

CHEAT SHEET

CSS FLEX & GRID - TAILWIND CSS

TABLE OF CONTENTS

1. Display Flex	5
1.1 Quotes Side-by-Side	5
Solution 1.1 Quotes Side-by-Side	5
1.2 Understanding Display Flex	5
2. Justify Content	5
2.1 Tabs Spaced Out	5
2.2 Understanding Justify Content	5
2.3 justify-content: flex-start;	5
2.4 justify-content: flex-end;	5
2.5 justify-content: center;	5
2.6 justify-content: space-between;	5
2.7 justify-content: space-around;	5
2.8 justify-content: space-evenly;	6
2.9 Card with Previous & Next Links	6
2.10 Team Profiles	6
3. Flex Wrap	6
3.1 Solution flex-wrap	6
3.1 Understanding Flex Wrap	6
3.2 Logos Wrapped	6
3.3 Solution Logos Wrapped	6
4. Align Items	7
4.1 Icon and Text	7
4.1 Solution Icon and Text	7
4.2 Understanding Align Items	7
4.3 Profile Card – Small	7
4.4 Services Section	7
4.5 Frequent Questions	8
4.6	

Center a div	8
4.7 Solution 1 Center a div	8
4.8 Solution 2 Center a div	8
5. Flex Direction	8
5.1 Possible Solution Flex Direction	8
5.2 Better Solution Flex Direction	8
5.3 Understanding Flex Direction	8
5.4 Main Axis and Cross Axis	9
5.5 Testimonial Card	9
5.6 Solution Testimonial Card	9
5.7 Alternating List of Profiles	9
5.8 Markup Alternating List of Profiles	9
5.9 Solution Alternating List of Profiles	9
6. Flex Grow	9
6.1 Markup Flex Grow	9
6.2 Solution Flex Grow	9
6.3 Understanding Flex Grow	9
6.4 Sticky Footer	10
6.5 Markup Sticky Footer	10
6.6 Solution Sticky Footer	10
6.7 Card with Header & Footer	10
6.8 Tabs Hover Effect	10
6.9 Solution Tabs Hover Effect	10
6.10 Variable Width Responsive Buttons	10
6.11 Markup Variable Width Responsive Buttons	11
6.12 Solution Variable Width Responsive Buttons	11
7. Flex Shrink	11
7.1 Itinerary	11
7.2 Markup Itinerary	11
7.3 Solution Itinerary	11
7.4 Understanding Flex Shrink	11
7.5 Profile Card – Large	11
7.6 Markup Profile Card – Large	11
7.7 Solution Profile Card – Large	11
8. Flex Basis	11
8.1 Split Screen Display	11

8.2 Markup Split Screen Display	12	Markup Profile with Rating	16
8.3 Possible Solution 1 Split Screen Display.....	12	12.5 Solution Profile with Rating.....	16
8.4 Possible Solution 2 Split Screen Display.....	12	13. Align Content	16
8.5 Better Solution Split Screen Display	12	13.1 Full Page Testimonials Section	16
8.6 Understanding Flex Basis.....	12	13.2 Markup Full Page Testimonials Section.....	17
8.7 Blog Post Display.....	12	13.3 Solution Full Page Testimonials Section	17
8.8 Markup Blog Post Display	12	13.4 Understanding Align Content	17
8.9 Solution Blog Post Display	12	14. Inline Flex.....	17
8.10 Pricing Plans.....	13	14.1 Social Media Icons	17
8.11 Markup Pricing Plans	13	14.2 Markup Social Media Icons	17
8.12 Possible Solution Pricing Plans	13	14.3 Solution Social Media Icons.....	17
8.13 Better Solution Pricing Plans	13	14.4 Understanding Inline Flex.....	17
9. Flex Shorthand Property	13	14.5 Comprehensive Examples for Flexbox.....	18
9.1 Understanding Flex.....	13	14.5.1 Article Preview	18
9.2 Navigation Bar with Centered Menu	13	14.5.2 Fitness Report	18
9.3 Markup Navigation Bar with Centered Menu	14	14.5.3 Tweet	18
9.4 Solution Navigation Bar with Centered Menu.....	14	16. Display Grid & Grid Template Columns	18
9.5 Image and Text in 2:1 Ratio	14	16.1 Full Page Gallery	18
9.6 Markup Image and Text in 2:1 Ratio.....	14	16.2 Markup Full Page Gallery	18
9.7 Solution Image and Text in 2:1 Ratio	14	16.3 Solution Full Page Gallery.....	18
10. Auto Margins.....	14	16.4 Understanding Display Grid.....	18
10.1 Notifications Menu Item.....	14	16.5 Understanding Grid Template Columns	19
10.2 Markup Notifications Menu Item	14	16.6 The CSS Property grid-template-columns & Values ..	19
10.3 Solution Notifications Menu Item	14	16.7 Layout with Sidebar.....	19
10.4-Footer with Multiple Columns	14	16.8 Markup Layout with Sidebar	19
10.5 Markup Footer with Multiple Columns	15	16.9 Solution Layout with Sidebar	19
10.6 Solution Footer with Multiple Columns.....	15	16.10 Services Grid.....	19
11. Order	15	16.11 Solution Services Grid.....	19
11.1 Responsive Navigation Bar	15	16.12 CSS Solution Services Grid	19
11.2 Markup Responsive Navigation Bar.....	15	16.13 Better Solution Services Grid.....	19
11.3 Solution Responsive Navigation Bar	15	16.14 Quick Bites Menu	19
11.4 Understanding Order.....	15	16.15 Markup Quick Bites Menu.....	19
12. Align Self	15	16.16 Solution Quick Bites Menu	20
12.1 Solution Align Self	16	17. Grid Template Rows	20
12.2 Understanding Align Self	16	17.1 Sticky Footer with Grid.....	20
12.3 Profile with Rating	16	17.2 Markup Sticky Footer with Grid	20
12.4			

17.3 Solution Sticky Footer with Grid	20	Understanding Justify Items	24
17.4 Understanding Grid Template Rows.....	20	21.5 Profile Card with Bio & Link Centered	24
18. Gap	20	21.6 Markup Profile Card with Bio & Link Centered	24
18.1 Pricing Plans with Grid	20	21.7 Solution Profile Card with Bio & Link Centered	24
18.2 Markup Pricing Plans with Grid	20	22. Align Items	24
18.3 Solution Pricing Plans with Grid.....	20	22.1 Image and Text Section	24
18.4 Understanding Column Gap	20	22.2 Markup Image and Text Section.....	24
18.5 Blog Posts Display	21	22.3 Solution Image and Text Section	24
18.6 Markup Blog Posts Display	21	22.4 Understanding Align Items in Grid	24
18.7 Responsive Solution Blog Posts Display.....	21	22.5 Featured Logos of Different Heights	25
18.8 Better Solution Blog Posts Display.....	21	22.6 Markup Featured Logos of Different Heights.....	25
18.9 Understanding Row Gap.....	21	22.7 Solution Featured Logos of Different Heights	25
18.10 Understanding Gap.....	21	23. Place Items.....	25
19. Justify Content	21	23.1 Center a div using Grid	25
19.1 Markup Justify Content	21	23.2 Markup Center a div using Grid.....	25
19.2 Solution Justify Content.....	21	23.3 Solution Center a div using Grid	25
19.3 Understanding Justify Content in Grid	21	23.4 Understanding Place Items.....	25
19.4 Shopping Cart Summary	22	24. Grid Column Start, End & Span	25
19.5 Markup Shopping Cart Summary.....	22	24.1 Horizontal Form.....	25
19.6 Solution Shopping Cart Summary	22	24.2 Markup Horizontal Form	26
20.Align Content	22	24.3 Solution Horizontal Form	26
20.1 Profile Card with Bio & Link	22	23.4 Understanding Column Start.....	26
20.2 Markup Profile Card with Bio & Link	22	23.5 Single Price Grid Component.....	26
20.3 Solution Profile Card with Bio & Link.....	22	23.6 Markup Single Price Grid Component	26
20.4 Understanding Align Content in Grid.....	22	23.7 Solution Single Price Grid Component	26
20.5 Featured Logos Center of Page.....	23	23.8 Understanding Column End	26
20.6 Markup Featured Logos Center of Page	23	23.9 Understanding Column Span.....	26
20.7 Solution Featured Logos Center of Page	23	23.10 Page Layout with Grid	27
20.8 A small variation	23	23.11 Markup Page Layout with Grid.....	27
20.9 Solution A small variation	23	23.12 Solution Page Layout with Grid	27
20.9 Better Solution A small variation	23	25. Grid Row	27
20.10 Understanding Place Content in Grid	23	25.1 Markup Grid Row	27
21. Justify Items	23	25.2 Solution Grid Row.....	27
21.1 Featured Logos of Different Widths	23	25.3 Understanding Row Start and Row End	27
21.2 Markup Featured Logos of Different Widths.....	23	25.4 Understanding Row Span	27
21.3 Solution Featured Logos of Different Widths	24	25.5 Responsive Services Section.....	27
21.4			

25.6 Markup Responsive Services Section.....	28
25.7 Solution Responsive Services Section	28
25.8 Testimonials Grid Section	28
25.9 Markup Testimonials Grid Section.....	28
25.10 Solution Testimonials Grid Section	28
26. Order	28
26.1 Markup Order	29
26.2 Solution Order	29
26.3 Understanding Order in Grid	29
27. Advanced Grid Template Values	29
27.1 Pricing Plans with Size Limits	29
27.2 HTML	29
27.3 Solution Pricing Plans with Size Limits	29
27.4 Understanding minmax()	29
27.5 Blog Post Page with Code Snippet	29
27.6 Solution Blog Post Page with Code Snippet.....	30
27.7 Responsive Grid without Media Queries	30
27.8 HTML Responsive Grid without Media Queries.....	30
27.9 Solution Responsive Grid without Media Queries.....	30
27.10 Understanding auto-fit	30
27.11 How the browser calculates	30
27.12 Understanding auto-fill.....	31
28. Grid Auto Flow	31
28.1 Analytics Section.....	31
28.2 HTML Analytics Section	31
28.3 Solution Analytics Section	31
28.4 Understanding Grid Auto Flow	31
29. Justify Self & Align Self	31
29.1 Restaurant Cards with Labels	31
29.2 Markup Restaurant Cards with Labels.....	31
29.3 Solution Restaurant Cards with Labels	31
29.4 Understanding Justify Self and Align Self.....	32
29.5 Caption at the Bottom of Image	32
29.6 Markup Caption at the Bottom of Image.....	32
29.7 Solution Caption at the Bottom of Image.....	32
30 Comprehensive Examples for Grid & Flexbox.....	32
30.1	

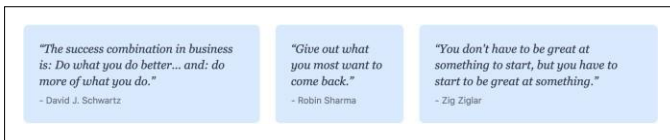
Services Section	32
30.2 Twitter Monthly Summary Card	32
30.3 Social Media Dashboard.....	32
Conclusion	32



1. DISPLAY FLEX

A parent element called flex container and at least one child element called flex item. Flexbox is a method that helps us arrange elements in one direction (horizontally or vertically) and control their dimensions, alignments, order of appearance and more.

