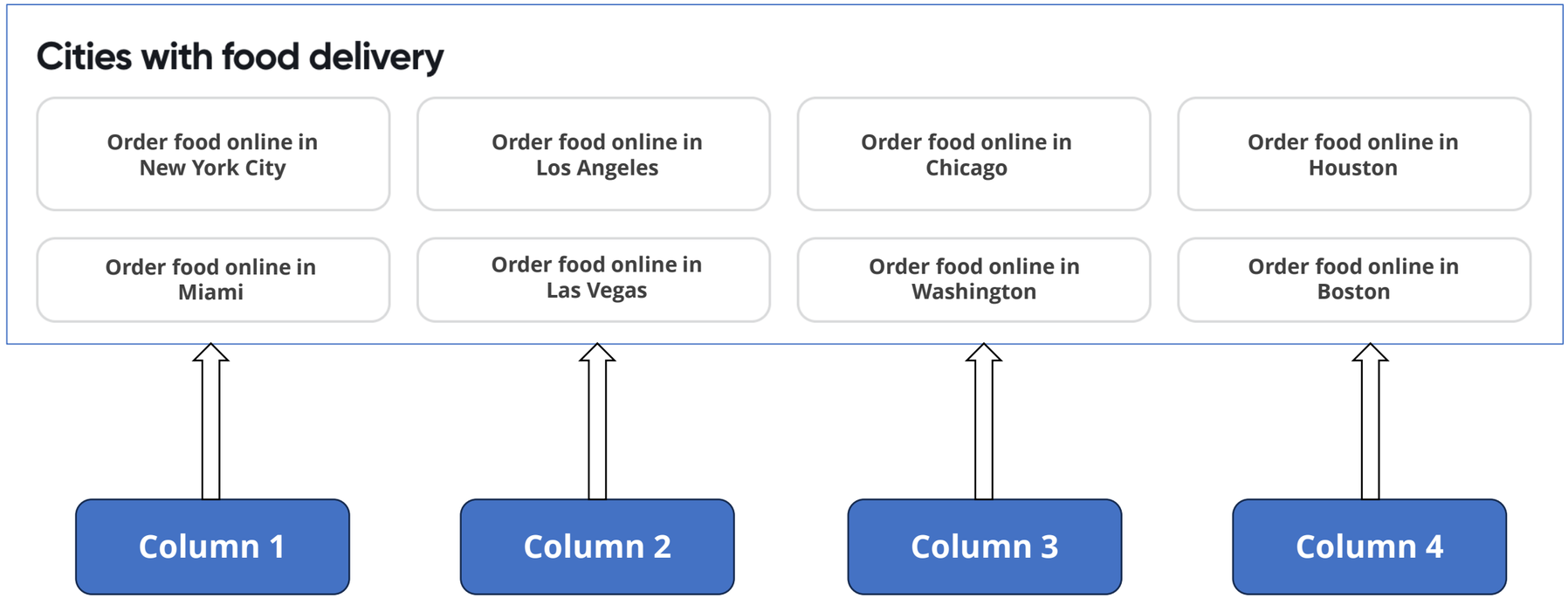
C. invisible md:visible

D. flex md:grid

Tailwind CSS Layout and Grid System

Tailwind Layout System

Tailwind allows developers to create responsive multi-column layouts using utility classes. It helps define column numbers and widths effortlessly for better content organization.



Tailwind’s column utilities help structure content into multiple columns for improved organization and readability.

Column Layout Options

Tailwind CSS supports column layouts from columns-1 to columns-12, along with columns-auto for dynamic sizing. Shown below are a few commonly used options:

1. **columns-1**: Creates a single-column layout for structured content display
2. **columns-auto**: Determines the optimal number of columns based on content
3. **columns-6**: Creates a layout with six evenly spaced columns for better content organization
4. **columns-12**: Supports up to 12 columns

Assisted Practice

Demonstrating Multi-Column Layouts in Tailwind CSS **Duration: 10 Min.**

Problem statement:
You have been assigned a task to demonstrate how to create a multi-column layout using Tailwind CSS. You need to define consecutive layouts on a new row with 3, 4, and 12 columns of different widths, respectively. The goal is to showcase how to use Tailwind's columns utilities to control column sizing.

