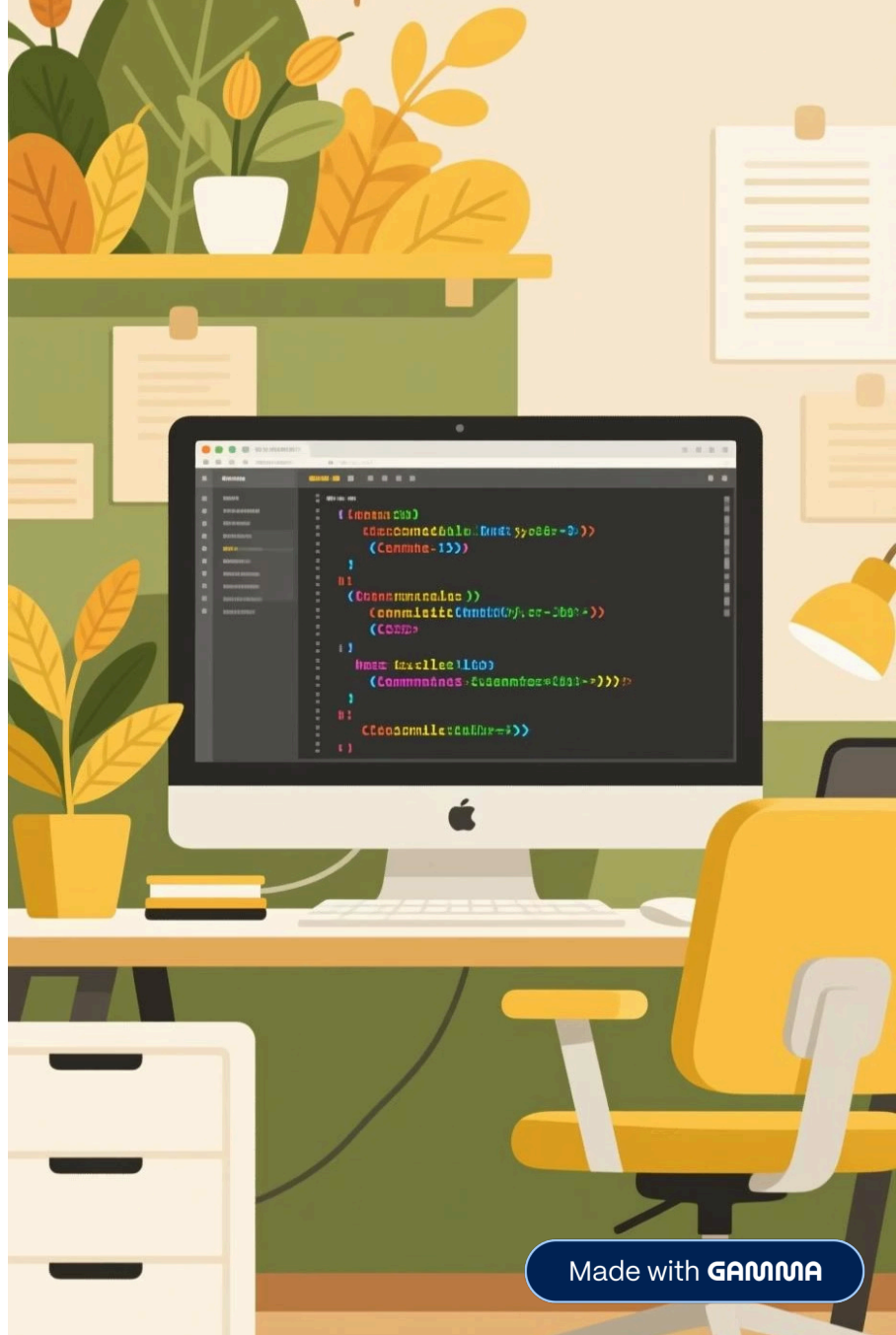
Tailwind CSS Fundamentals

A comprehensive guide for React + TypeScript developers



What is Tailwind CSS?

Tailwind CSS is a utility-first CSS framework built on top of standard CSS. It provides low-level utility classes that let you build completely custom designs directly in your markup without writing custom CSS.

Instead of creating separate stylesheets with semantic class names, you compose your designs using pre-built utility classes that map directly to CSS properties.

Why Choose Tailwind Over Plain CSS?

Rapid Development

Build interfaces faster by applying styles directly in your JSX using intuitive utility classes.

Design Consistency

Constrained design system ensures spacing, colours, and sizing remain consistent across your entire application.

Responsive by Default

Built-in responsive modifiers make it effortless to create mobile-first, adaptive layouts.

