



**Welcome to  
lecture 12!**

# Agenda

## Session Objectives

- The Problem with Traditional CSS Scaling
- What is Tailwind CSS?
  - Setting Up Tailwind
    - Demo
  - Core Utilities: Spacing, Sizing, Typography, Colors, Borders
    - Demo!
  - Basic Customization
    - Demo!
- Quiz
- Homework
- References

# Challenge: Why Does CSS Get Complicated?

- Writing CSS for small projects is straightforward. But as applications grow, we often encounter issues
  - Naming Things is Hard: Coming up with unique, descriptive, and consistent class names (.user-profile-card-header-title?) becomes a chore and can lead to inconsistencies
  - Specificity Conflicts ("Specificity Wars"): Styles unexpectedly override each other based on selector strength or CSS file order
    - This often leads to using !important as a quick fix, which makes maintenance harder
  - Unused CSS / Bloat: Large CSS files accumulate styles that might no longer be used, slowing down page load times. It's hard to confidently delete styles without breaking something
  - Consistency Across Teams/Features: Ensuring similar elements (like buttons or cards) look exactly the same across different parts of a large application requires strict conventions or duplication

# Tailwind CSS: A Utility-First Solution

- What is it?
  - A customizable, low-level CSS framework for building designs rapidly without writing much custom CSS
- Core Philosophy
  - Utility-First: Provides single-purpose utility classes (p-4, bg-blue-500, flex) that you combine directly in your HTML
  - Think of classes like direct CSS instructions (p-4 ≈ padding: 1rem;)
- How it differs from CSS libraries like Bootstrap or Material UI
  - Less Opinionated: Tailwind doesn't dictate what your components should look like; it gives you the tools to build your design. Bootstrap provides ready-made components with predefined styles you often need to override.
  - No Default Theme (Initially): You build the theme via your utility combinations. Customization is core, not an afterthought
  - HTML-Centric: Styling logic lives primarily in your HTML classes, not in separate CSS files defining component styles

# Example: Utility-First vs. Semantic CSS

HTML for a sample button with  
alert:

```
<div class="alert alert-warning alert-dismissible">  
  <button class="close-button">&times;</button>  
  <strong>Warning!</strong> Please review the details  
below.  
</div>
```

styles.css

```
/* styles.css */  
.alert { /* Base styles */  
  position: relative;  
  padding: 0.75rem 1.25rem;  
  margin-bottom: 1rem;  
  border: 1px solid transparent;  
  border-radius: 0.25rem;  
}  
.alert-warning { /* Variant styles */  
  color: #856404;  
  background-color: #fff3cd;  
  border-color: #ffeeba;  
}  
.alert-dismissible { padding-right: 3.5rem; } /* Space for button */  
.alert strong { font-weight: bold; } /* Child element style */  
.close-button {  
  position: absolute;  
  top: 0; right: 0;  
  padding: 0.75rem 1.25rem;  
  color: inherit; background: none; border: none; font-size: 1.5rem; /* Complex */  
}
```

# Example: Utility-First vs. Semantic CSS

Rebuilding the button using Tailwind

```
<div class="relative p-3 mb-4 border border-yellow-300 rounded bg-yellow-50 text-yellow-800">  
  <button class="absolute top-0 bottom-0 right-0 px-3 py-3 text-yellow-800 font-bold text-xl">&times;</button>  
  <strong class="font-bold">Warning!</strong> Please review the details below.  
</div>
```

- Takeaways

- Traditional: Requires defining multiple classes (alert, alert-warning, alert-dismissible, close-button) and managing their relationships and potential overrides in CSS. Requires context switching between HTML and CSS
- Tailwind
  - Styling is immediately visible in the HTML. No need to switch files or invent class names like alert-warning
  - Combining utilities (p-3, mb-4, border, bg-yellow-50, text-yellow-800) directly creates the style instead of using a separate stylesheet

A top-down view of a wooden desk. In the center is a large white rectangular paper. In the top-left corner of the paper is a small wooden bowl filled with various colored pencils and pens. In the top-right corner of the paper is a white mug filled with a frothy beverage. The background is the natural wood grain of the desk.