1.1 QUOTES SIDE-BY-SIDE



SOLUTION 1.1 QUOTES SIDE-BY-SIDE

```
1 <div class="flex">
2   <div> ... </div>
3   <div> ... </div>
4   <div> ... </div>
5 </div>
```

1.2 UNDERSTANDING DISPLAY FLEX

Flexbox is a method that helps us arrange elements in one direction (horizontally or vertically) and control their dimensions, alignments, order of appearance and more. For this, we need at least two elements - a parent element called flex container and at least one child element called flex item.

Tailwind Class	CSS Property & Value	Explanation
flex	display: flex;	Setting the display property of an element to flex makes it a flex container

2. JUSTIFY CONTENT

2.1 TABS SPACED OUT

To justify the multiple elements in a flex container by dividing equal space between all elements in the container.

```
1 <div class="menu flex justify-between">
2   <a> ... </a>
3   <a> ... </a>
4   <a> ... </a>
5   <a> ... </a>
6 </div>
```

2.2 UNDERSTANDING JUSTIFY CONTENT

The utility classes justify-* are used to control spacing of the flex items in the direction they are placed.

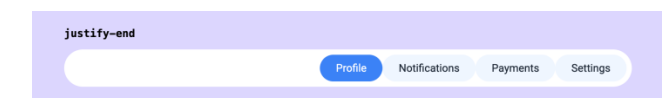
2.3 JUSTIFY-CONTENT: FLEX-START;

All items are placed at the beginning of the container with no spaces.



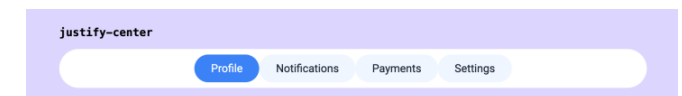
2.4 JUSTIFY-CONTENT: FLEX-END;

All items are placed at the end of the container with no spaces.



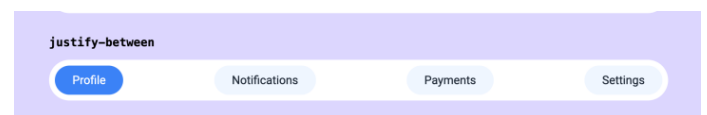
2.5 JUSTIFY-CONTENT: CENTER;

All items are placed at the center with no spaces.



2.6 JUSTIFY-CONTENT: SPACE-BETWEEN;

All items are spaced out as much as possible with first item at the beginning and last item at the end (We just saw this in action)



2.7 JUSTIFY-CONTENT: SPACE-AROUND;

Space before the flex items and after the flex items are half as much as space between the items



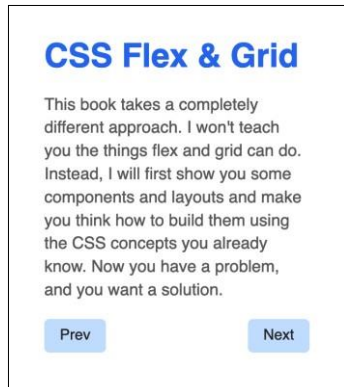
2.8 JUSTIFY-CONTENT: SPACE-EVENLY;

Space before, after and between the items are same



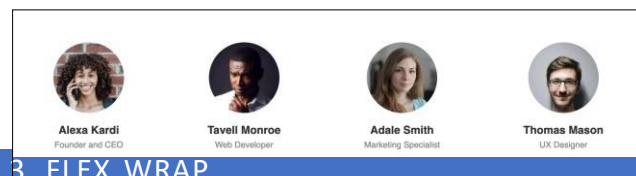
2.9 CARD WITH PREVIOUS & NEXT LINKS

"Prev" and "Next" buttons placed at the extreme ends of a card. This is a great example of flexbox with justify-* utilities used for alignment



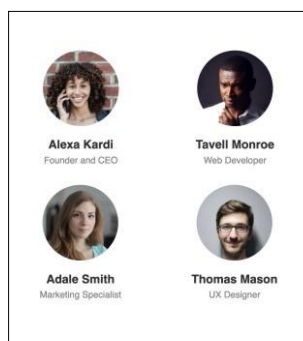
2.10 TEAM PROFILES

To adjust some space around the container, with flexbox and justify-around class for the container



3. FLEX WRAP

To move Items to next row for smaller screens. Add another class flex-wrap to the container element



3.1 SOLUTION FLEX-WRAP

```
1 <div class="container flex justify-around flex-wrap">
2   <div class="team-profile">
3     ...
4   </div>
5   <div class="team-profile">
6     ...
7   </div>
```

3.1 UNDERSTANDING FLEX WRAP

into the next row automatically.

Tailwind Class	CSS Property & Value	Explanation
flex-wrap	flex-wrap: wrap;	Items are wrapped into the next line if needed
flex-no-wrap	flex-wrap: nowrap;	Items are not wrapped even if it causes Overflow
flex-wrap-reverse	flex-wrap: wrap-reverse;	Items are wrapped in the reverse direction

3.2 LOGOS WRAPPED

to display a few logos of your clients in a row with spaces between and around them and you want them to be responsive on smaller screens. You can use justify-around for the spacing and the flex-wrap class to wrap the logos. Try out flex-wrap-reverse instead, to see the difference.



3.3 SOLUTION LOGOS WRAPPED

```

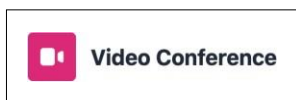
1 <div class="logos flex justify-around flex-wrap">
2   <img ... >
3   <img ... >
4   <img ... >
5   <img ... >
6 </div>

```

4. ALIGN ITEMS

4.1 ICON AND TEXT

An icon and text placed next to each other vertically centered.



4.1 SOLUTION ICON AND TEXT

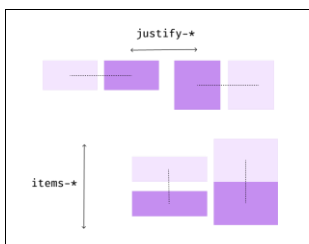
```

1 <div class="icon-wrap flex items-center">
2   <span>...</span>
3   <span>...</span>
4 </div>

```

4.2 UNDERSTANDING ALIGN ITEMS

justify-* can be used to space out the items horizontally, and items-* can be used to align items vertically.



Tailwind Class	CSS Property & Value	Explanation
items-stretch	align-items: stretch;	All items are stretched to fill the container
items-center	align-items: center;	All items are aligned to the center of the Container
items-start	align-items: flex-start;	All items are aligned to the beginning of the container (at the top in

		case of the above example)
items-end	align-items: flex-end;	All items are aligned to the end of the container (at the bottom in case of the above example)
items-baseline	align-items: baseline;	All items are positioned such that the base aligns to the end of the container

You can see the difference between these values below:



To understand the effect of items-baseline value, replace the svg icon with an alphabet and increase the font size by changing

```
1 <span class="icon"><svg>...</svg></span>
```

to

```
1 <span class="icon text-4xl">V</span>
```



The most used utilities are items-stretch and items-center

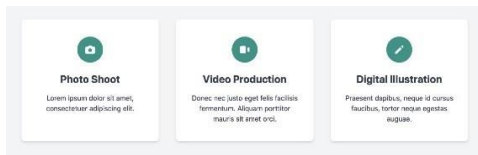
4.3 PROFILE CARD – SMALL

Many times, we need a component with an avatar and a couple of lines next to it. The items-center utility is very useful for such requirements



4.4 SERVICES SECTION

we don't want to set a fixed height to keep all the boxes the same height. you can remove the items-stretch class because it's the default. We can also use items-stretch utility class.

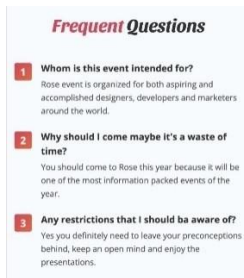


```
1 <div class="container items-end">
```

some questions are preceded by numbers aligned to the top.

4.5 FREQUENT QUESTIONS

Look at this example where some questions are preceded by numbers aligned to the top.



4.6 CENTER A DIV

With flexbox, using the justify-* and items-* utilities, it's super easy. Adding justify-center and items-center positions the child item at the center of the page horizontally and vertically.



4.7 SOLUTION 1 CENTER A DIV

```
1 <div class="w-full h-screen flex justify-center items-center">
2   <div class="item"> ... </div>
3 </div>
```

There's another way you could achieve the same result. Instead of using justify-center and items-center on container, we have used m-auto utility on the flex item to set the CSS margin property to auto.

4.8 SOLUTION 2 CENTER A DIV

```
1 <div class="w-full h-screen flex">
2   <div class="item m-auto"> ... </div>
3 </div>
```

5. FLEX DIRECTION

Here's an example you will come across a lot. Two or more items vertically centered within its container.



5.1 POSSIBLE SOLUTION FLEX DIRECTION

```
1 <div class="wrapper flex items-center">
2   <div class="w-full">
3     <a href="#" class="block link login-link">Login</a>
4     <a href="#" class="block link signup-link">Create account</a>
5   </div>
6 </div>
```

5.2 BETTER SOLUTION FLEX DIRECTION

Simple add the flex, flex-col and justify-center classes to the parent div:

```
1 <div class="wrapper flex flex-col justify-center">
2   <a href="#" class="link login-link">Login</a>
3   <a href="#" class="link signup-link">Create account</a>
4 </div>
```

5.3 UNDERSTANDING FLEX DIRECTION

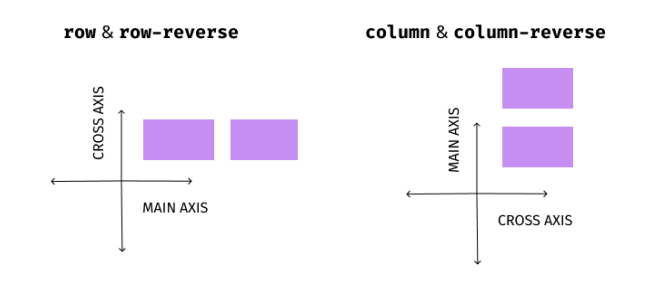
By default, all the child elements placed in a single row.

Tailwind Class	CSS Property & Value	Explanation
flex-row	flex-direction: row;	This is the default behavior. All items are placed in a single row from left to right
flex-col	flex-direction: column;	All items are placed in a single column from top to bottom
flex-row-reverse	flex-direction: row-reverse;	All items are placed in a single row from right to left

flex-col-reverse	flex-direction: col-reverse;	All items are placed in a single column from bottom to top
-------------------------	------------------------------	--

5.4 MAIN AXIS AND CROSS AXIS

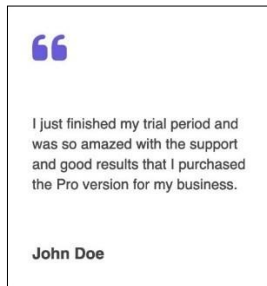
When the flex direction is row, X axis is the main axis and Y axis is the cross axis. But for flex direction column, Y axis is the main axis and X axis is the cross axis.



The justify-* utilities control spacing and alignment along the main axis, while the items-* utilities control alignment along the cross axis.

