FLEX CATALOGUE
of Tailwind CSS

Compiled By: Muhammad Rizwan

GET IN TOUCH

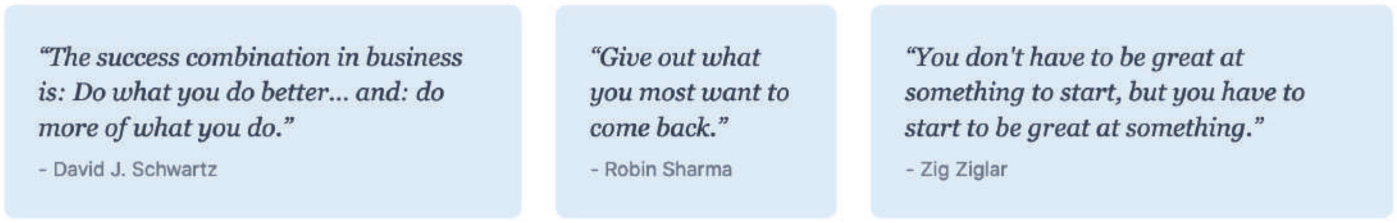
chouhdryrizwan

1. Display Flex

Example #1-A

Quotes Side-by-Side

Assume you have three motivational quotes to display on your web page in a single row (on Desktop screen size). You want the blocks to occupy the same height and hence adjust widths based on the length of each quote.



"The success combination in business is: Do what you do better... and: do more of what you do."
- David J. Schwartz

"Give out what you most want to come back."
- Robin Sharma

"You don't have to be great at something to start, but you have to start to be great at something."
- Zig Ziglar

You just need to add **flex** class to the parent container. Here's the full working demo. Tada!

```html
<div class="flex">
  <div> ... </div>
  <div> ... </div>
  <div> ... </div>
</div>
```

Demo Here

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.**

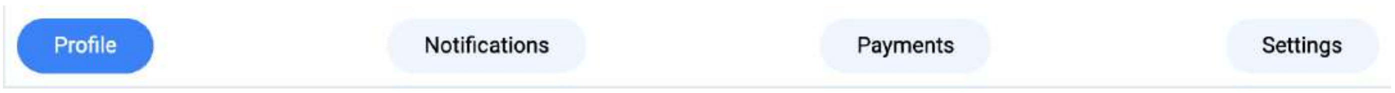
Tailwind Class	CSS Property & Value	Explanation
`flex`	`display: flex;`	Setting the `display` property of an element to `flex` makes it a flex container

2. Justify Content

Example #2-A

Tabs Spaced Out

Let's say you have a few tabs on your page and you want them to space out fully with thefirst tab on the extreme left, last tab on the extreme right and the middle ones spaced out evenly. These tabs have different widths. How would you do it?



Along with the flex class, we need to add just one more class justify-between to the same element. Let's learn more about these utilities.

```html
<div class="menu flex justify-between">
  <a> ... </a>
  <a> ... </a>
  <a> ... </a>
  <a> ... </a>
</div>
```