# Tailwind Setup

# Setting Up Tailwind CSS

- How Can We Add Tailwind to Our Project?
  - Content Delivery Network (CDN)
    - Add Tailwind using a single <script> tag in your HTML <head>
    - No installation or build step needed – just write HTML!
    - Uses a CDN to quickly load the necessary styles in the browser
    - Benefit: Instant setup, great for trying things out and today's lecture
    - Limitation: Less optimized (loads all styles initially) and fewer deep customization options
  - Build Process (Standard for Projects)
    - Uses npm (Node Package Manager) to install Tailwind
    - Requires configuration (tailwind.config.js) for customization and purging (removing unused styles for small production files)
    - Benefit: Optimized, fully customizable, production-ready
    - Note: We'll explore this method later in the course when we cover Node.js



# Example: Tailwind CSS in code

Notice the script tag:

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Tailwind Page</title>
  <script src="https://cdn.tailwindcss.com"></script>
</head>
<body>
</body>
</html>
```



**Demo Time!**



# Tailwind Utilities

# Exploring Core Tailwind Utilities

- We'll cover the most common utilities you'll use to style your elements
  - Spacing (Padding & Margin)
  - Sizing (Width & Height)
  - Typography (Text styling)
  - Colors (Background, Text, Border)
  - Borders (Width, Style, Radius)
  - Images

# Core Concept: Anatomy of a Utility Class

- Tailwind's utility classes generally follow predictable patterns, making them easier to learn and remember
- **Common Pattern:** {property\_abbreviation}-{value\_or\_modifier}
  - p-4 -> padding: 4 (maps to 1rem by default)
  - m-2 -> margin: 2 (maps to 0.5rem by default)
- **Pattern with Modifiers (Colors, Breakpoints):** {modifier}:{property}-{value} OR {property}-{color\_name}-{shade}
  - text-blue-500 -> text (color property): blue (color name) - 500 (shade)
  - bg-gray-100 -> background: gray (color name) - 100 (shade)
  - font-bold -> font (font-weight property): bold (value)
  - md:text-lg -> md (medium breakpoint modifier): text (font-size property) - lg (value)
- **Benefit:** the class name itself tells you the CSS property and value it applies

# Spacing: Padding & Margin

- Tailwind provides intuitive utilities for controlling space inside (padding) and outside (margin) elements, directly mapping to the Box Model concepts
- **Padding (p-):** Space inside the border
  - p-{size}: All sides (e.g., p-4 = padding: 1rem;)
  - px-{size}: Horizontal (left & right). E.g., px-6 = padding-left: 1.5rem; padding-right: 1.5rem;
  - py-{size}: Vertical (top & bottom). E.g., py-2 = padding-top: 0.5rem; padding-bottom: 0.5rem;
  - pt-{size}, pr-{size}, pb-{size}, pl-{size}: Directional (top, right, bottom, left). E.g., pb-8 = padding-bottom: 2rem;
- **Margin (m-):** Space outside the border
  - m-{size}: All sides. E.g., m-2 = margin: 0.5rem;
  - mx-{size}, my-{size}: Horizontal/Vertical. E.g: mx-auto = margin-left: auto; margin-right: auto; -> centers block elements with a defined width
  - mt-{size}, mr-{size}, mb-{size}, ml-{size}: Directional (e.g., mt-1 = margin-top: 0.25rem;)
  - Negative Margins: Prepend a hyphen (e.g., -m-4, -mt-2) for overlapping elements or specific layout adjustments

# The Spacing Scale

- Tailwind uses a default numeric scale
  - Starting at 0, 0.5, 1, 1.5, 2, 2.5, 3... up to 96
- By default, 1 unit = 0.25rem; so, 1 unit = 4px if base font size for the webpage is 16px
  - So, p-4 means  $4 * 0.25\text{rem} = 1\text{rem}$
- Note: always refer to the official [Tailwind documentation](#) for the exact scale and values
- Example