5.5 TESTIMONIAL CARD

We need to apply 4 utility classes to the container. card

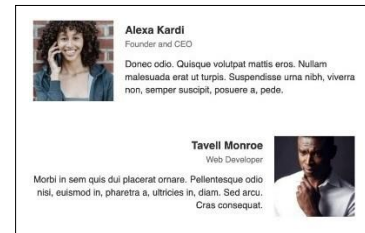


5.6 SOLUTION TESTIMONIAL CARD

```
1 <div class="card flex flex-col justify-between items-start">
2   <img ... >
3   <p> ... </p>
4   <span> ... </span>
5 </div>
```

5.7 ALTERNATING LIST OF PROFILES

Using flex-row-reverse only for even child items, you can achieve this without changing the order.



5.8 MARKUP ALTERNATING LIST OF PROFILES

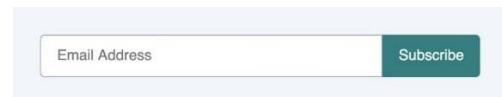
```
1 <div class="profile">
2   <img ... >
3   <div> ... </div>
4 </div>
5 <div class="profile">
6   <!-- Reverse the order -->
7   <div> ... </div>
8   <img ... >
9 </div>
```

5.9 SOLUTION ALTERNATING LIST OF PROFILES

```
1 <div class="profile flex items-center even:flex-row-reverse even:text-right">
2   <img ... >
3   <div> ... </div>
4 </div>
```

6. FLEX GROW

the text input occupies all the available horizontal space of its parent container. The class flex-grow is the first one to be used on a child element - the flex item.



6.1 MARKUP FLEX GROW

```
1 <div class="container flex">
2   <input ... >
3   <button> ... </button>
4 </div>
```

6.2 SOLUTION FLEX GROW

```
1 <input class="flex-grow" ... >
```

6.3 UNDERSTANDING FLEX GROW

By adding the flex-grow class to an element, we are changing the element's flex-grow to 1. In CSS, you can set this to any number greater than 0. This value is also called the grow factor. You can make the item occupy the left-over space (Left over width in case

of row direction, and left over height in case of column direction)

Email Address

Subscribe

In Tailwind CSS, we have only two utility classes available with respect to flex grow.

Tailwind Class	CSS Property & Value	Explanation
flex-grow	flex-grow: 1;	The item grows to occupy remaining space along the main axis
flex-grow-0	flex-grow: 0;	This is the default. The item occupies only as much space as needed even if more space is available

6.4 STICKY FOOTER

flexbox for the whole layout, with column direction and adding flex-grow to the main content.

Main Content



Footer

6.5 MARKUP STICKY FOOTER

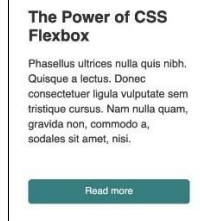
```
1 <div class="container">
2   <div class="main">... </div>
3   <footer>... </footer>
4 </div>
```

6.6 SOLUTION STICKY FOOTER

```
1 <div class="container min-h-screen flex flex-col">
2   <div class="main flex-grow">... </div>
3   <footer>... </footer>
4 </div>
```

6.7 CARD WITH HEADER & FOOTER

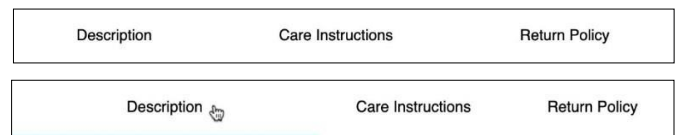
Let's say we have a card of a specific height - like a blogpost preview with title (as header), an excerpt and a "Read more" button (as footer). The excerpt might sometimes be small, but you would want your button to "stick" to the bottom of the card regardless of the height of the excerpt.



The code is very similar to our Sticky Footer example.

6.8 TABS HOVER EFFECT

Each tab has a variable width depending on the text. Once hovered, the active tab expands while the other two.



6.9 SOLUTION TABS HOVER EFFECT

Initially, all the tabs are set to flex-grow: 1 and on hover, we increase the value of flex-grow to any number depending on how wide you want the active tab to be.

```
1 <ul class="flex">
2   <li class="flex-grow hover:flex-grow-[3]">... </li>
3   <li class="flex-grow hover:flex-grow-[3]">... </li>
4   <li class="flex-grow hover:flex-grow-[3]">... </li>
5 </ul>
```

6.10 VARIABLE WIDTH RESPONSIVE BUTTONS

this is very easy with flex-grow. Also, when you set the flex-wrap property to wrap, you get a responsive solution without using any media queries.



6.11 MARKUP VARIABLE WIDTH RESPONSIVE BUTTONS

```
1 <div class="container">
2   <button type="button"> ... </button>
3   <button type="button"> ... </button>
4   <button type="button"> ... </button>
5 </div>
```

6.12 SOLUTION VARIABLE WIDTH RESPONSIVE BUTTONS

```
1 <div class="container flex flex-wrap">
2   <button type="button flex-grow"> ... </button>
3   <button type="button flex-grow"> ... </button>
4   <button type="button flex-grow-[2]"> ... </button>
5 </div>
```

7. FLEX SHRINK

7.1 ITINERARY

Here's a simple itinerary component with the description on left and time on right.



7.2 MARKUP ITINERARY

```
1 <div class="container">
2   <div> ... </div>
3   <span>10:00 AM</span>
4 </div>
```

7.3 SOLUTION ITINERARY

A HTML solution to this is to wrap the time in `<no> tags. This will prevent the time span from shrinking.`

```
1 <div class="container flex items-start">
2   <div> ... </div>
3   <span class="flex-shrink-0">10:00 AM</span>
4 </div>
```

7.4 UNDERSTANDING FLEX SHRINK

The default behavior of flex items is to shrink to fit in a single row or a single column of the container (unless `flex-wrap` is set to `wrap`). The way grow factor specifies how much additional space the item

should occupy; the shrink factor specifies how much space should be reduced from the flex item's initial width.

In Tailwind CSS, we have only two utility classes available with respect to flex shrink.

Tailwind Class	CSS Property & Value	Explanation
flex-shrink	flex-shrink: 1;	This is the default. The item shrinks along the main axis to fit in a single row.
flex-shrink-0	flex-shrink: 0;	The item does not shrink even if it causes the container to overflow

7.5 PROFILE CARD – LARGE

If you add a long description instead of small text, the image on the left shrinks to become an oval on smaller screens



Matt Cooper

A front end web developer from New York, USA. Currently working as a freelancer. Drop a mail or say hello 🙌

7.6 MARKUP PROFILE CARD – LARGE

```
1 <div class="profile">
2   <img ... >
3   <div> ... </div>
4 </div>
```

7.7 SOLUTION PROFILE CARD – LARGE

```
1 <div class="profile flex items-center">
2   <img class="flex-shrink-0" ... >
3   <div> ... </div>
4 </div>
```

8. FLEX BASIS

8.1 SPLIT SCREEN DISPLAY

On large screens, the page is split up horizontally and on smaller screens, it's split up vertically.



8.2 MARKUP SPLIT SCREEN DISPLAY

```
1 <div class="container">
2   <div class="split"> ... </div>
3   <div class="split"> ... </div>
4 </div>
```

8.3 POSSIBLE SOLUTION 1 SPLIT SCREEN DISPLAY

```
1 <div class="container w-full h-screen flex flex-col md:flex-row">
2   <div class="split h-1/2 md:w-1/2 md:h-full"> ... </div>
3   <div class="split h-1/2 md:w-1/2 md:h-full"> ... </div>
4 </div>
```

8.4 POSSIBLE SOLUTION 2 SPLIT SCREEN DISPLAY

```
1 <div class="container w-full h-screen flex flex-col md:flex-row">
2   <div class="split flex-grow"> ... </div>
3   <div class="split flex-grow"> ... </div>
4 </div>
```

8.5 BETTER SOLUTION SPLIT SCREEN DISPLAY

```
1 <div class="container w-full h-screen flex flex-col md:flex-row">
2   <div class="split basis-1/2"> ... </div>
3   <div class="split basis-1/2"> ... </div>
4 </div>
```

But basis-1/2 is not yet a utility class in v2.2.15. It will soon be added to v3+. Until then, we can add custom CSS like this

```
1 @layer utilities {
2   .basis-1\2 {
3     flex-basis: 50%;
4   }
5 }
```

8.6 UNDERSTANDING FLEX BASIS

The property flex-basis is another one that can be defined on the flex item along with flex-grow and flex-shrink. this property sets the initial size of the flex item - that is, width in case of row direction and height in case of column direction. Along with flex-grow and flex-shrink, this property helps decide the size of the flex item.

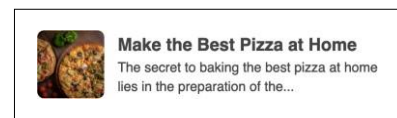
By default, the value of flex-basis is auto, which means the size is auto-calculated based on the width or height utilities. Once the basis-* utilities are added to the upcoming version in Tailwind CSS, you can

use it similar to the width and height utilities. Some of the example values are here:

Tailwind Class	CSS Property & Value	Explanation
basis-auto	flex-basis: auto;	This is the default value. The size is auto-calculated
basis-0	flex-basis: 0;	We will soon see a use-case for 0 value
basis-full	flex-basis: 100%;	The size is 100%
basis-1/2	flex-basis: 50%;	Percentage values like 25% , 50% , 75% , 33.33% , 66.67% and so on will be available
basis-24	flex-basis: 6rem	All the fixed values like 6rem that are available for width and height will be available

8.7 BLOG POST DISPLAY

It has an image on the left and long text on the right.



8.8 MARKUP BLOG POST DISPLAY

```
1 <div class="container">
2   <div>
3     <img ... >
4   </div>
5   <div> ... </div>
6 </div>
```

8.9 SOLUTION BLOG POST DISPLAY

```

1 <div class="container flex items-center">
2   <div class="mr-4 basis-20 flex-shrink-0">
3     <img ... >
4   </div>
5   <div> ... </div>
6 </div>

```

For a version lower than v3, we need the following custom styles too:

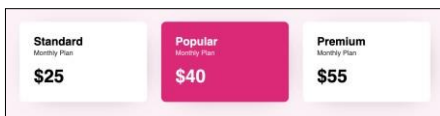
```

1 @layer utilities {
2   .basis-20 {
3     flex-basis: 5rem;
4   }
5 }

```

8.10 PRICING PLANS

Three equally sized blocks with margins in between is a very common pattern.



8.11 MARKUP PRICING PLANS

```

1 <div class="container">
2   <div class="plan"> ... </div>
3   <div class="plan"> ... </div>
4   <div class="plan"> ... </div>
5 </div>

```

8.12 POSSIBLE SOLUTION PRICING PLANS

Set basis-1/3 to all the flex items along with mx-4 for spacing between the plans:

```

1 <div class="container">
2   <div class="plan mx-4 basis-1/3"> ... </div>
3   <div class="plan mx-4 basis-1/3"> ... </div>
4   <div class="plan mx-4 basis-1/3"> ... </div>
5 </div>

```

for versions lower than 3.0, we need these custom styles:

```

1 @layer utilities {
2   .basis-1\3 {
3     flex-basis: 33.333333%;
4   }
5 }

```

8.13 BETTER SOLUTION PRICING PLANS

```

1 <div class="container">
2   <div class="plan mx-4 basis-0 flex-grow flex-shrink"> ... </div>
3   <div class="plan mx-4 basis-0 flex-grow flex-shrink"> ... </div>
4   <div class="plan mx-4 basis-0 flex-grow flex-shrink"> ... </div>
5 </div>

```

9. FLEX SHORTHAND PROPERTY

Instead of using three separate utilities flex-grow-*, flex-shrink-* and basis-*, we can make use of a single flex-* shorthand utility.

```

1 <div class="plan mx-4 flex-1"> ... </div>

```

9.1 UNDERSTANDING FLEX

The flex-* utility classes control how flex items both grow and shrink along with specifying an initial size. In Tailwind CSS, we have four of these utility classes that cover most of the use cases.