Outcome:
By the end of this demo, you will be able to apply different column sizes using Tailwind's columns utilities and verify whether you can align multiple column numbers whenever necessary.

Note: Refer to the demo document for detailed steps:
04_Demonstrating_Multi-Column_Layouts_in_Tailwind_CSS

Assisted Practice: Guidelines

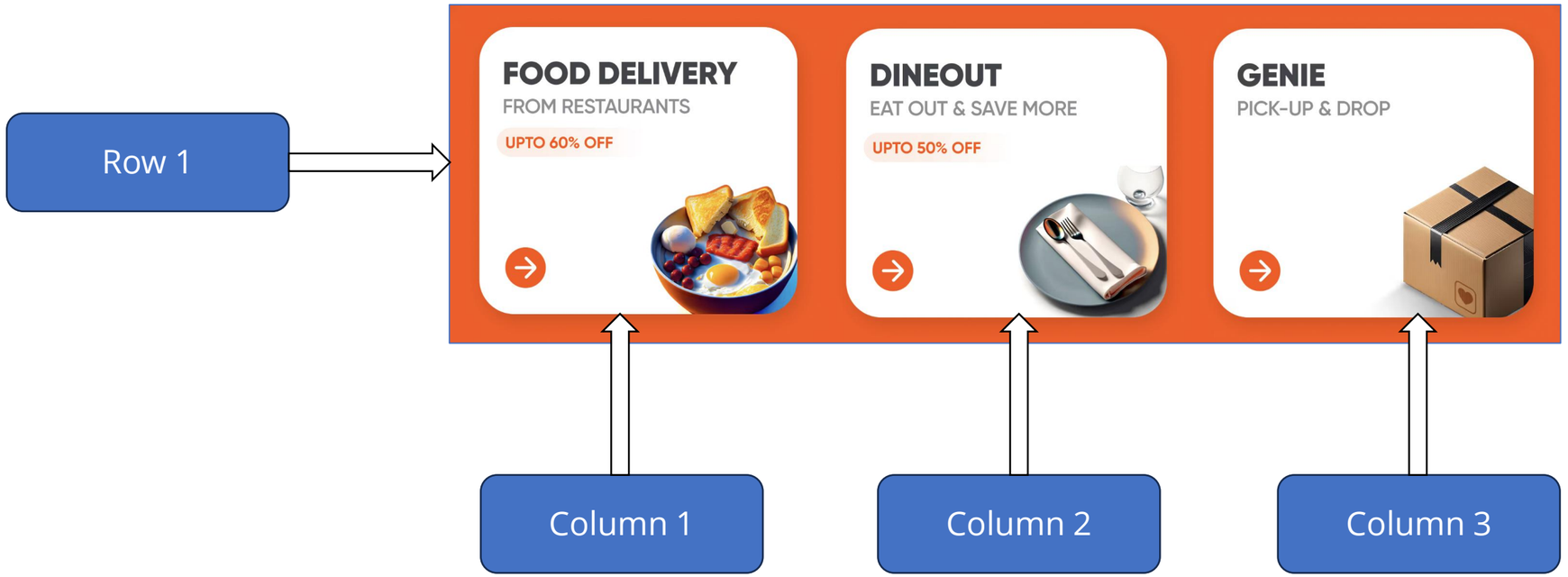
Steps to be followed:

1. Open Visual Studio Code and create index.html
2. Add layout-related code

ASSISTED PRACTICE

Tailwind Grid System

Tailwind enables the creation of complex, table-like layouts by defining rows, columns, and spacing using utility classes.



Note:

The image showcases the Tailwind Grid System with one row and three columns, representing food delivery, dining, and package pickup services.

Grid System Classes

Following are some important classes which are used to implement a grid column layout structure:

grid	Defines the element as a grid container, allowing child elements to align in rows and columns
grid-cols-<1-12>	Divides the container into ‘n’ number of equal-width columns as mentioned after grid-cols
gap-<0-96>	Controls the space between grid items (both row and column gaps). 0 means no gaps and 96 means highest spacing, 384 px

The layout and grid system in Tailwind helps define structural columns and rows, the flex system adds powerful control over item alignment and distribution within those layouts.