```
<div class="py-4 mx-8 bg-slate-100">Content with space</div>
<div class="w-64 mx-auto bg-lime-100">Centered Box</div>
<p class="mt-6">Space above this paragraph.</p>
```

# Sizing: Width & Height

- Control the dimensions of your elements using Tailwind's sizing utilities
- Width (w-):
  - Fixed: w-{size} using the spacing scale (e.g., w-16 = 4rem, w-64 = 16rem).
  - Fractions: w-1/2 (50%), w-1/3, w-2/3, w-1/4, w-3/4, w-1/5... w-11/12, w-full (100%). Percentage-based, relative to the parent container.
  - Viewport: w-screen (100% of viewport width).
  - Content-Based: w-auto (browser default), w-min (min-content), w-max (max-content).
- Height (h-)
  - Fixed: h-{size} using the spacing scale (e.g., h-10 = 2.5rem, h-32 = 8rem).
  - Fractions: h-1/2, h-full (100%) - Often requires the parent element to have a defined height.
  - Viewport: h-screen (100% of viewport height).
  - Content-Based: h-auto.



# Width & Height (Contd.)

- Max/Min Sizing: Useful for setting boundaries
  - max-w-{size}: Constrains maximum width (e.g., max-w-md, max-w-7xl, max-w-full). Very common for page containers.
  - min-w-{size}: Ensures minimum width.
  - max-h-{size}: Constraints maximum height (e.g., max-h-48).
  - min-h-{size}: Ensures minimum height (e.g., min-h-screen for full viewport height sections).
- Example

```
<div class="max-w-4xl mx-auto p-4 bg-white">Page Content</div>
<aside class="w-48 min-h-screen bg-gray-200"></aside>
<div class="flex"> <div class="w-1/2 bg-red-100 p-4">Column 1</div>
  <div class="w-1/2 bg-blue-100 p-4">Column 2</div>
</div>
```



**Demo Time!**

# Typography: Styling Your Text

- Tailwind offers comprehensive utilities for controlling text appearance
- **Font Size: text-**
  - Examples: text-xs, text-sm, text-base (default, usually 16px), text-lg, text-xl... text-9xl
- **Font Weight: font-**
  - Examples: font-light (300), font-normal (400), font-medium (500), font-semibold (600), font-bold (700), font-extrabold (800)
- **Text Color: text-**
  - Uses the standard color palette (e.g., text-gray-800, text-blue-600, text-white)
  - We'll cover more in colors slide
- **Text Alignment: text-**
  - Values: text-left, text-center, text-right, text-justify
- **Line Height: leading- (controls line-height for readability)**
  - Examples: leading-none (1), leading-tight (1.25), leading-snug (1.375), leading-normal (1.5), leading-relaxed (1.625), leading-loose (2)

# Typography: Styling Your Text (Contd.)

- Letter Spacing: tracking-
  - Examples: tracking-tighter, tracking-tight, tracking-normal, tracking-wide, tracking-wider
- Text Decoration/Style: underline, line-through, no-underline, italic, not-italic, uppercase, lowercase, capitalize
- Example

```
<h1 class="text-3xl font-bold text-center text-gray-900 mb-4">Document Title</h1>
<p class="text-base font-normal text-gray-700 leading-relaxed tracking-wide">
  This is a standard paragraph with relaxed line height and wider letter spacing for readability. Use <span class="font-semibold italic text-indigo-600">Tailwind</span> utilities.
</p>
<a href="#" class="text-blue-500 underline hover:text-blue-700">Read More</a>
```

# Colors: Backgrounds, Text, and Borders

- Tailwind includes an extensive, curated default color palette, making consistent color usage easy
- Color Palette
  - Includes colors like gray, red, yellow, green, blue, indigo, purple, pink, and more
- Shades
  - Each color typically comes in shades from 50 (lightest) to 950 (darkest), e.g., blue-50, blue-100, ..., blue-900
- Applying Colors
  - Background Color (bg-): Sets the background-color. Examples: bg-gray-100, bg-teal-500, bg-black.
  - Text Color (text-): Sets the color. Examples: text-red-700, text-gray-500, text-white.
  - Border Color (border-): Sets border-color. Needs a border width utility (border, border-2, etc.) to be visible. Examples: border-gray-300, border-green-600.
  - Placeholder Color (placeholder-): Styles placeholder text in inputs. Example: placeholder-gray-400

# Typography: Styling Your Text (Contd.)

- Opacity Utilities: Add transparency. Combine with color utilities
  - Examples: bg-opacity-75, text-opacity-50.
  - Usage: <div class="bg-blue-500 bg-opacity-50">...</div>
- Finding Colors: The official Tailwind documentation has an excellent interactive color palette reference – essential for choosing shades
- Example