Tailwind Class	CSS Property & Value	Explanation
flex-1	flex: 1 1 0%;	Flex item grows and shrinks as needed ignoring the initial size. If this is used on multiple items, all the items take up equal space.
flex-auto	flex: 1 1 auto;	Flex item grows and shrinks as needed considering the initial size. If this is used on multiple items, all the items take up space based on their content.
flex-initial	flex: 0 1 auto;	This is the default. The item shrinks when space is less but does not grow when there's space available. Initial size is auto-calculated.
flex-none	flex: none;	The item does not grow, nor shrink.

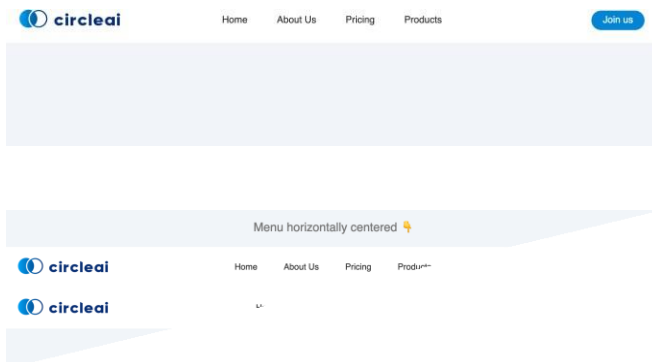
it's good to understand the syntax of the CSS flex property if you ever need to customize. The flex property may be specified using one, two, or three values separated by spaces.

Syntax

flex: <flex-grow> <flex-shrink> <flex-basis>

9.2 NAVIGATION BAR WITH CENTERED MENU

Though it looks simply, it's not straightforward to implement.



9.3 MARKUP NAVIGATION BAR WITH CENTERED MENU

```
1 <header>
2   <a>
3     <img ... >
4   </a>
5   <ul> ... </ul>
6   <span>
7     <button>... </button>
8   </span>
9 </header>
```

9.4 SOLUTION NAVIGATION BAR WITH CENTERED MENU

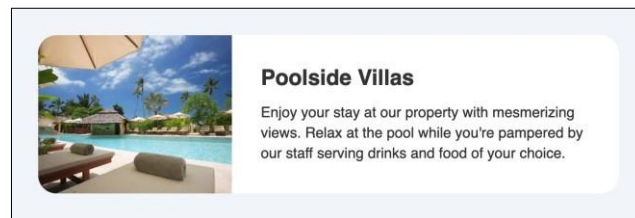
We want to achieve the first result. So how do we do it? If the elements a and span are of same width, then justify-between will help us achieve the desired result. Along with the flex-1 utility, we also need text-right for the span element to push the button to the right of the span

Markup

```
1 <header class="flex justify-between items-center">
2   <a class="flex-1">
3     <img ... >
4   </a>
5   <ul> ... </ul>
6   <span class="flex-1 text-right">
7     <button>... </button>
8   </span>
9 </header>
```

9.5 IMAGE AND TEXT IN 2:1 RATIO

You must have seen so many components with two elements placed side-by-side with widths in the ratio 2:1 or 1:2. Here's one such example



9.6 MARKUP IMAGE AND TEXT IN 2:1 RATIO

```
1 <div class="container">
2   <img ... >
3   <div class="details"> ... </div>
4 </div>
```

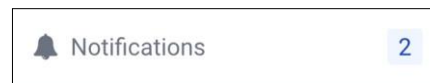
9.7 SOLUTION IMAGE AND TEXT IN 2:1 RATIO

```
1 <div class="container flex ">
2   <img class="flex-1 w-full object-cover" ... >
3   <div class="flex-[2] details"> ... </div>
4 </div>
```

10. AUTO MARGINS

10.1 NOTIFICATIONS MENU ITEM

Here's an example of a very small component with icon and text on left, and a count on right



10.2 MARKUP NOTIFICATIONS MENU ITEM

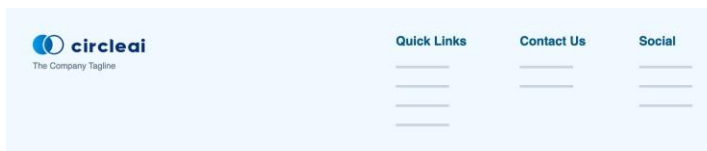
```
1 <div class="container">
2   <i> ... </i>
3   <span class="text">Notifications</span>
4   <span class="count">2</span>
5 </div>
```

10.3 SOLUTION NOTIFICATIONS MENU ITEM

```
1 <div class="container flex align-center">
2   <i> ... </i>
3   <span>Notifications</span>
4   <span class="count ml-auto">2</span>
5 </div>
```

10.4-FOOTER WITH MULTIPLE COLUMNS

This is another common footer structure with logo on the left and a few columns "pushed" to the right.



10.5 MARKUP FOOTER WITH MULTIPLE COLUMNS

```
1 <footer>
2   <div class="footer-col"> ... </div>
3   <div class="footer-col"> ... </div>
4   <div class="footer-col"> ... </div>
5   <div class="footer-col"> ... </div>
6 </footer>
```

10.6 SOLUTION FOOTER WITH MULTIPLE COLUMNS

```
1 <footer class="flex">
2   <div class="footer-col"> ... </div>
3   <div class="footer-col ml-auto"> ... </div>
4   <div class="footer-col"> ... </div>
5   <div class="footer-col"> ... </div>
6 </footer>
```

11. ORDER

11.1 RESPONSIVE NAVIGATION BAR

Assume that you have just 3 links in the navigation bar and you want those links to appear in the second row on mobile screens.



11.2 MARKUP RESPONSIVE NAVIGATION BAR

```
1 <header>
2   <a>
3     <img ... >
4   </a>
5   <ul> ... </ul>
6   <span>
7     <button> ... </button>
8   </span>
9 </header>
```

Tailwind CSS, on small screens we first need to do two things: 1. Change the order of the ul element to appear last 2. Make the ul element occupy full

width. Then using the md: prefix, 1. Set the order of ul back to 0 2. Set the width of ul back to auto

11.3 SOLUTION RESPONSIVE NAVIGATION BAR

```
1 <ul class="order-last flex-[100%] md:order-none md:flex-auto"> ...
  </ul>
```

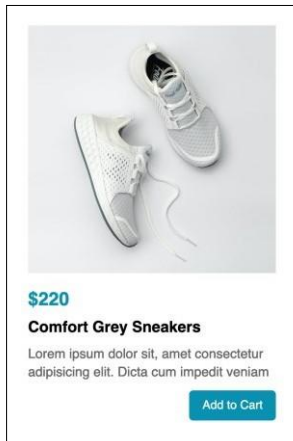
11.4 UNDERSTANDING ORDER

Some of the common utilities for order are here:

Tailwind Class	CSS Property & Value	Explanation
order-1	order: 1;	Any number from 1 to 12 are available similarly using order-2, order-3
order-first	order: -9999;	The item gets placed at the beginning because the value is a large negative number
order-last	order: 9999;	The item gets placed at the end because the value is a large positive number
order-none	order: 0;	This is the default.

12. ALIGN SELF

By default, all the elements are stretched full width (along the cross axis). But you want the button alone to be pushed to the right instead of stretching full width.



12.1 SOLUTION ALIGN SELF

```

1 <div class="container flex flex-col">
2   <img ... >
3   <span> ... </span>
4   <h3> ... </h3>
5   <p> ... </p>
6   <button class="self-end"> ... </button>
7 </div>

```

12.2 UNDERSTANDING ALIGN SELF

The self-* utilities for a flex item are similar to the items-* utilities. These classes override the items-* classes applied to the parent container

Tailwind Class	CSS Property & Value	Explanation
self-stretch	align-self: stretch;	The item is stretched to fill the container along the cross axis
self-center	align-self: center;	The item is placed at the center of the container along the cross axis
self-start	align-self: flex-start;	The item is placed at the beginning of the container (at the top for row direction and at the left for column direction)
self-end	align-self: flex-end;	The item is placed at the end of the container (at the bottom for row

		direction and at the right for column direction)
self-baseline	align-self: baseline;	The item is positioned such that the base aligns to the end of the container (applies only for row direction)

12.3 PROFILE WITH RATING

This one has a rating at the top right corner of the card. While the image and text are center aligned vertically, the rating is aligned to the top.



12.4 MARKUP PROFILE WITH RATING

```

1 <div class="container">
2   <img ... >
3   <div> ... </div>
4   <div class="rating"> ... </div>
5 </div>

```

12.5 SOLUTION PROFILE WITH RATING

```

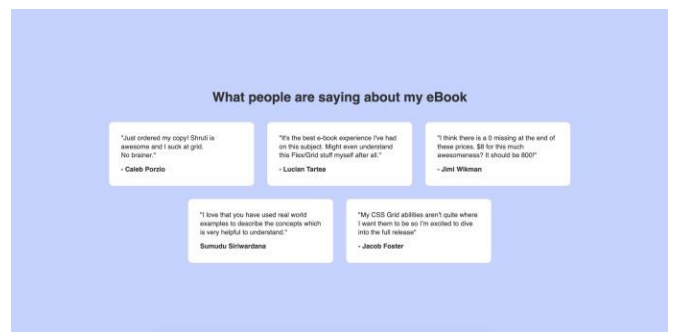
1 <div class="container flex items-center">
2   <img ... >
3   <div> ... </div>
4   <div class="rating ml-auto self-start"> ... </div>
5 </div>

```

13. ALIGN CONTENT

13.1 FULL PAGE TESTIMONIALS SECTION

Let's say you have a few testimonial cards as flex items wrapped in multiple rows. You want these items to be center aligned vertically in a full height page.



13.2 MARKUP FULL PAGE TESTIMONIALS SECTION

```
1 <div class="container">
2   <div class="testimonial">...</div>
3   <!-- Four more testimonial divs -->
4 </div>
```

13.3 SOLUTION FULL PAGE TESTIMONIALS SECTION

```
1 <div class="container flex flex-wrap justify-center content-center">
2   ...
3 </div>
```

13.4 UNDERSTANDING ALIGN CONTENT

The content-* utilities are used on the flex container for aligning multi-line flex items along the cross axis. It works only for flex items that flow into multiple rows or columns. The available utilities are mentioned below:

Tailwind Class	CSS Property & Value	Explanation
content-start	align-content: flex-start;	The items are packed at the beginning of the container
content-end	align-content: flex-end;	The items are packed at the end of the container
content-center	align-content: center;	The items are packed at the center of the Container
content-between	align-content: space-between;	The rows / columns are spaced out as much as possible with first line at the beginning and last line at the end
content-around	align-content: space-around;	Space at the beginning and the end are half as much as space between the lines

content-evenly

align-content: space-evenly;

Space at the beginning, end and between the lines are same

14. INLINE FLEX

14.1 SOCIAL MEDIA ICONS

Let's say you want a row of rounded icons with each icon placed at the exact center within the circle like this.