Demo Here

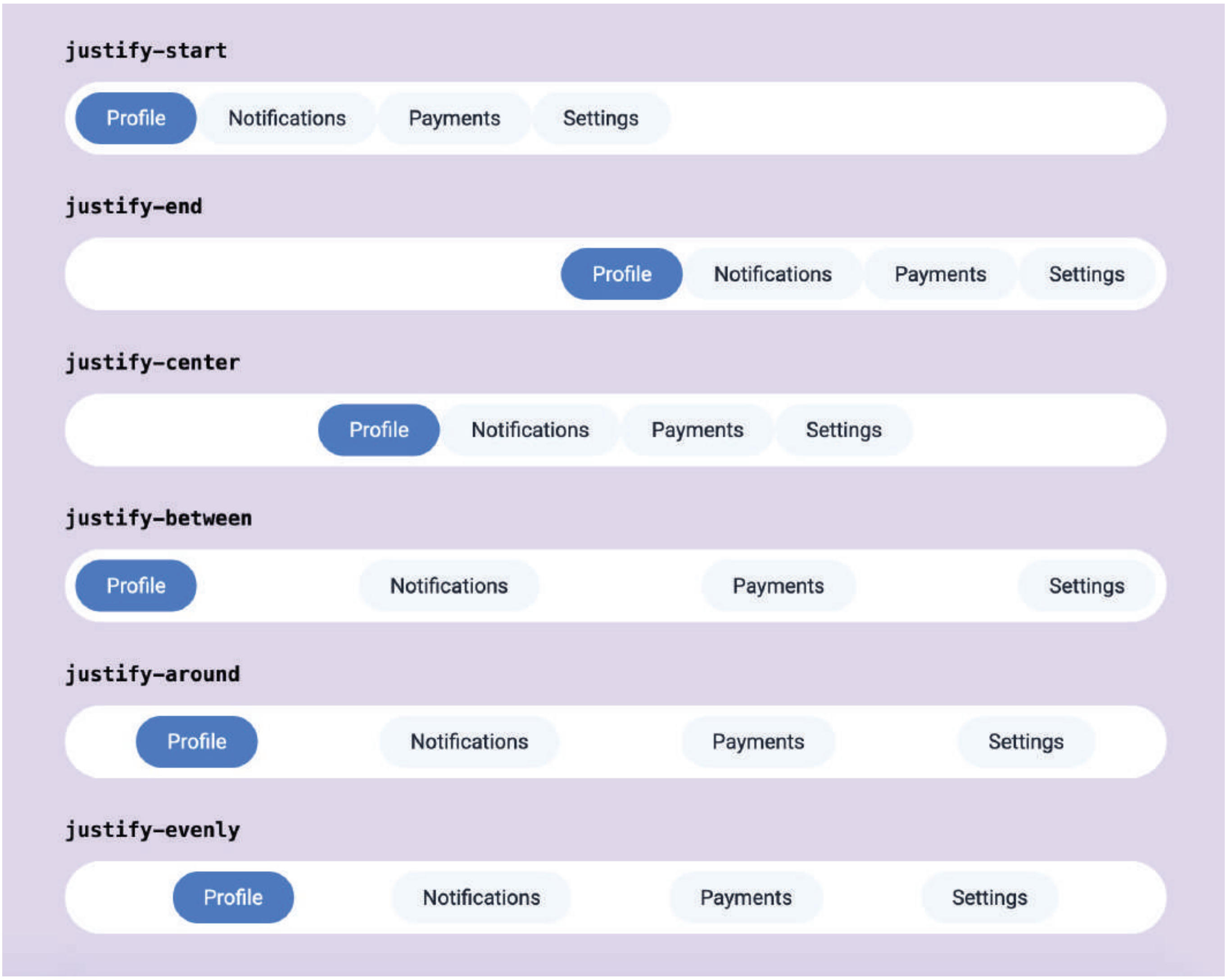
Understanding Justify Content:

Before we understand these utilities, there's something else you need to know. The moment we add a flex class to an element, we saw that the children get placed next to each other in one single row. This is a default behaviour. However, we can place them all one below the other in a single column instead. We will get to that a little later.

The utility classes justify-* are used to control spacing of the flex items in the direction they are placed. In our above example, it's the horizontal direction.

justify-between is one of the available utilities we just used. Some more utilities are mentioned below:

Tailwind Class	CSS Property & Value	Explanation
`justify-start`	`justify-content: flex-start;`	All items are placed at the beginning of the container with no spaces
`justify-end`	`justify-content: flex-end;`	All items are placed at the end of the container with no spaces
`justify-center`	`justify-content: center;`	All items are placed at the center with no spaces
`justify-between`	`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)
`justify-around`	`justify-content: space-around;`	Space *before* the flex items and *after* the flex items are half as much as space between the items
`justify-evenly`	`justify-content: space-evenly;`	Space before, after and between the items are same

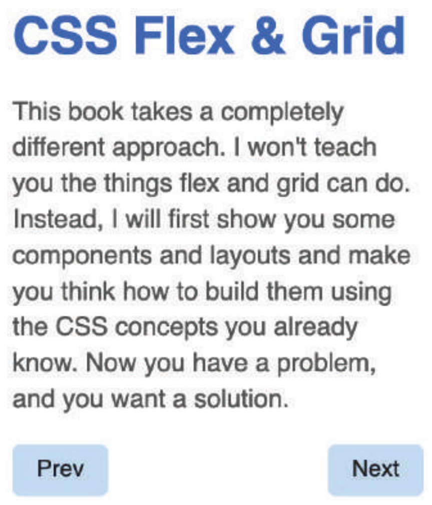


Demo Here

Example #2-B

Card with Previous & Next Links

Many times we need two elements at the extreme ends of a section / container, like these"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.