```
<button class="bg-purple-600 hover:bg-purple-700 text-white font-bold py-2 px-4 rounded">
```

```
  Submit
```

```
</button>
```

```
<div class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded relative mt-4" role="alert">
```

```
  <strong class="font-bold">Error!</strong>
```

```
  <span class="block sm:inline">Something went wrong.</span>
```

```
</div>
```



**Demo Time!**

# Borders: Width, Style, Color & Radius

- Tailwind makes it easy to add and style borders on elements
- **Border Width (border, border-{size})**
  - border: Applies a 1px solid border to all sides. This is often needed first for border colors to show.
  - border-0, border-2, border-4, border-8: Specific pixel widths (2px, 4px, 8px by default).
  - Directional Widths: border-t-{size}, border-r-{size}, border-b-{size}, border-l-{size} (e.g., border-b-2 for a 2px bottom border).
- **Border Color (border-{color}-{shade})**
  - Uses the standard color palette (e.g., border-gray-300, border-indigo-500).
  - Remember to set a border width (border, border-2, etc.) for the color to be visible
- **Border Style (border-{style})**
  - border-solid (default, usually implied by border).
  - Alternatives: border-dashed, border-dotted, border-double, border-none



# Borders (Contd.)

- **Border Radius (rounded, rounded-{size}): Controls corner rounding**
  - Scale: rounded-sm, rounded (4px default), rounded-md (6px), rounded-lg (8px), rounded-xl, rounded-2xl, rounded-3xl.
  - Full Rounding: rounded-full (for circles/pills).
  - Directional/Corner Rounding: rounded-t-lg (top corners), rounded-r-md (right corners), rounded-bl-xl (bottom-left), etc.
- **Example**

```
<input type="text" class="border border-gray-300 rounded-md p-2">
```

```
<div class="border-b-4 border-dashed border-red-500 py-4 my-4">Dashed Bottom Border</div>
```

```

```

```
<div class="bg-white shadow-md rounded-t-xl p-4">Card Top</div>
```

# Styling Images with Utilities

- Apply sizing, borders, and object-fit utilities to style images effectively
- Sizing
  - Use w-{size} and h-{size} (e.g., w-32, h-32, w-full, h-auto). Remember h-full needs a parent with defined height
- Object Fit (object-{fit}): Controls how the image content resizes within its element dimensions if aspect ratios differ.
  - object-cover: Image covers the entire area, cropping if necessary. Keeps aspect ratio. (Very common for cards/avatars).
  - object-contain: Image fits within the area without cropping, potentially leaving empty space (letterboxing). Keeps aspect ratio.
  - object-fill: Stretches/squashes the image to fill the area, ignoring aspect ratio.
  - object-none: Displays image at original size, cropping if needed.
  - object-scale-down: Behaves like contain if image is larger than container, like none if smaller.

# Styling Images with Utilities (Contd.)

- Aspect Ratio (aspect-{ratio}): Often applied to a parent div to maintain intrinsic ratio, especially for responsive videos/images.
  - Examples: aspect-square (1:1), aspect-video (16:9), aspect-[4/3] (arbitrary). The image inside can then use w-full h-full object-cover
- Example