14.2 MARKUP SOCIAL MEDIA ICONS

```
1 <a class="icon flex justify-center items-center" href="#">
2   <i class="fa fa-twitter"></i>
3 </a>
4 <a class="icon flex justify-center items-center" href="#">
5   <i class="fa fa-linkedin"></i>
6 </a>
7 <a class="icon flex justify-center items-center" href="#">
8   <i class="fa fa-github"></i>
9 </a>
```

14.3 SOLUTION SOCIAL MEDIA ICONS

Now simply change flex to inline-flex

```
1 <a class="icon inline-flex justify-center items-center" href="#">
2   ...
3 </a>
4 <a class="icon inline-flex justify-center items-center" href="#">
5   ...
6 </a>
7 <a class="icon inline-flex justify-center items-center" href="#">
8   ...
9 </a>
```

14.4 UNDERSTANDING INLINE FLEX

the utility inline-flex does not affect the flex items. It makes the flex container itself behave like an inline element instead of a block element.

Tailwind Class	CSS Property & Value	Explanation
inline-flex	display: inline-flex;	Makes the flex container itself behave like an inline element

14.5 COMPREHENSIVE EXAMPLES FOR FLEXBOX

14.5.1 ARTICLE PREVIEW

Use the utilities flex, flex-col, align-items, ml-auto and arbitrary values for flex-* to achieve the 40% and 60% widths.

14.5.2 FITNESS REPORT



Use the utilities flex, flex-wrap, flex-* for the outer flexbox. Use flex-col, justify-* and min-w-* to allow the blocks to wrap.

14.5.3 TWEET

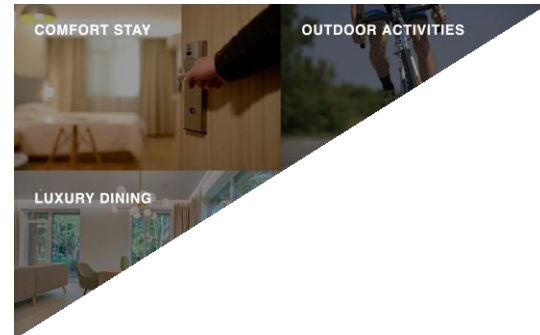
There's so much to learn from a single "tweet" design. This is an exact mockup of a tweet in the timeline on the Twitter web app (except for the font).



Here you will need to create three flexbox containers! One for the entire tweet. Two for the name, handle, date and options. And third one for the row with actions having "reply", "retweet" etc. Use the utilities flex, justify-*, items-* and auto margins.

16. DISPLAY GRID & GRID TEMPLATE COLUMNS

16.1 FULL PAGE GALLERY



16.2 MARKUP FULL PAGE GALLERY

```
1 <div class="container min-h-screen">
2   <div class="item">...</div>
3   <!-- Three more items -->
4 </div>
```

16.3 SOLUTION FULL PAGE GALLERY

Now you need to add two utility classes to the .container element to arrange the child elements in a grid form.

```
1 <div class="container min-h-screen grid grid-cols-2">
2   ...
3 </div>
```

16.4 UNDERSTANDING DISPLAY GRID

While flexbox helps us arrange elements in one dimension (row or column), grid is a method that helps us arrange and align elements in both the dimensions with rows and columns. we need at least two elements - a parent element called grid container and at least one child element called grid item

Tailwind Class	CSS Property & Value	Explanation
grid	display: grid;	Setting the display property of an element to grid makes it a grid container

16.5 UNDERSTANDING GRID TEMPLATE COLUMNS

The utilities `grid-cols-*` is used to specify how many columns you need and of what size each.

Tailwind Class	Explanation
<code>grid-cols-1</code>	Creates one grid column occupying full width of the container
<code>grid-cols-2</code>	Creates two grid columns occupying 50% width each
<code>grid-cols-3</code>	Creates three grid columns occupying 33.33% width each

16.6 THE CSS PROPERTY GRID-TEMPLATE-COLUMNS & VALUES

CSS Syntax:

`grid-template-columns: <value> <value> ...`

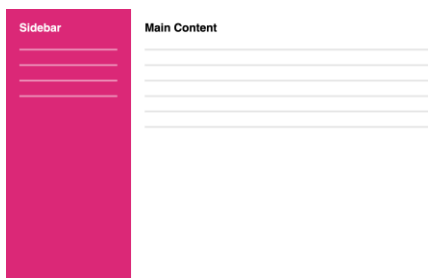
Using the CSS property `grid-template-columns`, we can specify the widths of each column in %, `px`, `rem` etc., separated by spaces

`grid-template-columns: 50% 50%`

Since we don't have a Tailwind utility for such values, we can use arbitrary values for achieving the same output

`grid-cols-[40%,60%]`

16.7 LAYOUT WITH SIDEBAR



16.8 MARKUP LAYOUT WITH SIDEBAR

```
1 <div class="container min-h-screen">
2   <div class="sidebar">... </div>
3   <div class="main">... </div>
4 </div>
```

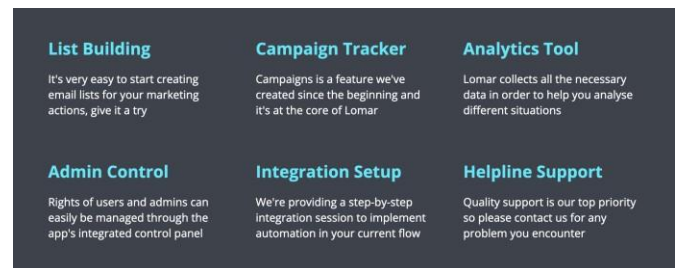
16.9 SOLUTION LAYOUT WITH SIDEBAR

```
1 <div class="container min-h-screen grid grid-cols-[22rem,1fr]">
2   ...
3 </div>
```

The `fr` Unit

The `fr` is short for fraction representing fraction of the remaining space.

16.10 SERVICES GRID



16.11 SOLUTION SERVICES GRID

```
1 <div class="container grid grid-cols-3">
2   <div class="item">... </div>
3   ← Five more items here →
4 </div>
```

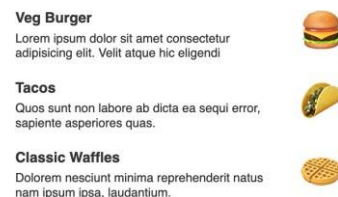
16.12 CSS SOLUTION SERVICES GRID

```
1 .container {
2   display: grid;
3   grid-template-columns: 1fr 1fr 1fr;
4 }
```

16.13 BETTER SOLUTION SERVICES GRID

```
1 .container {
2   display: grid;
3   grid-template-columns: repeat(3, 1fr);
4 }
```

16.14 QUICK BITES MENU



16.15 MARKUP QUICK BITES MENU

```

1 <div class="container">
2   <div class="item">... </div>
3   <span class="icon">... </span>
4   <div class="item">... </div>
5   <span class="icon">... </span>
6   <div class="item">... </div>
7   <span class="icon">... </span>
8 </div>

```

16.16 SOLUTION QUICK BITES MENU

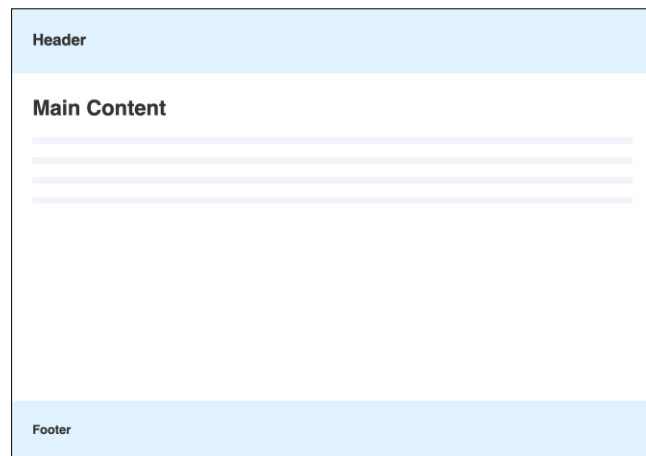
```

1 <div class="container grid grid-cols-[1fr,auto]">
2   ...
3 </div>

```

17. GRID TEMPLATE ROWS

17.1 STICKY FOOTER WITH GRID



17.2 MARKUP STICKY FOOTER WITH GRID

```

1 <div class="container min-h-screen">
2   <header>... </header>
3   <div class="main">... </div>
4   <footer>... </footer>
5 </div>

```

17.3 SOLUTION STICKY FOOTER WITH GRID

```

1 <div class="container min-h-screen grid grid-rows-[auto,1fr,auto]">
2   ...
3 </div>

```

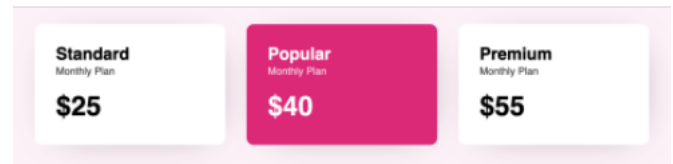
17.4 UNDERSTANDING GRID TEMPLATE ROWS

we needed rows of different heights; hence we used arbitrary values. But if we need equal height rows, we have `grid-rows-*` utility classes for that in Tailwind.

Tailwind Class	Explanation
<code>grid-rows-1</code>	Creates one grid row occupying full height of the container
<code>grid-rows-2</code>	Creates two grid rows occupying 50% height each
<code>grid-rows-3</code>	Creates three grid rows occupying 33.33% height each

18. GAP

18.1 PRICING PLANS WITH GRID



18.2 MARKUP PRICING PLANS WITH GRID

```

1 <div class="container">
2   <div class="plan">... </div>
3   <div class="plan">... </div>
4   <div class="plan">... </div>
5 </div>

```

18.3 SOLUTION PRICING PLANS WITH GRID

```

1 <div class="container grid grid-cols-3 gap-x-8">
2   ...
3 </div>

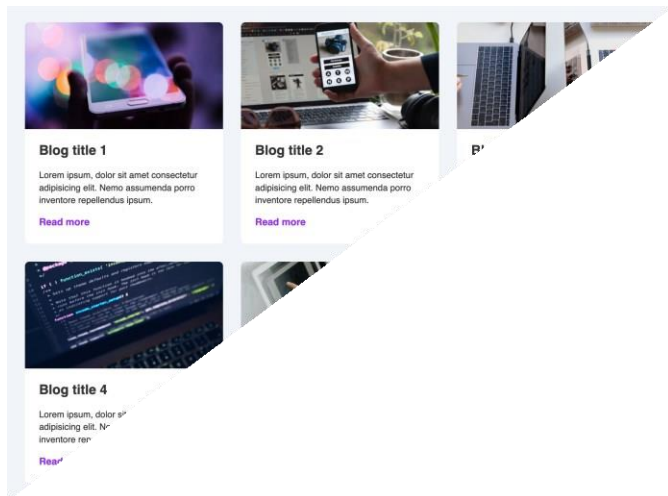
```

18.4 UNDERSTANDING COLUMN GAP

The `gap-x-*` utilities set the size of the horizontal gap (also known as gutters) between columns

Tailwind Class	CSS Property & Value	Explanation
<code>gap-x-0</code>	<code>column-gap: 0;</code>	Gap between columns is 0
<code>gap-x-4</code>	<code>column-gap: 1rem;</code>	Gap between columns is 1rem
<code>gap-x-6</code>	<code>column-gap: 1.5rem;</code>	Gap between columns is 1.5rem
<code>gap-x-8</code>	<code>column-gap: 2rem;</code>	Gap between columns is 2rem

18.5 BLOG POSTS DISPLAY



18.6 MARKUP BLOG POSTS DISPLAY

```
1 <div class="container">
2   <div class="item">... </div>
3   <!-- Five more item cards -->
4 </div>
```

18.7 RESPONSIVE SOLUTION BLOG POSTS DISPLAY

```
1 <div class="container grid sm:grid-cols-2 md:grid-cols-3 gap-x-8 gap-y-8">
2   ...
3 </div>
```

18.8 BETTER SOLUTION BLOG POSTS DISPLAY

```
1 <div class="container grid sm:grid-cols-2 md:grid-cols-3 gap-8">
2   ...
3 </div>
```

18.9 UNDERSTANDING ROW GAP

The `gap-y-*` utilities set the size of the vertical gap (also known as gutters) between rows.