CSS Flex & Grid

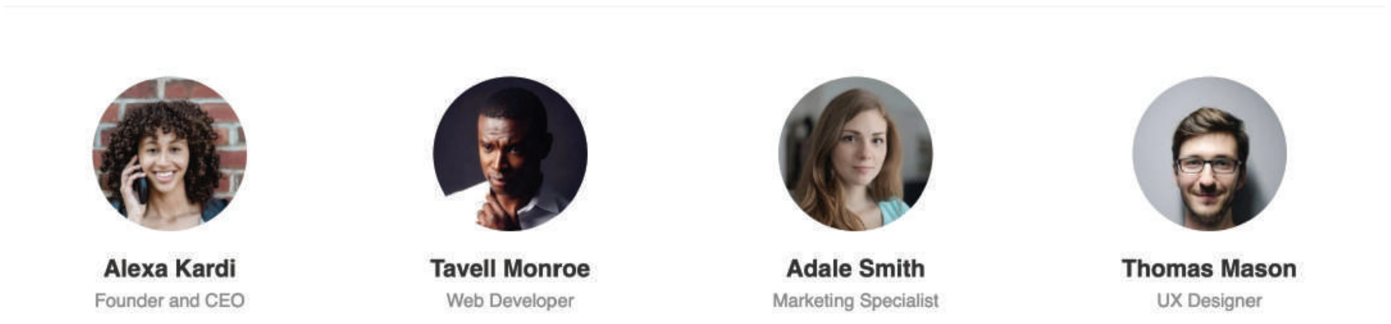
This book takes a completely different approach. I won't teach you the things flex and grid can do. Instead, I will first show you some components and layouts and make you think how to build them using the CSS concepts you already know. Now you have a problem, and you want a solution.

Demo Here

Example #2-C

Team Profiles

Assume you need to design a "Team" section to display profiles of four people as you cansee below. Notice that there is some space to the extreme right and left. This is best achieved with flexbox and justify-around class for the container.



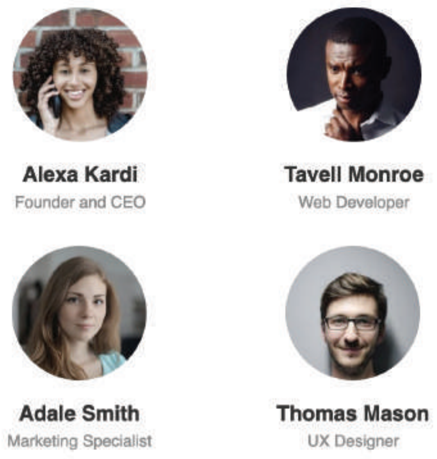
Demo Here

3. Flex Wrap

Example #3-A

Responsive Team Profiles

The above examples work great with desktop screen sizes. But try resizing the output panel to a mobile screen size and you will either notice a horizontal scrollbar or the design breaks in some way. How can we make those items move to next row for smaller screens like this?



Here's what you can do. Add another class flex-wrap to the container element:

```
<div class="container flex justify-around flex-wrap">
  <div class="team-profile">
    ...
  </div>
  <!-- Three more team profiles -->
</div>
```

Demo Here

Understanding Flex Wrap:

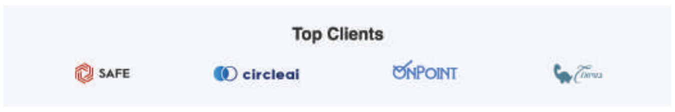
The flex-wrap utility class makes the flex items wrap if you run out of space. The default behaviour is to not wrap, which is why the child items do not move into the next row automatically.