```

```

```
<div class="w-full aspect-video bg-gray-300">   
</div>
```

```

```



**Demo Time!**



# Customization in Tailwind

# Basic Customization: Arbitrary Values

- The Need
  - What if the default Tailwind theme (colors, spacing scale) doesn't have the exact value you need for a specific element, and you don't want to write separate CSS?
- The Solution
  - Tailwind (even via CDN) allows "arbitrary values" using square bracket [] notation directly within the class name
- **Syntax:** {utility}-{arbitrary\_value}
  - The value inside the brackets is interpreted literally by Tailwind's Just-in-Time (JIT) engine (included in the Play CDN)
- Examples
  - Colors: Need a very specific hex color?
    - Twitter blue: `bg-[#1DA1F2]`
    - Tomato color using RGB: `text-[rgb(255,99,71)]`

# Examples: Basic Customization

- Examples

- Colors: Need a very specific hex color?
  - Twitter blue: `bg-[#1DA1F2]`
  - Tomato color using RGB: `text-[rgb(255,99,71)]`
- Spacing: Need a pixel-perfect margin or padding?
  - Add 11 pixels to the margin top: `mt-[11px]`
  - Padding as a percentage: `p-[3%]`
- Sizing: Need a specific width or height?
  - Fixed width: `w-[550px]`
  - Height based on viewport minus fixed header, requires calc: `h-[calc(100vh-80px)]`
- Set Font size to 14 pixels: `text-[14px]`
- Grid/Flex Basis: `basis-[300px]`

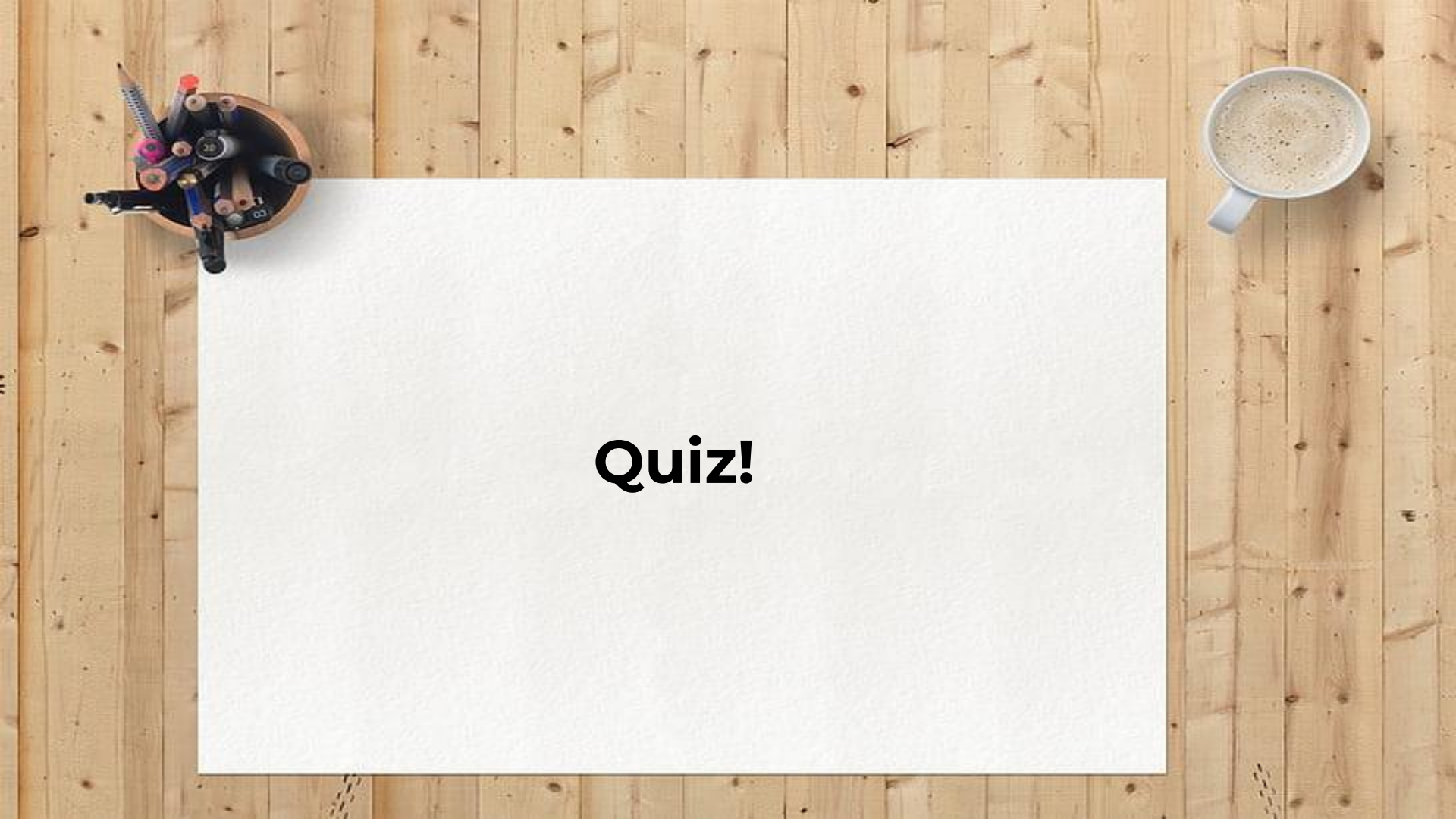
# Beyond Basics: Customization via `tailwind.config.js`

- While arbitrary values handle one-offs, Tailwind's true power lies in its deep customization via the `tailwind.config.js` file (used with the Build Process/NPM method).
- Purpose of `tailwind.config.js`
  - Theming: Define or extend your project's specific design tokens – colors, spacing scale, font sizes, breakpoints, border radii, etc.
  - Adding Custom Fonts: Integrate web fonts easily.
  - Extending Variants: Control which utilities get responsive (md:), state (hover:, focus:) or other variants generated.
- Why Mention It Now?
  - While we use the CDN today for simplicity, real-world projects typically use the build process
  - Once we learn NPM in Lecture 23, you'll have the foundation to create your custom Tailwind config!





**Demo Time!**

A top-down view of a wooden desk. In the center is a large white rectangular paper. In the top-left corner of the paper is a small wooden bowl filled with various colored pencils and pens. In the top-right corner of the paper is a white mug filled with a frothy beverage. The word "Quiz!" is written in the center of the white paper.