18.10 UNDERSTANDING GAP

The utility `gap` is used to set the same spacing between rows and columns at once. The available utilities are similar to that of `gap-x-*` and `gap-y-*`.

19. JUSTIFY CONTENT



19.1 MARKUP JUSTIFY CONTENT

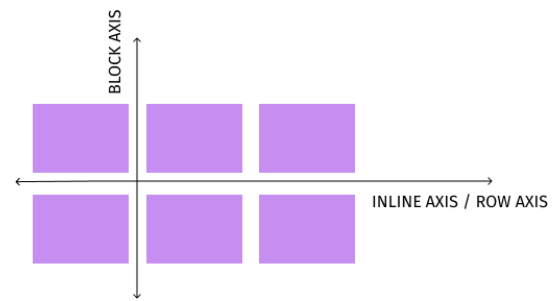
```
1 <div class="container grid grid-cols-[repeat(4,auto)] gap-12">
2   <img ... >
3   <!-- Seven more img elements -->
4 </div>
```

19.2 SOLUTION JUSTIFY CONTENT

```
1 <div class="container grid grid-cols-[repeat(4,auto)] gap-12 justify-between">
2   ...
3 </div>
```

19.3 UNDERSTANDING JUSTIFY CONTENT IN GRID

In flexbox, you have the main axis and the cross axis. Similarly, while working with grid, you have the inline axis and the block axis.

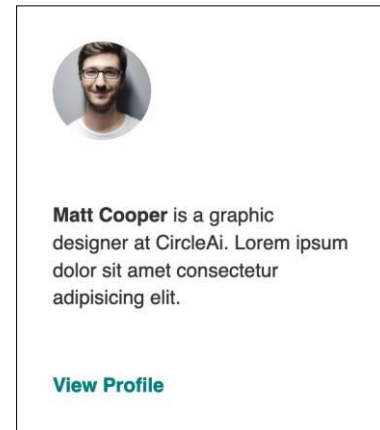


The `justify-*` utilities are used to control placement of the grid items along the inline/row axis - which is the horizontal direction.

Tailwind Class	CSS Property & Value	Explanation
justify-start	<code>justify-content: flex-start;</code>	All columns are placed at the beginning of the container
justify-end	<code>justify-content: flex-end;</code>	All columns are placed at the end of the container
justify-center	<code>justify-content: center;</code>	All columns are placed at the center
justify-between	<code>justify-content:</code>	All columns are spaced out as

	space-between;	much as possible with first column at the beginning and last column at the end (We just saw this in action)
justify-around	justify-content: space-around;	Space before the columns and after the columns are half as much as space between the columns
justify-evenly	justify-content: space-evenly;	Space before, after and between the columns are same

20.1 PROFILE CARD WITH BIO & LINK



20.2 MARKUP PROFILE CARD WITH BIO & LINK

```
1 <div class="card h-96 grid">
2   <img ... />
3   <p> ... </p>
4   <a> ... </a>
5 </div>
```

20.3 SOLUTION PROFILE CARD WITH BIO & LINK



```
1 <div class="card h-96 grid content-between">
2   ...
3 </div>
```

20.4 UNDERSTANDING ALIGN CONTENT IN GRID

The justify-* utilities control placement of columns within container, while content-* utilities control placement of rows within the container

Tailwind Class	CSS Property & Value	Explanation
content-start	align-content: flex-start;	All rows are placed at the beginning of the container
content-end	align-content: flex-end;	All rows are placed at the end of the container
content-center	align-content: center;	All rows are placed at the end

19.4 SHOPPING CART SUMMARY

	Stylish Tote Bag Brown Color Women's Tote Bag #368798	Quantity : <input type="text" value="1"/>	\$99.00
	Sunglasses Glasses with wooden frame #756328	Quantity : <input type="text" value="1"/>	\$142.00

19.5 MARKUP SHOPPING CART SUMMARY

```
1 <div class="container grid grid-cols-[repeat(4,auto)] gap-y-8 gap-x-4">
2   <img ... >
3   <div class="desc"> ... </div>
4   <div class="qty"> ... </div>
5   <div class="price"> ... </div>
6   ...
7 </div>
```

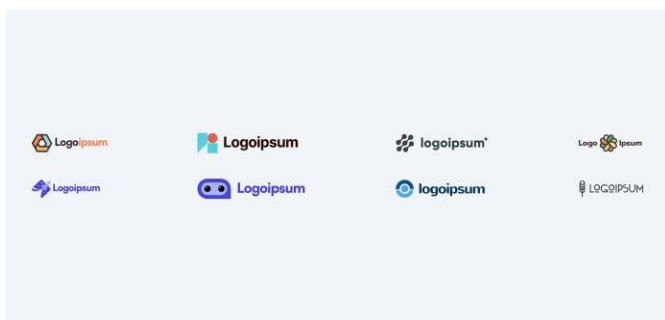
19.6 SOLUTION SHOPPING CART SUMMARY

```
1 <div class="container grid grid-cols-[repeat(4,auto)] gap-y-8 gap-x-4 justify-between">
2   ...
3   <div class="price text-right"> ... </div>
4   ...
5   <div class="price text-right"> ... </div>
6 </div>
```

20.ALIGN CONTENT

content-between	align-content: space-between;	All rows are spaced out as much as possible with first row at the beginning and last row at the end (We just saw this in action)
content-around	align-content: space-around;	Space before the rows and after the rows are half as much as space between the rows
content-evenly	align-content: space-evenly;	Space before, after and between the rows are same

20.5 FEATURED LOGOS CENTER OF PAGE



20.6 MARKUP FEATURED LOGOS CENTER OF PAGE

```
1 <div class="container min-h-screen grid grid-cols-[repeat(4,auto)]
  gap-12 justify-between">
2   <img ... >
3   <!-- Seven more img elements -->
4 </div>
```

20.7 SOLUTION FEATURED LOGOS CENTER OF PAGE

```
1 <div class="container min-h-screen grid grid-cols-[repeat(4,auto)]
  gap-12 justify-between content-center">
2   ...
3 </div>
```

20.8 A SMALL VARIATION

20.9 SOLUTION A SMALL VARIATION

```
1 <div class="container ... justify-center content-center">
2   ...
3 </div>
```

20.9 BETTER SOLUTION A SMALL VARIATION

```
1 <div class="container ... place-content-center">
2   ...
3 </div>
```

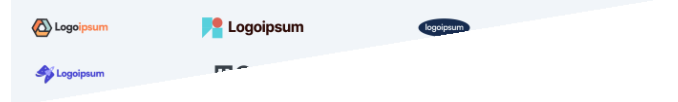
20.10 UNDERSTANDING PLACE CONTENT IN GRID

The place-content-* utilities allows you to control the spacing of grid items along both the block and inline axes at once. But this is possible only when you want the placement of rows and columns to be the same

21. JUSTIFY ITEMS

21.1 FEATURED LOGOS OF DIFFERENT WIDTHS

if you add some smaller or wider logos, you need to center align the logos in each column.



21.2 MARKUP FEATURED LOGOS OF DIFFERENT WIDTHS

```
1 <div class="container grid grid-cols-[repeat(4,auto)] gap-12 justify-
  between">
2   <img ... >
3   <!-- Seven more img elements -->
4 </div>
```

21.3 SOLUTION FEATURED LOGOS OF DIFFERENT WIDTHS

```
1 <div class="container grid grid-cols-[repeat(4,auto)] gap-12 justify-between justify-items-center">
2   ...
3 </div>
```

21.4 UNDERSTANDING JUSTIFY ITEMS

The `justify-items-*` utilities allows us to horizontally align the content within the columns, while the previous utilities `justify-*` allows us to control spacing of the entire columns. The available utilities in Tailwind are:

Tailwind Class	CSS Property & Value	Explanation
justify-items-start	<code>justify-items: start;</code>	All items are placed at the beginning of them columns (horizontally)
justify-items-end	<code>justify-items: end;</code>	All items are placed at the end of their columns (horizontally)
justify-items-center	<code>justify-items: center;</code>	All items are placed at the center of them columns (horizontally)
justify-items-stretch	<code>justify-items: stretch;</code>	The items are stretched to occupy full width of the column if possible

21.5 PROFILE CARD WITH BIO & LINK CENTERED



21.6 MARKUP PROFILE CARD WITH BIO & LINK CENTERED

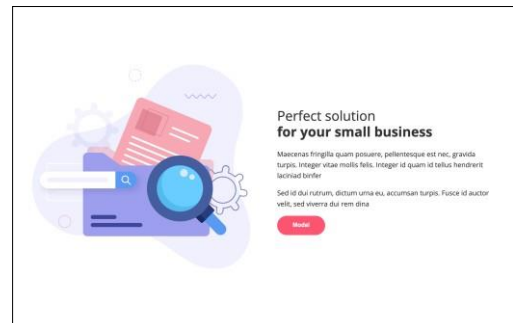
```
1 <div class="card h-96 grid content-between">
2   <img ... >
3   <p> ... </p>
4   <a> ... </a>
5 </div>
```

21.7 SOLUTION PROFILE CARD WITH BIO & LINK CENTERED

```
1 <div class="card h-96 grid content-between justify-items-center text-center">
2   ...
3 </div>
```

22. ALIGN ITEMS

22.1 IMAGE AND TEXT SECTION



22.2 MARKUP IMAGE AND TEXT SECTION

```
1 <section class="container min-h-screen grid grid-cols-2 gap-16">
2   <img ... >
3   <div> ... </div>
4 </section>
```

22.3 SOLUTION IMAGE AND TEXT SECTION

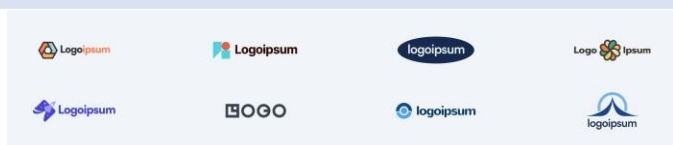
```
1 <section class="container min-h-screen grid grid-cols-2 gap-16 items-center">
2   ...
3 </section>
```

22.4 UNDERSTANDING ALIGN ITEMS IN GRID

The items-* utilities allow us to vertically align the content within the rows, while the previous property content-* allowed us to control spacing of entire rows.