Traditional CSS Approach

```
<button class="my-big-red-button">

.my-big-red-button {
  background-color: red;
  padding: 16px;
  border-radius: 8px;
}
```

Tailwind CSS Approach

```
<button class="bg-red-500 p-4 rounded-lg">
```

Everything defined inline with utility classes – no separate CSS file needed!

Getting Started: Installation

01

Add TypeScript to React

If your project doesn't already have TypeScript, follow the official Create React App guide to add it.

https://create-react-app.dev/docs/adding-typescript/

02

Install Tailwind CSS

Follow the official Tailwind installation guide for Vite (the recommended build tool for modern React projects).

https://tailwindcss.com/docs/installation/using-vite

03

Configure Your Project

Set up your tailwind.config.js file and include Tailwind directives in your main CSS file to start using utility classes.

Spacing: Margins & Padding

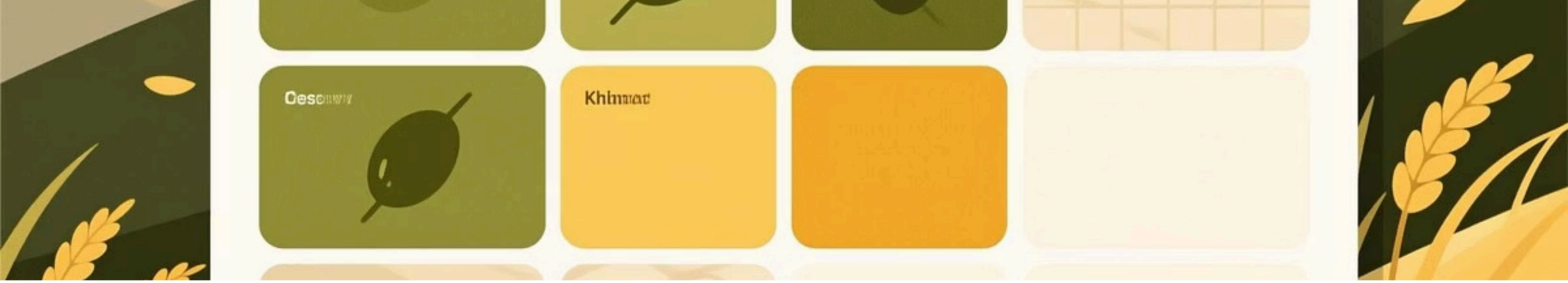
Control the space around and inside elements with intuitive utility classes. Each number represents a multiple of 0.25rem (4px).

Margin Utilities

- **m-4**: Margin on all sides
- **mt-8**: Margin top only
- **mb-2**: Margin bottom only
- **mx-auto**: Horizontal centring
- **my-5**: Vertical margins
- **-m-4**: Negative margin (overlap)

Padding Utilities

- **p-4**: Padding on all sides
- **pt-8**: Padding top only
- **px-8**: Horizontal padding
- **py-1**: Vertical padding
- **pr-6**: Padding right only
- **pl-5**: Padding left only



Sizing: Width & Height

Fixed Sizing

w-40, **h-24**: Fixed dimensions using the spacing scale. **w-px**, **h-px**: Exactly 1 pixel. **size-16**: Both width and height set to same value.

Custom Values

w-[150px], **h-[3rem]**: Specify any custom value not on the default scale using square brackets.

Relative Sizing

w-full, **h-full**: 100% of parent. **w-1/2**, **h-3/4**: Fractional sizing. **w-screen**, **h-screen**: 100% of viewport (100vw/vh).

Content-Based Sizing

w-auto, **h-auto**: Browser default. **w-fit**, **h-fit**: Based on content size. **w-min**, **h-min**: Minimum without overflow. **w-max**, **h-max**: Maximum without wrapping.

Typography & Colours

Font Families

- **font-sans**: Sans-serif typeface
- **font-serif**: Serif typeface
- **font-mono**: Monospace typeface

Font Sizes

- **text-xs**: Extra small text
- **text-sm**: Small text
- **text-base**: Default body size
- **text-lg**, **text-xl**: Large sizes
- **text-2xl** to **text-9xl**: Heading sizes

Font Weights & Styles

- **font-light**: Thin (300)
- **font-normal**: Regular (400)
- **font-bold**: Thick (700)
- **italic**, **not-italic**: Italics control