Assisted Practice

Demonstrating Grid-Columns Layout in Tailwind CSS **Duration: 10 Min.**

Problem statement:
You have been assigned a task to demonstrate how to create a responsive grid-based layout using Tailwind CSS. You need to showcase how to define the number of columns, control column width using col-span, and adjust the spacing with a gap. The goal is to create a structured and responsive layout using Tailwind's grid utilities.

Outcome:
By the end of this demo, you will be able to create a structured and responsive layout using Tailwind CSS Grid utilities that includes defining the number of columns, controlling column width using col-span and adjusting spacing with gap to achieve a flexible grid structure.

Note: Refer to the demo document for detailed steps:
05_Demonstrating_Grid-Columns_Layout_in_Tailwind_CSS

Assisted Practice: Guidelines

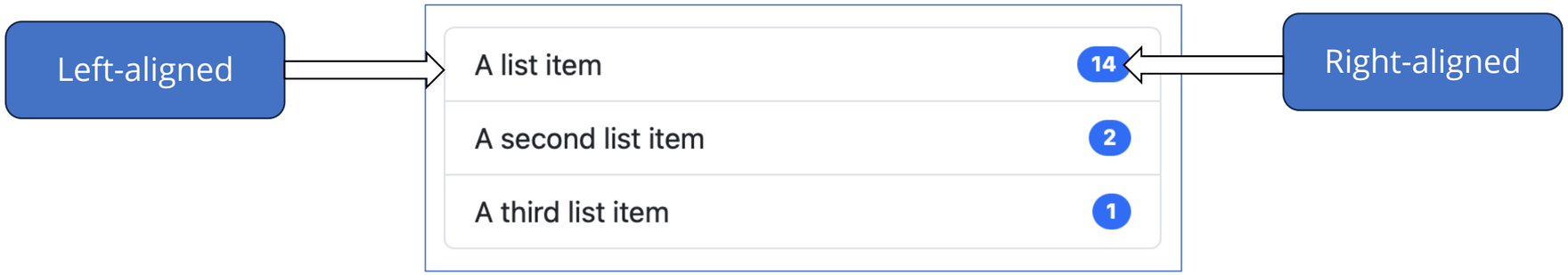
Steps to be followed:

1. Open Visual Studio Code and create index.html
2. Implement the grid layout

Understanding Tailwind Flex System

Tailwind Flex System: Introduction

It enables flexible layouts by managing element direction, alignment, and spacing with minimal code.