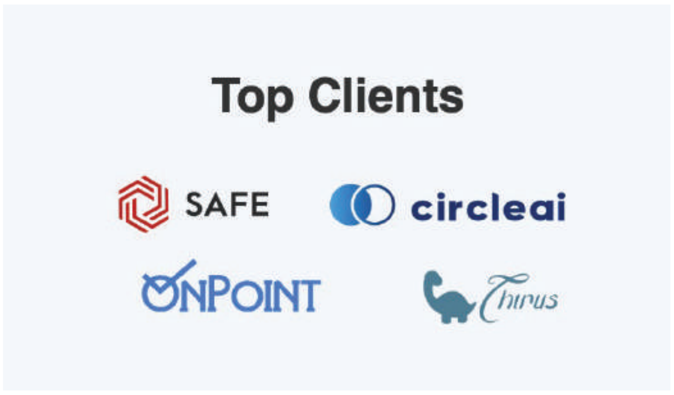
Tailwind Class	CSS Property & Value	Explanation
`flex-wrap`	`flex-wrap: wrap;`	Items are wrapped into the next line if needed
`flex-nowrap`	`flex-wrap: nowrap;`	Items are not wrapped even if it causes overflow
`flex-wrap-reverse`	`flex-wrap: wrap-reverse;`	Items are wrapped in the reverse direction

Example #3-B

Logos Wrapped

Let's say you need 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.

```html
<div class="logos flex justify-around flex-wrap">
  <img ... >
  <img ... >
  <img ... >
  <img ... >
</div>
```

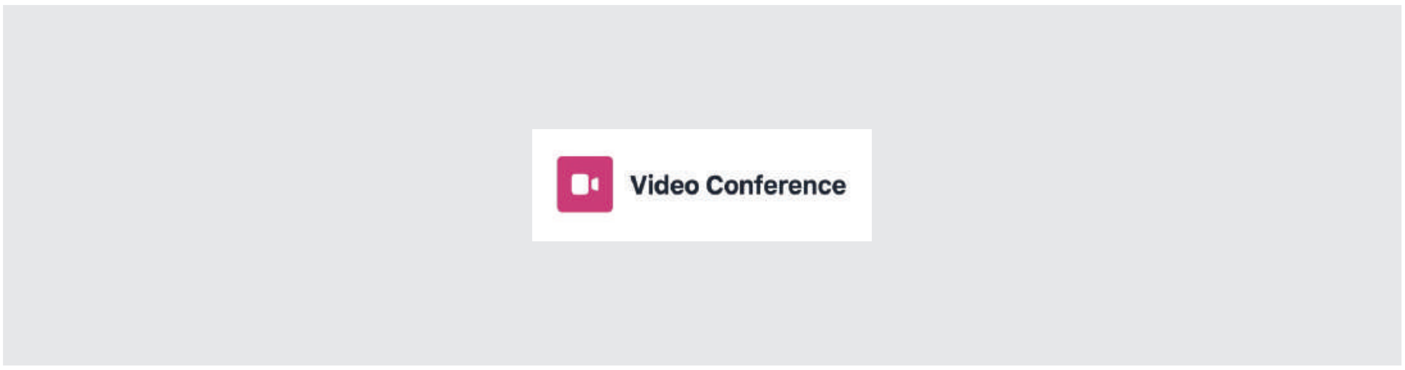
Demo Here

4. Align Items

Example #4-A

Icon and Text

Let's look at another simple use-case of flexbox. An icon and text placed next to each other vertically centered.



You can try adding align-middle class for the .icon . But that's not sufficient. You will need to add align-middle to the .icon-text too. While you might be okay with this adjustment, this is better done with flex.

Instead of the align-middle utilities, add these two classes to the .icon-wrap element.

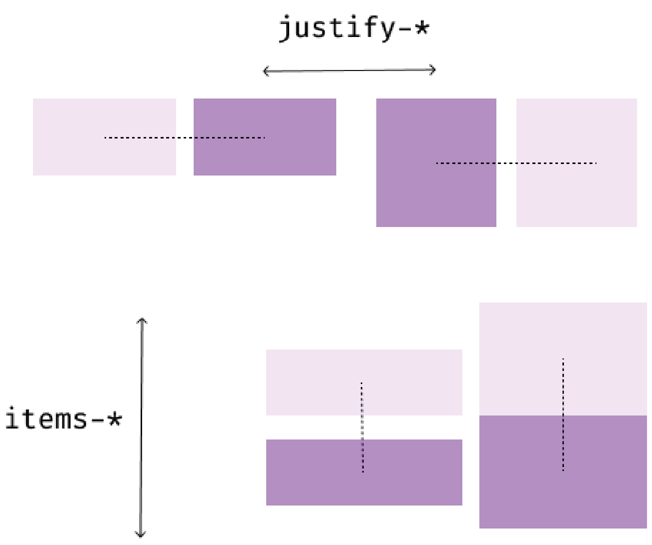
```
<div class="icon-wrap flex items-center">
  <span> ... </span>
  <span> ... </span>
</div>
```