Text Alignment

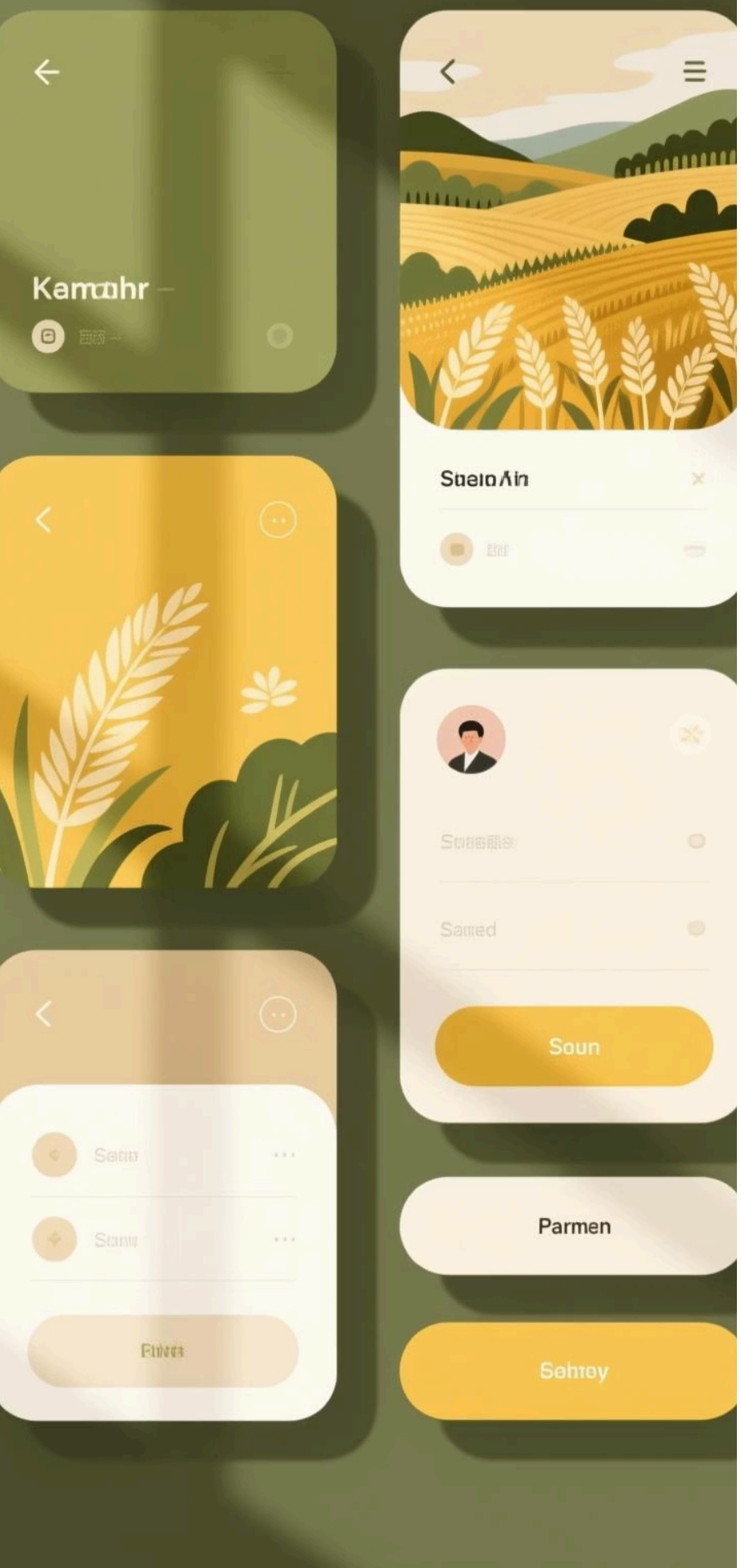
- **text-left**, **text-center**, **text-right**
- **text-justify**: Stretch to edges

Text Colours

text-gray-500, **text-blue-600**, **text-red-700**

Background Colours

bg-red-500, **bg-sky-100**, **bg-gray-800**, bg-white



Borders, Shadows & Layout

Border Utilities

- **border**: Default 1px border
- **border-2**: 2px thick border
- **border-t-4**: Top border only (4px)
- **border-x-8**: Left and right borders
- **border-blue-500**: Border colour