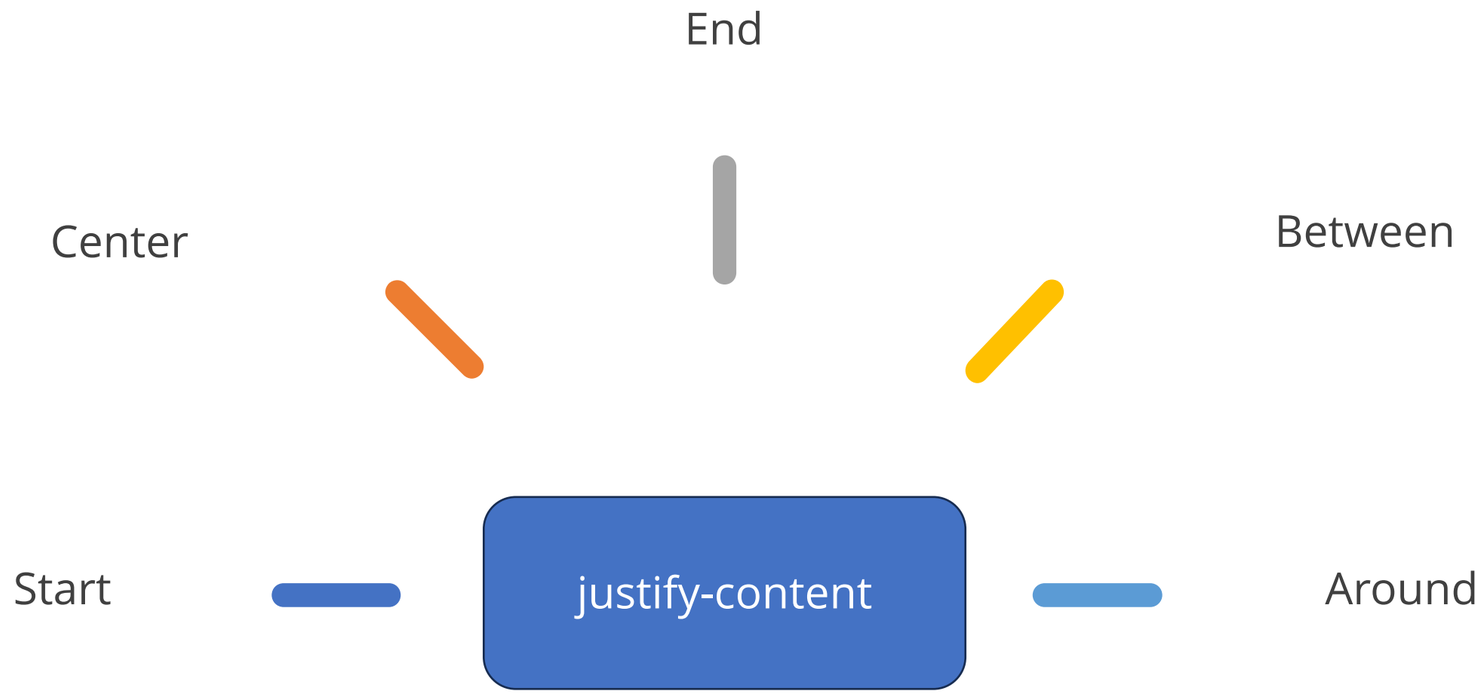


Features of the Tailwind Flex System:

- Aligns text and elements independently, both vertically and horizontally
- Uses flex utilities to control layout structure and positioning
- Achieves simultaneous multi-alignment for efficient UI design

Horizontal Flex Alignment

It refers to the process of positioning flex container items along the horizontal (X) axis using the **justify-content** property in a flexible layout.



This property controls how space is distributed between and around flex items, ensuring proper alignment based on predefined values such as start, center, end, between, and around.

justify-content

The justify-content property in the flexbox layout system defines how flex items are distributed along the main axis of a flex container.