Demo Here

Apart from flex , we added just one more utility - items-center . Let's learn more about this.

Understanding Align Items:

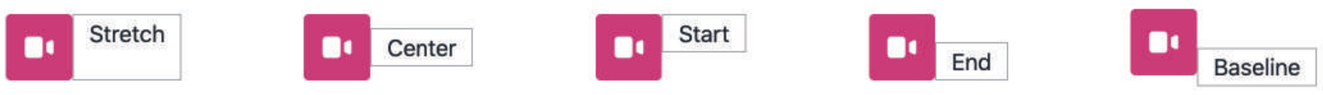
The justify-* utilities allow us to control the spacing and alignment of the flex items in the direction they are placed (Horizontally in all our previous examples). While items-* utilities allow us to control the alignment in its perpendicular direction. This illustration might give you a better idea:



In case of all our above examples, justify-* can be used to space out the items horizontally, and items-* can be used to align items vertically. This is useful especialy when the height of each item is different. items-center is one of the available utilties we just used. Some more utilities are mentioned below:

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)*

You can see the difference between these values below:



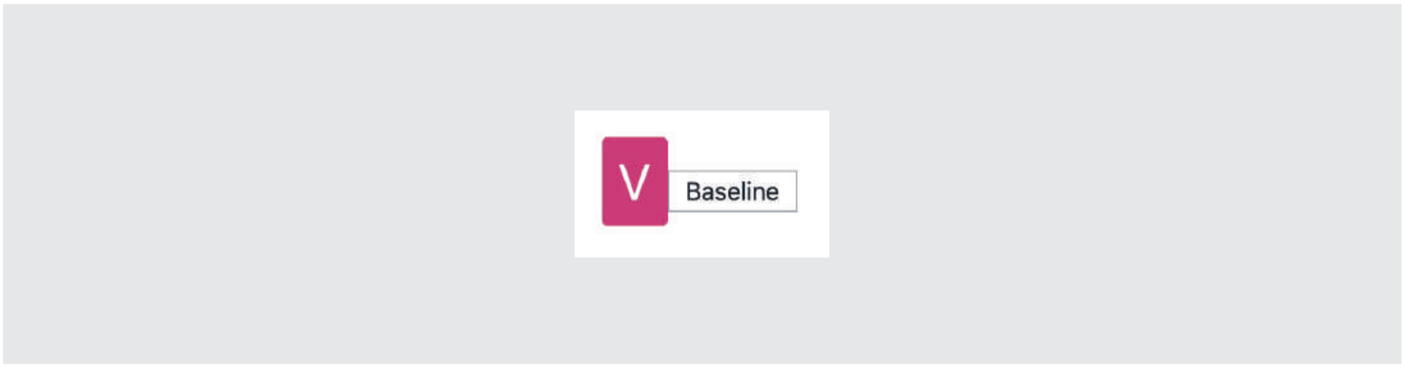
Demo Here

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

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

to

