

A decorative border at the top of the slide features a variety of white flowers and green foliage. On the left, there are clusters of small, delicate white flowers. In the center, a single white rose is visible. To the right, there are more clusters of small white flowers and a single white rose. The background is a light-colored wooden surface with a visible grain.

Welcome to lecture 13!

Agenda

Session Objectives

- Recap
 - CSS Flexbox
 - Tailwind Basics
- The "Why"
 - Complex Responsive Layouts
- Tailwind
 - Flexbox Utilities & Live Coding
 - Grid Utilities & Live Coding
 - Fine-Tuning Layouts with Arbitrary Values
- Break & Exercise
- Responsive Design with Tailwind & Live Coding
- Quiz Time!

Recap: CSS Flexbox Fundamentals

- What problem did CSS Flexbox solve?
 - Provided a reliable one-dimensional way to align and distribute space among items without floats or tables
- Key Concepts Recap
 - Container (display: flex) & Items.
 - Main Axis & Cross Axis (depend on flex-direction).
 - justify-content: Alignment on Main Axis.
 - align-items: Alignment on Cross Axis.
 - flex-wrap: Handling overflow

Recap: Tailwind CSS Basics

- What's the benefit of Tailwind?
 - Utility-First: Style directly in HTML (p-4, bg-blue-500, font-bold)
- Core Utilities:
 - Spacing (m-, p-), Sizing (w-, h-), Colors (bg-, text-), Typography (text-, font-), Borders (border, rounded-)
- Setup
 - Using `<script src="https://cdn.tailwindcss.com"></script>`.
- Arbitrary Values: What are they & how do we use them?
 - What: Custom values you specify when Tailwind doesn't support it
 - How: [...] for one-offs (e.g., mt-[11px])
- Example

```
<div class="p-4 bg-indigo-600 text-white rounded-md  
font-semibold">Styled Box</div>
```

Why: Complex Responsive Layouts

- As we've seen, modern interfaces demand structure and adaptability across devices
- How do sites like these handle layout efficiently?

Stripe

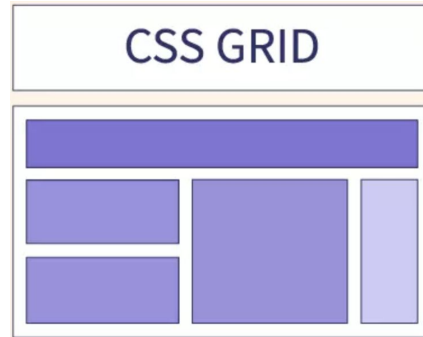
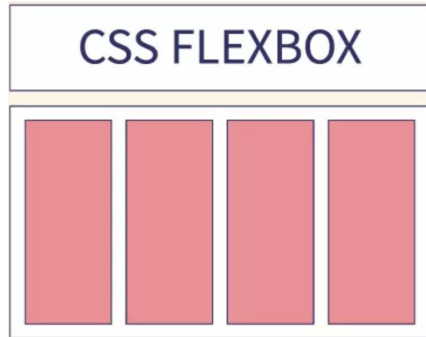
The screenshot shows the Stripe API documentation for the 'Pagination' section. The left sidebar contains a navigation menu with links like 'Expanding Responses', 'Idempotent requests', 'Metadata', 'Pagination' (highlighted), 'Search', 'Auto-pagination', 'Request IDs', 'Versioning', 'Core Resources' (with a dropdown arrow), 'Balance', 'Balance Transactions', 'Charges', 'Customers', 'Customer Session', 'Disputes', 'Events', 'Event Destinations', 'Files', 'File Links', 'FX Quotes', and 'Mandates'. The main content area is titled 'Pagination' and explains that all top-level API resources support bulk fetches through 'list' methods. It details the parameters: 'limit' (optional, default 10), 'starting_after' (optional object ID), and 'ending_before'. A 'RESPONSE' section shows a JSON snippet for a customer list. The right sidebar contains links for '2025-04-30.basil', 'API Reference', 'Docs', 'Support', and 'Sign in'.