Tailwind Class	CSS Property & Value	Explanation
items-stretch	align-items: stretch;	All items are stretched to fill the container
items-center	align-items: center;	All items are aligned to the center of the Container
items-start	align-items: flex-start;	All items are aligned to the beginning of the container (at the top in case of the above example)
items-end	align-items: flex-end;	All items are aligned to the end of the container (at the bottom in case of the above example)
items-baseline	align-items: baseline;	All items are positioned such that the base aligns to the end of the container (will we talk about this soon)

22.5 FEATURED LOGOS OF DIFFERENT HEIGHTS



22.6 MARKUP FEATURED LOGOS OF DIFFERENT HEIGHTS

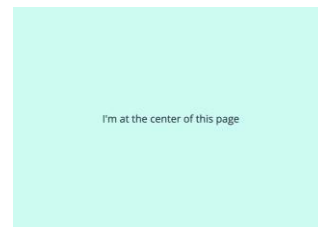
```
1 <div class="container ... justify-between justify-items-center">
2   <img ... >
3   <!-- Seven more img elements -->
4 </div>
```

22.7 SOLUTION FEATURED LOGOS OF DIFFERENT HEIGHTS

```
1 <div class="container ... justify-between justify-items-center items-center">
2   ...
3 </div>
```

23. PLACE ITEMS

23.1 CENTER A DIV USING GRID



23.2 MARKUP CENTER A DIV USING GRID

```
1 <div class="container">
2   <div class="item">
3     ...
4   </div>
5 </div>
```

23.3 SOLUTION CENTER A DIV USING GRID

```
1 <div class="container grid place-items-center">
2   <div class="item">
3     ...
4   </div>
5 </div>
```

23.4 UNDERSTANDING PLACE ITEMS

The place-items-* utilities allows you to align the items horizontally within columns and vertically within rows at once. we could use the place-items-* utilities. The available utilities are similar to that of justify-items-* and items-.*.

24. GRID COLUMN START, END & SPAN

24.1 HORIZONTAL FORM

Full Name

Full Name

Email Address

Email

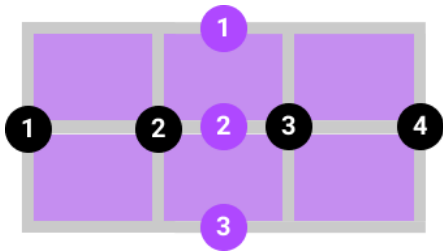
24.2 MARKUP HORIZONTAL FORM

```
1 <form class="grid grid-cols-[auto,1fr] items-center gap-y-6 gap-x-12">
2   <label></label>
3   <input .. />
4   <label></label>
5   <input .. />
6   <button ...>Create Account</button>
7 </form>
```

24.3 SOLUTION HORIZONTAL FORM

```
1 <form class="...">
2   ...
3   <button class="col-start-2" ...>Create Account</button>
4 </form>
```

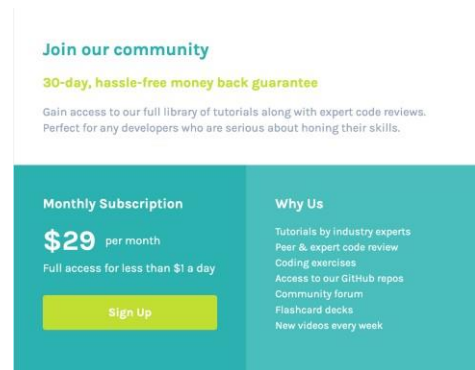
23.4 UNDERSTANDING COLUMN START



All the grid related utilities we saw so far are applied to the grid container. Now, we will see a few that are applied to the grid items. The `col-start-*` is one of them. It specifies the item's start position. Some of the available Tailwind utilities for this are:

Tailwind Class	CSS Property & Value	Explanation
<code>col-start-1</code>	<code>grid-column-start: 1;</code>	The item starts at column line 1
<code>col-start-2</code>	<code>grid-column-start: 2;</code>	The item starts at column line 2

23.5 SINGLE PRICE GRID COMPONENT



23.6 MARKUP SINGLE PRICE GRID COMPONENT

```
1 <div class="container grid sm:grid-cols-2">
2   <div class="component-header">...</div>
3   <div class="subscription">...</div>
4   <div class="why">...</div>
5 </div>
```

23.7 SOLUTION SINGLE PRICE GRID COMPONENT

```
1 <div class="container grid sm:grid-cols-2">
2   <div class="component-header sm:col-start-1 sm:col-end-3">...</div>
3   ...
4 </div>
```

23.8 UNDERSTANDING COLUMN END

The utilities `col-end-*` is another set of grid item's utilities. It specifies the item's end position. The Tailwind utilities available for this are similar to `col-start-*`. We can alternatively use another set of utilities `col-span-*` which can be used to specify the number of columns to span. This is usually used along with either `col-start-*` or `col-end-*`.

So, another solution to the previous example can be:

```
1 <div class="component-header sm:col-start-1 sm:col-span-2">...</div>
```

You can also skip the `col-start-1` here:

```
1 <div class="component-header sm:col-span-2">...</div>
```

Or

```
1 <div class="component-header sm:col-span-full">...</div>
```

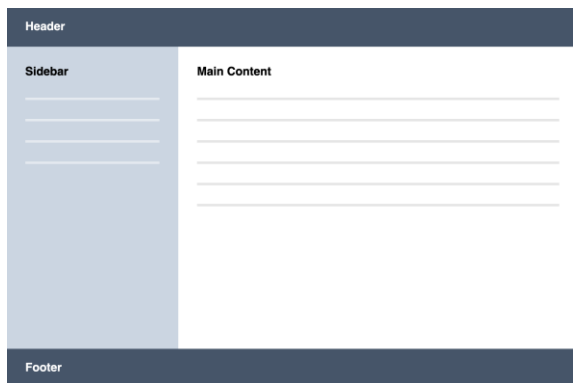
Or

```
1 <div class="component-header sm:col-span-2 sm:col-end-3">...</div>
```

23.9 UNDERSTANDING COLUMN SPAN

The `col-span-*` utilities can be used on a grid item to specify how many columns to span. This is usually used along with either `col-start-*` or `col-end-*`. But if used alone, the default start and end lines are considered. The available utilities in Tailwind are `col-span-1` up to `col-span-12` along with a helpful `col-span-full` which makes the grid item span across all the columns in the grid.

23.10 PAGE LAYOUT WITH GRID



23.11 MARKUP PAGE LAYOUT WITH GRID

```
1 <div class="container min-h-screen">
2   <header> ... </header>
3   <div class="sidebar"> ... </div>
4   <div class="main"> ... </div>
5   <footer> ... </footer>
6 </div>
```

23.12 SOLUTION PAGE LAYOUT WITH GRID

```
1 <div class="container min-h-screen grid grid-cols-[22rem,1fr] grid-rows-[auto,1fr,auto]">
2   <header class="col-span-full"> ... </header>
3   ...
4   <footer class="col-span-full"> ... </footer>
5 </div>
```

25. GRID ROW

Full Name

Message

Email Address

25.1 MARKUP GRID ROW

```
1 <form class="grid grid-cols-2 gap-6">
2   <div>
3     <label> ... </label>
4     <input ... />
5   </div>
6   <div>
7     <label> ... </label>
```

```
8     <input ... />
9   </div>
10  <div class="message-block">
11    <label> ... </label>
12    <textarea> ... </textarea>
13  </div>
14  <button> ... </button>
15 </form>
```

25.2 SOLUTION GRID ROW

```
1 <form class="grid grid-cols-2 gap-6">
2   ...
3   <div class="col-start-2 row-start-1 row-end-3">
4     <label> ... </label>
5     <textarea> ... </textarea>
6   </div>
7   <button class="col-span-full"> ... </button>
8 </form>
```

25.3 UNDERSTANDING ROW START AND ROW END

The utilities `row-start-*` and `row-end-*` are also grid item's properties. `row-start-*` specifies the item's start position and `row-end-*` specifies the item's end position with respect to row lines.

25.4 UNDERSTANDING ROW SPAN

Similar to `col-span-*` utilities, we also have `row-span-*` utilities for grid items to specify how many rows to span. This is usually used along with either `row-start-*` or `row-end-*`. But if used alone, the default start and end lines are considered. The available utilities in Tailwind are `row-span-1` up to `row-span-12` along with a helpful `row-span-full`

which makes the grid item span across all the columns in the grid.

25.5 RESPONSIVE SERVICES SECTION



List Building

It's very easy to start creating email lists for your marketing actions, give it a try

Campaign Tracker

Campaigns is a feature we've created since the beginning and it's at the core of Lomar

Analytics Tool

Lomar collects all the necessary data to help you analyse different situations

Admin Control

Rights of users and admins can easily be managed through the control panel

List Building

It's very easy to start creating email lists for your marketing actions, give it a try

Campaign Tracker

Campaigns is a feature we've created since the beginning and it's at the core of Lomar

Analytics Tool

Lomar collects all the necessary data to help you analyse different situations

Admin Control

Rights of users and admins can easily be managed through the control panel

Integration Setup

We're providing a step-by-step integration session to implement automation

Help Line Support

Quality support is our top priority so please contact us for any problem you encounter



List Building

It's very easy to start creating email lists for your marketing actions, give it a try

Campaign Tracker

Campaigns is a feature we've created since the beginning and it's at the core of Lomar

Analytics Tool

Lomar collects all the necessary data to help you analyse different situations



Admin Control

Rights of users and admins can easily be managed through the control panel

Integration Setup

We're providing a step-by-step integration session to implement automation

Help Line Support

Quality support is our top priority so please contact us for any problem you encounter

```
1 <section class="grid sm:grid-cols-2 md:grid-cols-3 gap-x-8">
2   <img class="sm:row-start-1 sm:row-end-3 md:row-end-2 md:col-start-2"
3   ... >
4   <div class="md:col-start-1 md:text-right">
5     ...
6   </div>
7   <div class="sm:col-start-2 md:col-start-3">
8     ...
9   </div>
10 </section>
```

25.8 TESTIMONIALS GRID SECTION



25.9 MARKUP TESTIMONIALS GRID SECTION

```
1 <section>
2   <div class="violet"> ... </div>
3   <div class="gray"> ... </div>
4   <div class="white"> ... </div>
5   <div class="dark"> ... </div>
6   <div class="white-long"> ... </div>
7 </section>
```

25.10 SOLUTION TESTIMONIALS GRID SECTION

```
1 <section class="grid lg:grid-cols-4 gap-8">
2   <div class="violet lg:col-span-2"> ... </div>
3   <div class="gray"> ... </div>
4   <div class="white lg:row-start-2"> ... </div>
5   <div class="dark lg:col-span-2"> ... </div>
6   <div class="white-long lg:row-start-1 lg:row-span-2 lg:col-start-4"> ... </div>
```

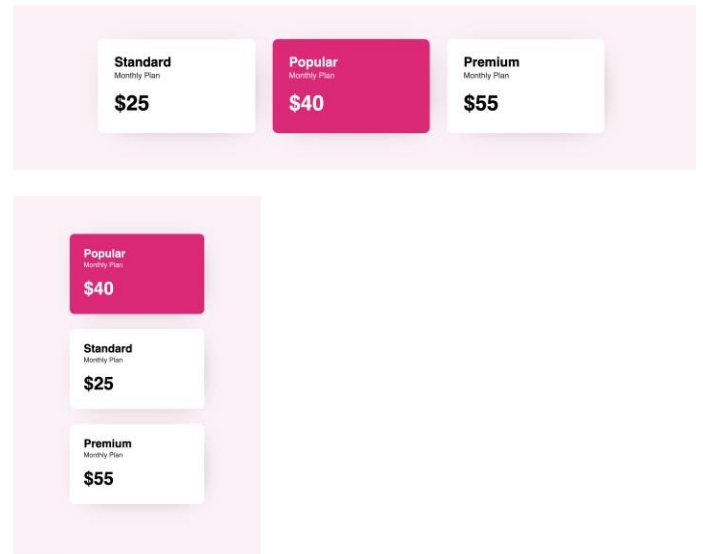
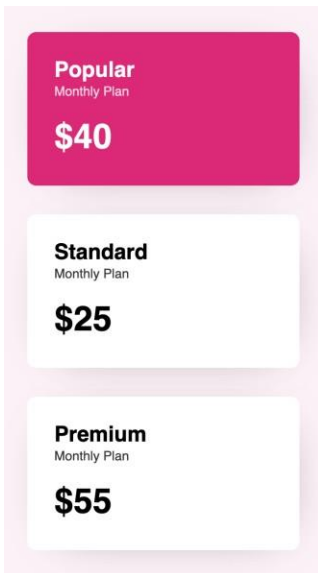
26. ORDER

we place the Popular plan first, followed by Standard and Premium while keeping the order same on desktop.