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



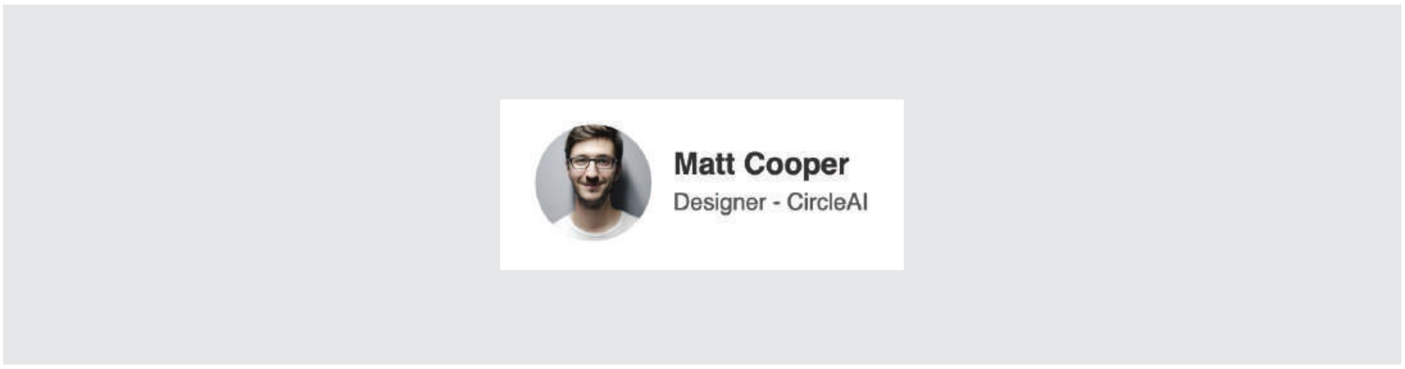
Now you can notice that the base of V is aligned with the base of the word "Baseline", almost like both of them are placed on an invisible line.

The most used utilities are items-stretch and items-center . So let's look at more of those examples.

Example #4-B

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:



Demo Here

Example #4-C

Services Section

When we need to list services as in the below screenshot, the text for one service may occupy 2 lines and for another it may occupy 1 or 3 lines. But we don't want to set a fixed height to keep all the boxes the same height. This is the best use case for the default behaviour of flex items which can also be applied using items-stretch utility class.

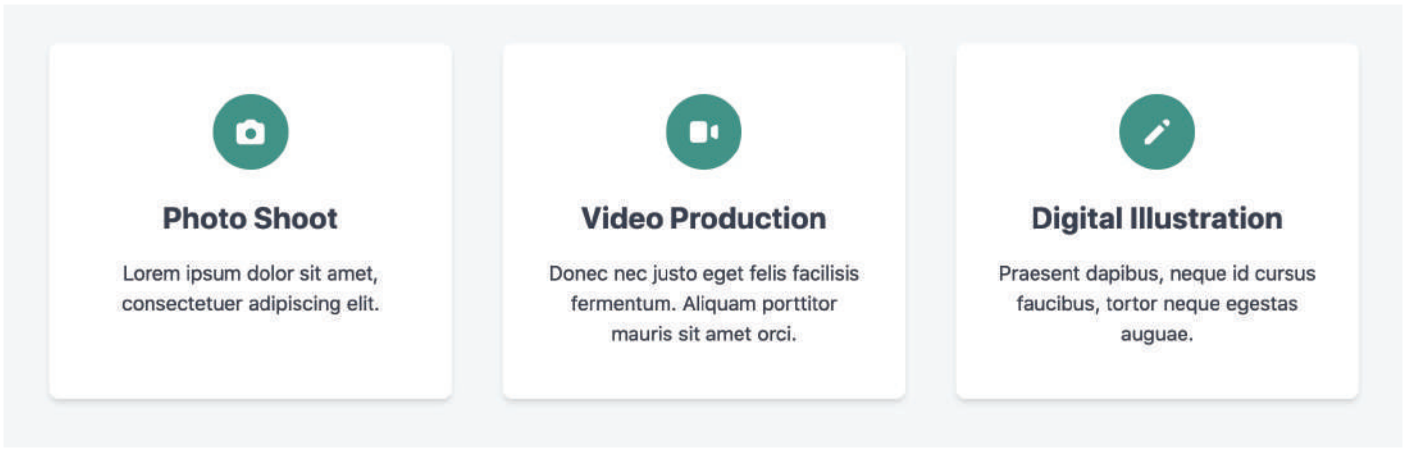


Photo Shoot
Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Video Production
Donec nec justo eget felis facilisis fermentum. Aliquam porttitor mauris sit amet orci.
Digital Illustration
Praesent dapibus, neque id cursus faucibus, tortor neque egestas auguae.