Linear

The screenshot shows the Linear product page. At the top, there's a navigation bar with links for 'Product', 'Resources', 'Pricing', 'Customers', 'Blog', 'Contact', 'Log in', and 'Sign Up'. Below the navigation bar is a grid of icons representing various features. The main heading is 'The new standard for modern product development'. Below this, there's a subheading 'Set the product direction with projects and initiatives' and a line graph showing a trend. Further down, there's a section titled 'Make progress with issue tracking and cycle planning' with a line graph. The bottom section is titled 'Instant analytics for any stream of work' and shows a mobile app interface. The footer contains links for 'Customer Requests', 'Build what customers actually want', and 'Turn workplace requests into actionable issues'.

The "How": Tailwind's Layout Powerhouses

- Tailwind provides intuitive utilities for CSS Flexbox and Grid.
- Flexbox (flex ...):
 - Best for 1D layouts (rows OR columns).
- Use Cases: Navbars, button groups, aligning items inside components.
- Grid (grid ...):
 - Best for 2D layouts (rows AND columns).
 - Use Cases: Page structure, dashboards, galleries, complex forms.
- They often work together!





Flexbox with Tailwind Utilities

Enabling Flexbox & Direction

- **Apply to the parent container:**
 - flex: Block-level flex container.
 - inline-flex: Inline-level flex container.
- **Set item flow direction:**
 - flex-row (Default - Left to Right)
 - flex-row-reverse (Right to Left)
 - flex-col (Top to Bottom)
 - flex-col-reverse (Bottom to Top)

Aligning Items: Cross Axis

- `item-*` aligns children along the cross axis
- On the container:
 - `items-start`: Align to start.
 - `items-center`: Align to center (Very Common!).
 - `items-end`: Align to end.
 - `items-stretch` (Default): Fill cross axis dimension.
 - `items-baseline`: Align text baselines.
- Override individual items with `self-*` (e.g., `self-center`).
- Quick Check: Where will items inside the div align?
 - `<div class="flex h-24 items-end"> ... </div>`

Justifying Content: Main Axis

- **justify-*** aligns children along the main axis
- **On the container:**
 - justify-start (Default): Items at start.
 - justify-center: Items in center.
 - justify-end: Items at end.
 - justify-between: Space between items only (Common!).
 - justify-around: Space around each item.
 - justify-evenly: Space evenly distributed.
- **Override individual items with self-* (e.g., self-center).**
- **Quick Check: How will Left/Right be positioned?**
 - `<div class="flex justify-between">`
 `Left`
 `Right`
 `</div>`

Handling Overflow & Spacing

- How did we handle overflow and spacing in flexbox?
 - flex-wrap for overflow; gap for spacing
- Control wrapping when items don't fit:
 - flex-nowrap (Default)
 - flex-wrap (Wrap to next line - Essential for RWD!)
 - flex-wrap-reverse
- Add space between items (preferred over margins):
 - gap-{size} (e.g., gap-4 -> 1rem row and column gap)
 - gap-x-{size} (Horizontal gap only)
 - gap-y-{size} (Vertical gap only)

Sizing Flex Items

- **Control how items resize**
 - flex-1: Grow & shrink equally. Most common way to fill space. CSS reference - flex: 1 1 0%
 - flex-auto: Grow & shrink based on content size. CSS reference - flex: 1 1 auto
 - flex-initial: Shrink if needed, don't grow. CSS reference - flex: 0 1 auto
 - flex-none: Fixed size, no grow/shrink. CSS reference: flex: none
- **Use grow, shrink, basis-{size} for fine-grained control**
- **Quick Check: How will the Search Bar size itself?**
 - ```
<div class="flex">
 <div class="w-20">Logo</div>
 <div class="flex-1">Search Bar</div>
</div>
```
  - Answer: It will grow to fill remaining space