**Quiz!**

# Question 1

- What is the core philosophy behind Tailwind CSS?
  - A) Component-Based Styling
  - B) Semantic Class Naming
  - C) Utility-First Styling
  - D) Object-Oriented CSS

Correct Answer: C

## Question 2

- Which method did we use today to add Tailwind CSS to our project for quick setup?
  - A) Installing via NPM
  - B) Using the Play CDN `<script>` tag
  - C) Importing a CSS file
  - D) Using a `<link>` tag

Correct Answer: B

## Question 3

- Which Tailwind utility class would add padding to only the left and right sides of an element?
  - A) p-4
  - B) py-4
  - C) pl-4
  - D) px-4

Correct Answer: D

## Question 4

- How would you set the background color of a div to a medium-dark blue (shade 600) in Tailwind?
  - A) color: blue-600;
  - B) text-blue-600
  - C) bg-blue-600
  - D) background: blue-600;

Correct Answer: C

## Question 5

- If you want to make text bold and large, which combination of classes might you use?
  - A) font-semibold text-sm
  - B) font-bold text-2xl
  - C) text-bold size-large
  - D) fw-bold fs-lg

Correct Answer: B

# Question 6

- What class is typically needed before a border-red-500 class will have a visible effect?
  - A) border-style-solid
  - B) border (or border-2, border-4, etc.)
  - C) outline-red-500
  - D) color-red-500

Correct Answer: B



# Question 7

- How can you apply a specific, non-standard margin-top value like 13px to an element?
  - A) `mt-[13px]`
  - B) `margin-top: 13px;`
  - C) `mt-custom-13`
  - D) `m-t-[13]`

Correct Answer: A

# Question 8

- Which prefix is used to apply a style only when the mouse cursor is over an element?
  - A) active:
  - B) focus:
  - C) visited:
  - D) hover:

Correct Answer: D

# Homework: Portfolio Project Showcase

- Task: Create a section showcasing two or more portfolio projects using Tailwind
- Sections
  - Project Container: A div acting as a card (w/ background, padding, rounding, shadow/border). Try arrange your cards in a clean manner (in a row or column)
  - Each card would contain:
    - Project Image
    - Project Title
    - Short Description
    - Tech Stack Tags (eg: HTML, CSS)
    - Link Button

# References

- [Official Tailwind CSS Documentation](#)
  - Best place to start!
- [Tailwind Play](#)
  - The online playground using the CDN.
  - Great for experimenting with classes and sharing snippets without local setup
- [Tailwind Labs](#) Youtube Channel
  - Videos from the creators covering concepts, new features, and build tutorials