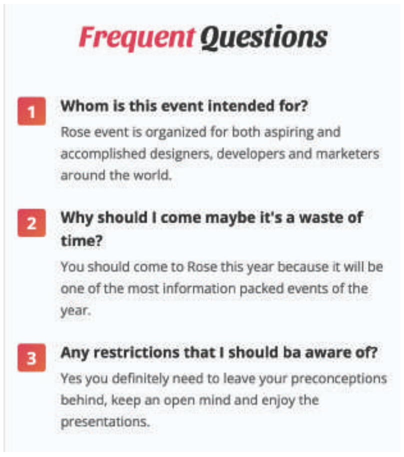
In the above link, you can remove the items-stretch class because its the default. To understand the difference better, change the class to items-end.

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

Example #4-D

Frequent Questions

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



Frequent Questions

1 **Whom is this event intended for?**
Rose event is organized for both aspiring and accomplished designers, developers and marketers around the world.

2 **Why should I come maybe it's a waste of time?**
You should come to Rose this year because it will be one of the most information packed events of the year.

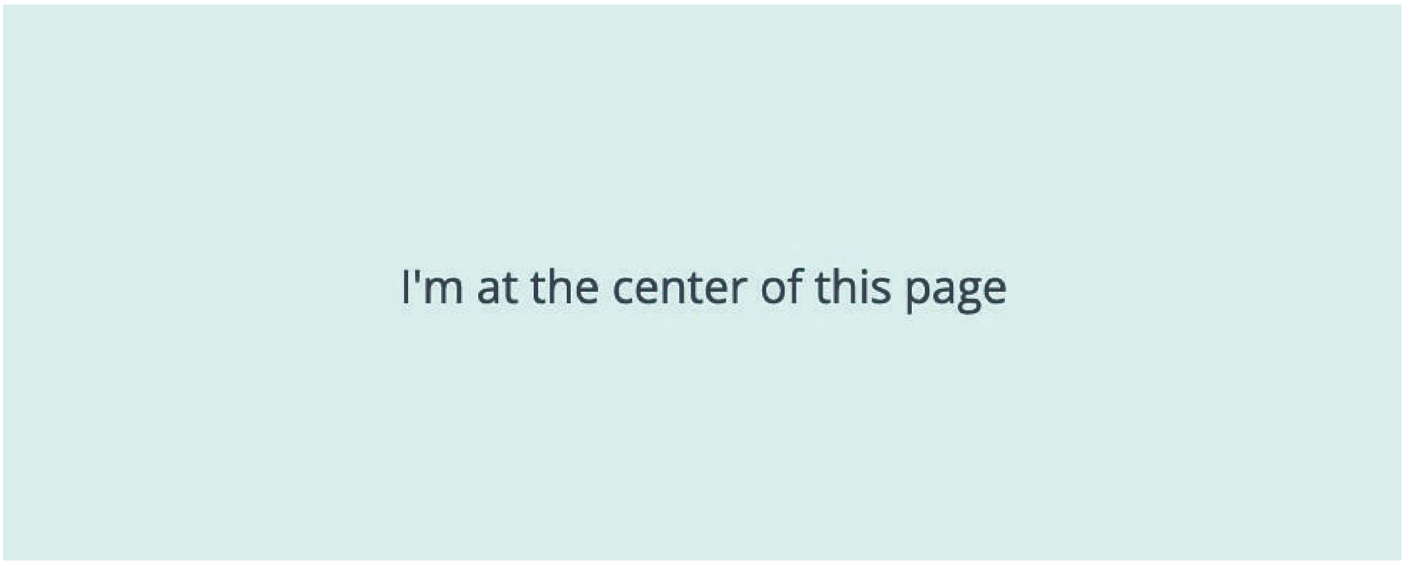
3 **Any restrictions that I should ba aware of?**
Yes you definitely need to leave your preconceptions behind, keep an open mind and enjoy the presentations.

Demo Here

Example #4-E

Center a div

This is something you will always encounter. You want to center a div or any element within its parent, but there's no straightforward way to center it both horizontally and vertically. With flexbox, using the justify-* and items-* utilities, it's super easy.



Solution # 1

We have a container occupying full screen using w-full and h-screen . Within this, is one .item div that we wish to center within the container. Adding flex utility to the parent makes it a flex container and the .item becomes a flex child. In all the previous examples, we always used more than one flex item. But in this example we need only one.