**Demo Time!**

# Flexbox: Use Cases & Summary

- **Strength**
  - Aligning items in a single dimension (rows OR columns)
  - Distributing space along that dimension
  - Great for component-level layouts (navbars, cards, forms, button groups)
- **Tailwind makes these common patterns fast to implement**
- **Quick Check: How will the Search Bar size itself?**
  - ```
<div class="flex">  
  <div class="w-20">Logo</div>  
  <div class="flex-1">Search Bar</div>  
</div>
```
 - Answer: It will grow to fill remaining space

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 pencil holder containing several pens and pencils. In the top-right corner of the desk is a white mug filled with a frothy beverage. The text "Tailwind Grid" is centered on the white paper.

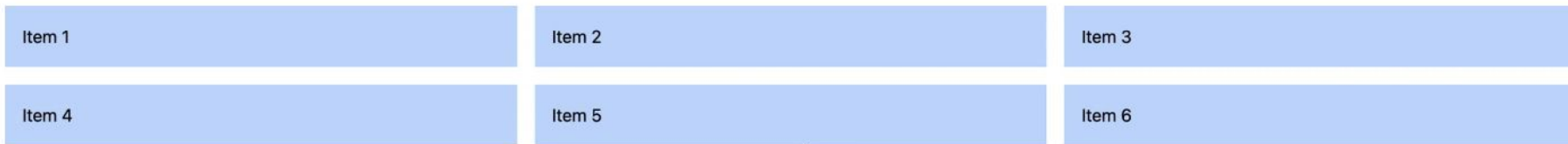
Tailwind Grid

Enabling Grid & Defining Tracks

- Apply grid to the parent container.
- Define Columns (Most Common)
 - `grid-cols-{number}` -> e.g., `grid-cols-3` (3 equal columns)
 - Tailwind uses `repeat({number}, minmax(0, 1fr))` under the hood.
- Define Rows (Optional, often implicit):
 - `grid-rows-{number}` -> e.g., `grid-rows-2`
- Use arbitrary values for complex tracks: `grid-cols-[200px_1fr_auto]`
- Quick Check: How many columns will this grid have?
 - `<div class="grid grid-cols-4"> ... </div>`
 - Answer: 4

Grid Gaps

- Spacing between grid cells (gutters)
- Works just like Flexbox gap
 - `gap-{size}` - Row and Column gaps)
 - `gap-x-{size}` - Column gap only
 - `gap-y-{size}` - Row gap only



Spanning Items Across Cells

- Make grid items occupy multiple tracks:
 - `col-span-{number}` -> e.g., `col-span-2` - Item spans 2 columns
 - `row-span-{number}` -> e.g., `row-span-2` - Item spans 2 rows
- Control exact placement (Less common, but available):
 - `col-start-{line}`, `col-end-{line}`
 - `row-start-{line}`, `row-end-{line}`
- Quick Check: How much horizontal space does 'A' take?
 - ```
<div class="grid grid-cols-3">
 <div class="col-span-2">A</div>
 <div>B</div>
</div>
```
  - Answer: 2/3rds or 2 columns

# Fine-Tuning with Arbitrary Values

- What was the purpose of [...]?
  - Define custom values not available in Tailwind's framework
- Very useful for precise layout needs
  - Flex Basis: `basis-[45%]` or `basis-[300px]`
  - Grid Tracks: `grid-cols-[200px_1fr_auto]` or `grid-rows-[auto_1fr_auto]`
  - Specific Gaps: `gap-[18px]`
  - Exact Sizing: `w-[600px]` or `h-[calc(100vh-80px)]`
- Use when standard utilities (`col-3`, `basis-1/2`, etc.) aren't exact enough before using arbitrary values. Ideally, use the standard theme values for consistency