25.6 MARKUP RESPONSIVE SERVICES SECTION

```
1 <section class="grid sm:grid-cols-2 md:grid-cols-3 gap-x-8">
2   <img ... >
3   <div> ... </div>
4   <div> ... </div>
5 </section>
```

25.7 SOLUTION RESPONSIVE SERVICES SECTION



27.2 HTML

```
1 <div class="container grid sm:grid-cols-3 gap-8">
2   <div class="plan"> ... </div>
3   <div class="plan plan-highlight"> ... </div>
4   <div class="plan"> ... </div>
5 </div>
```

27.3 SOLUTION PRICING PLANS WITH SIZE LIMITS

There is no pure Tailwind solution for this. We can add a max-w- and min-w- to the plan element itself to limit the width of the cards. But then the column width still remains large which makes it impossible to center align the three cards together on larger screens. So, here's the perfect solution:

```
1 <div class="container
2     grid
3     grid-cols-[minmax(auto,18rem)]
4     sm:grid-cols-[repeat(3,minmax(auto,18rem))]
5     justify-center
6     gap-8">
7   ...
8 </div>
```

27.4 UNDERSTANDING MINMAX()

The minmax() function takes in two parameters - min and max. It specifies a size range greater than or equal to min and less than or equal to max. Both these values can be any length values in px, %, rem or even values like 1fr, min-content or max-content.

27.5 BLOG POST PAGE WITH CODE SNIPPET

26.1 MARKUP ORDER

```
1 <div class="container grid sm:grid-cols-3 gap-8">
2   <div class="plan"> ... </div>
3   <div class="plan plan-highlight"> ... </div>
4   <div class="plan"> ... </div>
5 </div>
```

26.2 SOLUTION ORDER

The plan-highlight element is the popular plan that we want to place first on mobile screens.

```
1 <div class="plan plan-highlight order-first sm:order-none"> ... </div>
```

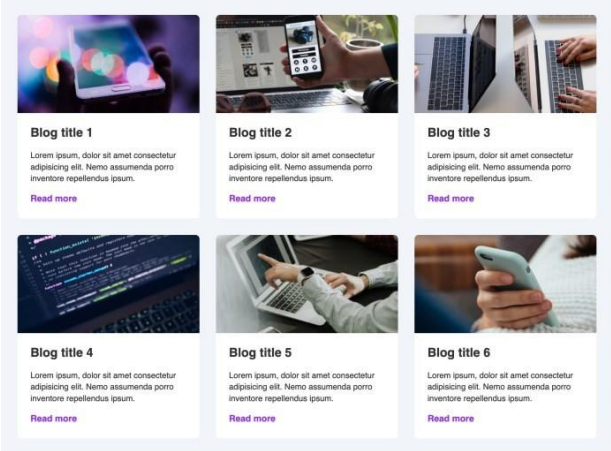
26.3 UNDERSTANDING ORDER IN GRID

The same order utilities that we saw with respect to flexbox can be used for grid items too. The value can be any number - positive or negative. The items with greater order value appear later on the web page compared to the items with lesser value irrespective of their appearance in the markup. If no order is specified, by default the value is 0 for all the elements and they follow the same order as they appear in HTML. That's what happens at sm breakpoint.

27. ADVANCED GRID TEMPLATE VALUES

27.1 PRICING PLANS WITH SIZE LIMITS

When we set max-width full and overflow-scroll to the pre-element we expect it to occupy a maximum of 100% width and display a horizontal scrollbar.



27.6 SOLUTION BLOG POST PAGE WITH CODE SNIPPET

```
1 <section class="min-h-screen grid grid-cols-[minmax(0,1fr),16rem]">
2   ...
3 </section>
```

Now that the range is 0 to 1fr , it works fine. If none of this makes sense, just remember one thing - minmax(0, 1fr) is always a better option than 1fr . Which is why, if you look at the Tailwind classes grid-cols-* , their equivalent CSS values are:

Tailwind Class	CSS Property & Value
grid-cols-1	grid-template-columns: repeat(1, minmax(0, 1fr));
grid-cols-2	grid-template-columns: repeat(2, minmax(0, 1fr));

27.7 RESPONSIVE GRID WITHOUT MEDIA QUERIES

Grid has a way to decide how many columns to create based on the space available. But there's no Tailwind solution for this too. We will be using arbitrary values again.

27.8 HTML RESPONSIVE GRID WITHOUT MEDIA QUERIES

```
1 <div class="container">
2   <div class="item">... </div>
3   <!-- Five more item cards -->
4 </div>
```

27.9 SOLUTION RESPONSIVE GRID WITHOUT MEDIA QUERIES

```
1 <div class="container grid grid-cols-[repeat(auto-fit,minmax(16rem,1fr))] gap-8">
2   ...
3 </div>
```

If you find these arbitrary values too hard to read, feel free to use custom CSS

```
1 .container {
2   ...
3   grid-template-columns: repeat(auto-fit,minmax(16rem,1fr));
4 }
```

27.10 UNDERSTANDING AUTO-FIT

The keyword auto-fit tells the browser to handle the number of columns and their sizes such that the elements will wrap when the width is not large enough to fit them in without any overflow. The 1fr in the second value of repeat ensures that in case there's more space available

27.11 HOW THE BROWSER CALCULATES

To understand the above example better, assume we have a screen width of 40rem. The container has a padding of 2rem on left and right. you really need not worry about all the above calculation. Ideally you just need one CSS rule to create a responsive grid layout of equally sized columns.

```
1 grid-template-columns: repeat(auto-fit, minmax(<fixed-width-value>, 1fr));
```

Now consider a scenario where you have just one blog post.

```
1 grid-template-columns: repeat(auto-fill, minmax(16rem, 1fr));
```

27.12 UNDERSTANDING AUTO-FILL

auto-fit distributes the remaining space leaving no empty space in the row. auto-fill creates blank columns of the same size as the items. This is very similar to auto-fit. You decide whether to use auto-fit or auto-fill.

28. GRID AUTO FLOW

28.1 ANALYTICS SECTION

11.5k

Tweets

9.3k

Followers

776

Following

28.2 HTML ANALYTICS SECTION

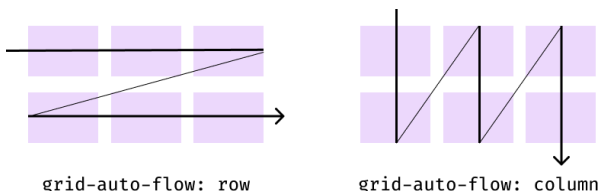
```
1 <section>
2   <span>11.5k</span>
3   <p>Tweets</p>
4   <span> ... </span>
5   <p> ... </p>
6   <span> ... </span>
7   <p> ... </p>
8 </section>
```

28.3 SOLUTION ANALYTICS SECTION

```
1 <section class="grid grid-rows-[auto,auto] grid-flow-col justify-between">
2   ...
3 </section>
```

28.4 UNDERSTANDING GRID AUTO FLOW

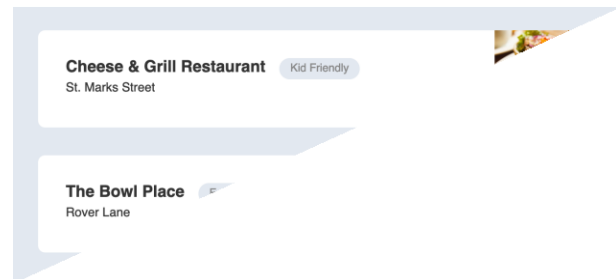
The Tailwind equivalent for this property with value row is grid-flow-row (The default) and for column is grid-flow-col



Tailwind Class	CSS Property & Value	Explanation
grid-flow-row	grid-auto-flow: row;	The items get placed one by one filling one row after the other
grid-flow-col	grid-auto-flow: column;	The items get placed one by one filling one column after the other

29.1 RESTAURANT CARDS WITH LABELS

Let's say we need to list restaurants with name, street, label and a picture just like the screenshot below. You already know how this is done with flexbox. Now let's see how to do this using grid.



29.2 MARKUP RESTAURANT CARDS WITH LABELS

```
1 <div class="container grid grid-cols-[auto,auto,1fr]">
2   <div class="info"> ... </div>
3   <span class="label"> ... </span>
4   <img ... >
5 </div>
```

29.3 SOLUTION RESTAURANT CARDS WITH LABELS

```
1 <div class="container grid grid-cols-[auto,auto,1fr]">
2   ...
3   <span class="label self-start"></span>
4   <img class="justify-self-end" ... >
5 </div>
```

29.4 UNDERSTANDING JUSTIFY SELF AND ALIGN SELF

The utilities `justify-self-*` is used on a grid item. When the content of the item is smaller than the width of the column, we can use this property to control the alignment along the row axis (horizontal direction). The utilities `self-*` is also used on a grid item. When the content of the item is shorter than the height of the row, we can use this property to control the alignment along the block axis (vertical direction). The available utilities are similar to that of `justify-items-*` and `items-*`.

29.5 CAPTION AT THE BOTTOM OF IMAGE

Here's an example where you wish to place a caption with a transparent overlay on the image sticking to the bottom.



29.6 MARKUP CAPTION AT THE BOTTOM OF IMAGE

```
1 <figure>
2   <img ... >
3   <figcaption> ... </figcaption>
4 </figure>
```

29.7 SOLUTION CAPTION AT THE BOTTOM OF IMAGE

```
1 <figure class="grid">
2   <img class="col-start-1 col-end-2 row-start-1 row-end-2" ... >
3   <figcaption class="col-start-1 col-end-2 row-start-1 row-end-2 self-
4     end">...</figcaption>
5 </figure>
```

30 COMPREHENSIVE EXAMPLES FOR GRID & FLEXBOX

30.1 SERVICES SECTION

This is a responsive services section in a grid format from a template by Inovatik. On mobile screens, two columns collapse into one. This is a great example of flexbox within a grid layout.



30.2 TWITTER MONTHLY SUMMARY CARD

Look at this card with one month summary of a Twitter profile along with some profile info. This is a good example of flexbox and grid together in a component.



30.3 SOCIAL MEDIA DASHBOARD



CONCLUSION

Congratulations! & You have reached the end of this book.

Thank you!