Border Radius

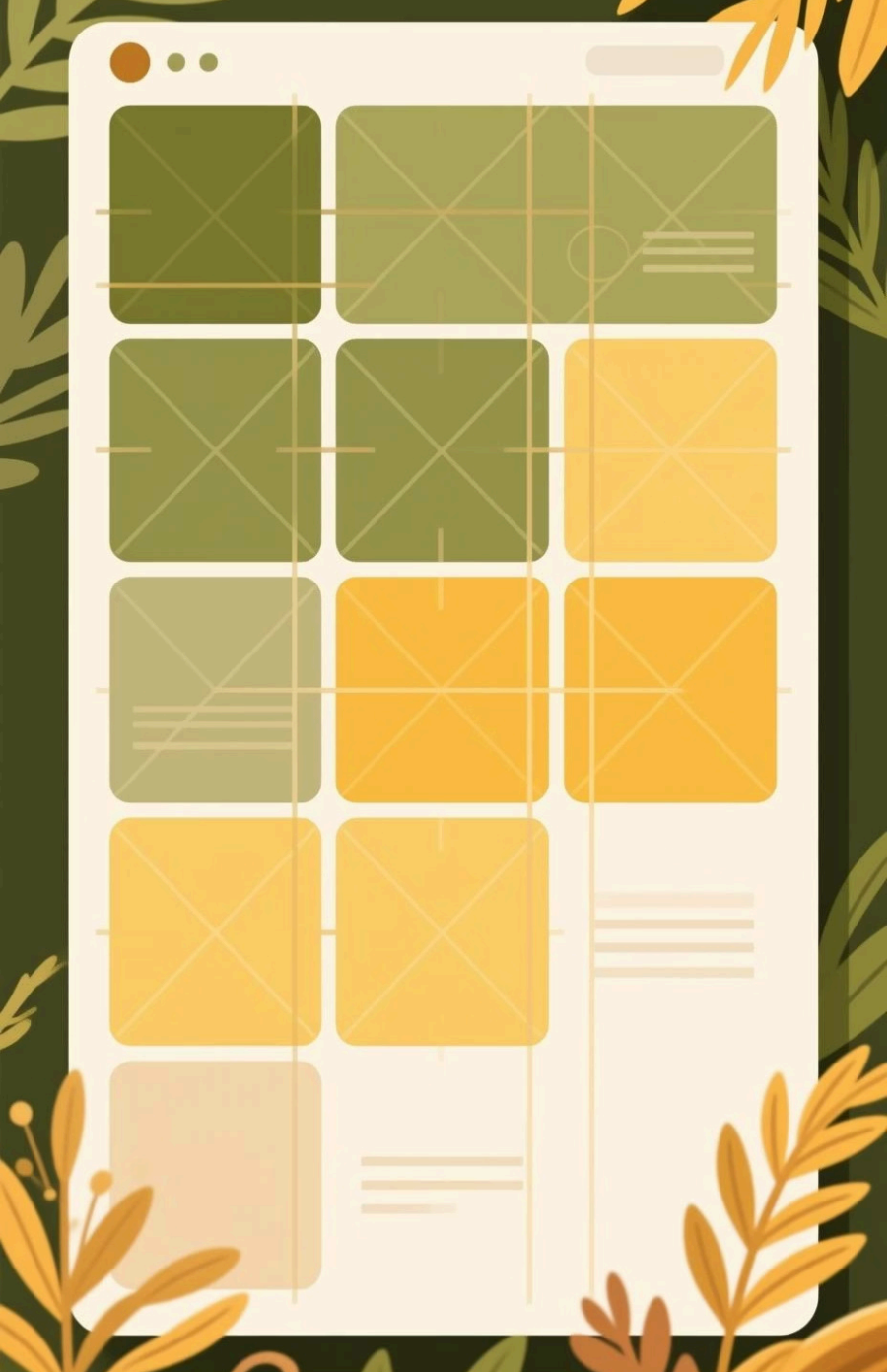
- **rounded-sm**: Small rounding
- **rounded-lg**: Large rounding
- **rounded-full**: Perfect circle

Display Types

- **block**: Full width, line break
- **inline**: Width of content only
- **inline-block**: Hybrid approach
- **hidden**: Completely invisible

Overflow Control

- **overflow-hidden**: Cut off content
- **overflow-scroll**: Always show scrollbars
- **overflow-auto**: Scrollbars when needed
- **overflow-x-scroll**: Horizontal only



Flexbox Layouts

Create flexible, responsive layouts with Tailwind's comprehensive flexbox utilities.

Enable Flex

flex: Sets display to flex container

Direction

flex-row: Horizontal layout

flex-col: Vertical layout

Main Axis

justify-start, **justify-center**, **justify-end**, **justify-between**

Cross Axis

items-start, **items-center**, **items-end**, items-stretch

Responsive Design & States

Mobile-First Approach

Define mobile styles first, then use breakpoint prefixes for larger screens: **sm:**, **md:**, **lg:**, **xl:**

Example: **lg:w-1/2 md:text-left**

Interactive States

Add state prefixes for interactive elements:

- **hover:bg-blue-700**: On mouse hover
- **focus:border-indigo-500**: On focus

Breakpoint Reference

- **sm**: Tablets (portrait)
- **md**: Tablets (landscape)
- **lg**: Laptops
- **xl**: Large desktops

You're now equipped with Tailwind CSS fundamentals! Start building beautiful, responsive interfaces with utility-first styling in your React + TypeScript projects.