**Demo Time!**



# Grid: Use Cases & Summary

- **Strengths**
  - Structuring the overall page layout (2D).
  - Complex layouts needing simultaneous row AND column control.
  - Dashboards, image/card galleries, forms needing precise alignment
- **The go-to for arranging elements in two dimensions**

# Break Time + Exercise!

- Task: Create a "Notification" component (Icon + Text, responsive stacking).
  - Container: Padded, border, rounded, light background.
  - Content: Circular icon left, Title+Message right (centered vertically).
  - Responsive: Icon stacks above centered text on screens
- Try it: <https://play.tailwindcss.com/>

A top-down view of a wooden desk. In the center is a large white sheet of 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.

# **Responsive Design**

# Tailwind's Responsive Prefixes

- Mobile-First: Design for small screens by default (no prefix).
- Use prefixes to override styles at larger breakpoints:
  - sm: ( $\geq 640\text{px}$ )
  - md: ( $\geq 768\text{px}$ )
  - lg: ( $\geq 1024\text{px}$ )
  - xl: ( $\geq 1280\text{px}$ )
  - 2xl: ( $\geq 1536\text{px}$ )
- Example: w-full lg:w-1/2 (Full width on mobile, half width on large screens & up)

# Applying Responsive Prefixes

- Recall the syntax
  - {breakpoint}:{utility}
- Prefix any utility class to make it responsive
- Layout Examples:
  - Stack columns: flex flex-col md:flex-row
  - Change grid columns: grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4
  - Adjust alignment: items-center lg:items-start
  - Modify spacing: p-4 md:p-6 lg:p-8
- Visibility Example
  - hidden md:block (Hide on mobile, show as block on medium+)



**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

- Which utility class creates a block-level flex container and arranges children horizontally by default?
  - A) grid
  - B) flex flex-col
  - C) flex
  - D) inline-flex

Correct Answer: C

## Question 2

- To distribute space evenly between flex items, pushing the first/last to the edges, use
  - A) justify-center
  - B) justify-evenly
  - C) items-between
  - D) justify-between

Correct Answer: D

# Question 3

- What utility creates a grid with 2 equal columns?
  - A) flex flex-2
  - B) grid grid-cols-2
  - C) col-span-2
  - D) grid-cols-[1fr\_1fr]

Correct Answer: B

# Question 4

- You have a 4-column grid. How do you make an item span the first 3 columns?
  - A) `grid-cols-3`
  - B) `col-span-3`
  - C) `cols-3`
  - D) `col-start-1 col-end-4`

Correct Answer: D is also valid but B is simpler.

# Question 5

- Using a mobile-first approach, p-2 lg:p-6 means
  - A) Padding is 6 on large screens, 2 otherwise.
  - B) Padding is 2 on large screens, 6 otherwise.
  - C) Padding is 2 always, except 6 on medium screens.
  - D) Padding is 2 always, except 6 below large screens.

Correct Answer: D



# Question 6

- Which class combination would stack items vertically on mobile but place them side-by-side on medium screens and up?
  - A) flex flex-row md:flex-col
  - B) grid grid-cols-1 md:grid-cols-2
  - C) flex flex-col md:flex-row
  - D) flex md:grid

Correct Answer: C

# Question 7

- What is the primary advantage of using gap-\* over margins for spacing flex/grid items?
  - A) gap works on inline elements.
  - B) gap only applies space between items, not at the edges.
  - C) gap allows negative values.
  - D) gap is faster for the browser to render.

Correct Answer: B

# References

- **Official Tailwind CSS Docs (Essential!):**
  - [Flexbox & Grid](#): start here. Check out the sidebar to cover more sections
  - [Responsive Design](#)
- [Tailwind Play](#)
  - Experiment!
- [Tailwind Labs YouTube](#)
  - Official Tutorials
- **CSS Concepts (Underlying Principles):**
  - [CSS-Tricks Flexbox Guide](#)
  - [CSS-Tricks Grid Guide](#)