justify-content-start

justify-content-end

justify-content-center

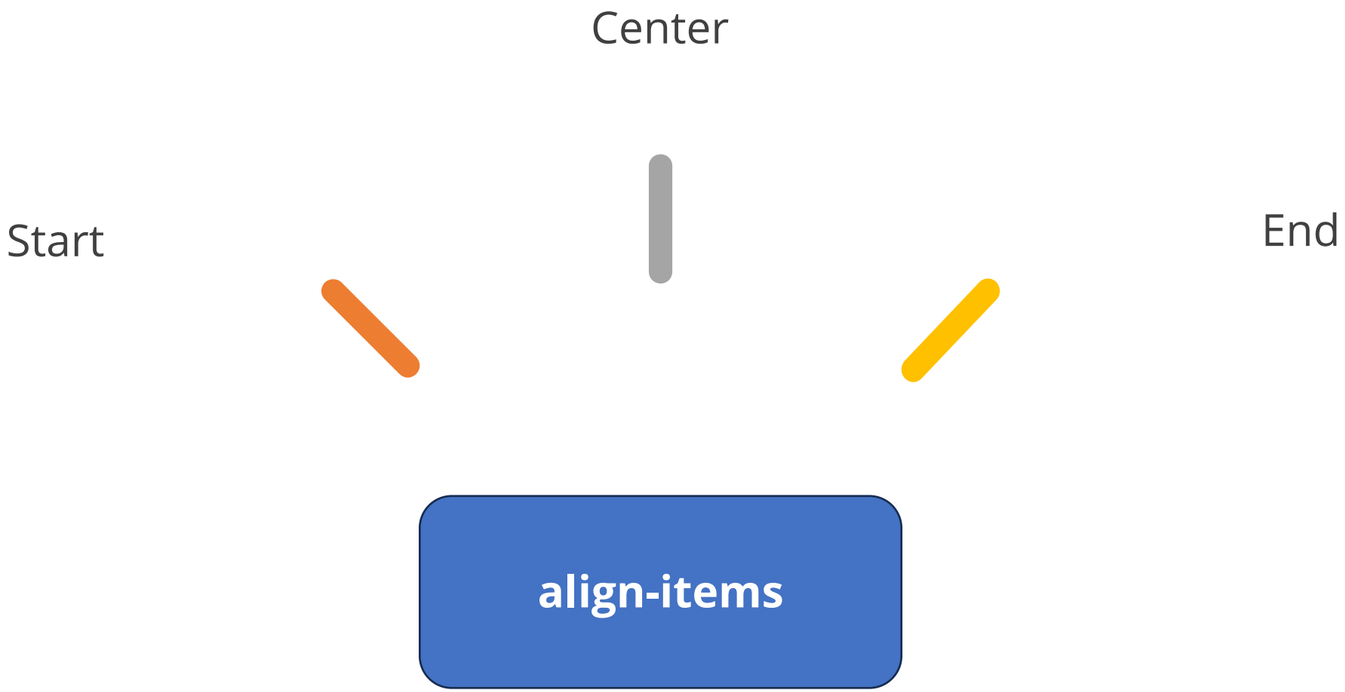
justify-content-end

justify-content-between

It determines the positioning of items by allocating space between or around them based on the selected alignment option.

Vertical Flex Alignment

It is the arrangement of flex items along the vertical axis in a flex container using **align-items** and align-self properties to control their distribution based on height.



Note:

The image illustrates different vertical alignment options **align-items** in a flex container, including start, center, and end positions.

align-items

The align-items property in CSS Flexbox controls the vertical alignment of flex items along the cross axis within a flex container.

align-items-start

align-items-center

align-items-end

Assisted Practice

Demonstrating Flex Behavior in Tailwind CSS **Duration: 10 Min.**

Problem statement:
You have been assigned a task to demonstrate how to create flexible layouts using Tailwind CSS. You need to showcase how the flex, justify-*, and items-* utilities work together to align and distribute content within a container. The goal is to help understand how to create dynamic and responsive layouts using Tailwind's flex utilities.

Outcome:
By the end of this demo, you will be able to create responsive flex containers using flex and flex-* utilities, align and justify items using justify-* and items-* classes, and control the direction and wrapping behavior of flex items effectively.

Note: Refer to the demo document for detailed steps:
06_Demonstrating_Flex_Behavior_in_Tailwind_CSS

Assisted Practice: Guidelines

Steps to be followed:

1. Open Visual Studio Code and create index.html
2. Implement the flex layout

Quick Check

A developer is creating a horizontal menu with three buttons and wants them to be evenly spaced, ensuring the first and last buttons align with the edges of the container. Which justify-content class in Tailwind CSS should the developer apply to achieve this layout?

A. justify-start

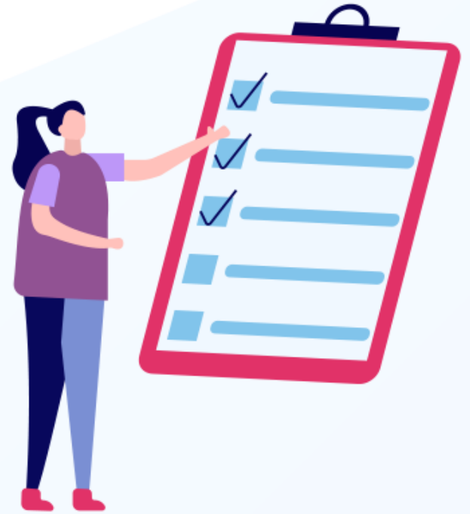
B. justify-center

C. justify-between

D. justify-end

Key Takeaways

- Tailwind is an open-source CSS framework that provides a utility-first approach to styling web applications.
- Tailwind CSS is built on core concepts that enhance flexibility and efficiency in web development.
- Colour-shade system represents a structured approach in defining colors with varying intensity levels, helping maintain consistency in design.
- Tailwind grid system enables the creation of complex, table-like layouts by defining rows, columns, and spacing using utility classes.
- The align-items property in CSS Flexbox controls the vertical alignment of flex items along the cross axis within a flex container



Thank You