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

Adding justify-center and items-center positions the child item at the center of the page horizontally and vertically.

Demo Here

Try changing the width and height of the parent div in the link above to see how the .item still remains at the center of the container.

Solution # 2

There's another way you could achieve the same result

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

Here, 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 . You can use any of the two methods above that suits you.

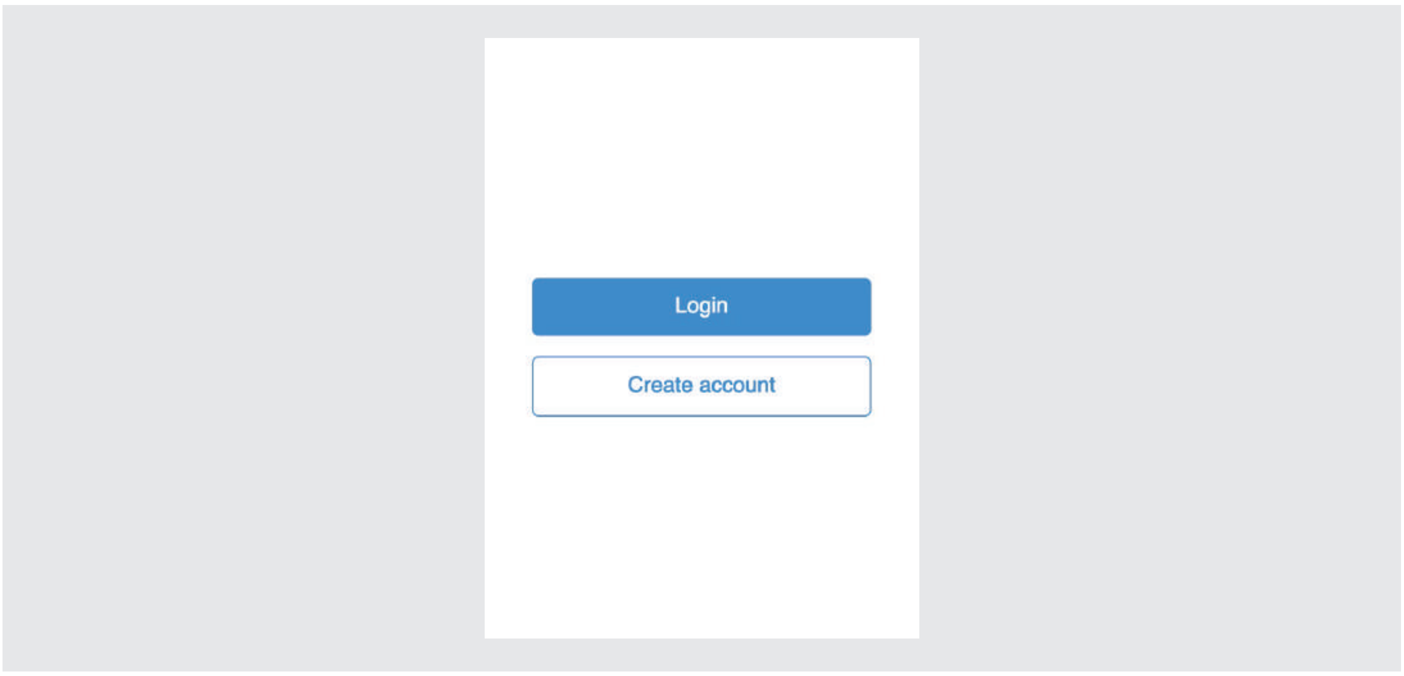
👉 Demo Here

5. Flex Direction

Example #5-A

Welcome Screen

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



Possible Solution:

There are multiple approaches to this. In case you used the concepts your learned so far, you might have tried adding an additional div within wrapper and added flex and items-center to the wrapper. Additionally adding block utilities to the links.

```html
<div class="wrapper flex items-center">
  <div class="w-full">
    <a href="#" class="block link login-link">Login</a>
    <a href="#" class="block link signup-link">Create account</a>
  </div>
</div>
```

While the above solution works, it's a long one! There's a better approach. You can instead try this:

Better Solution:

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

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

👉 Demo Here

Understanding Flex Direction:

The first thing we learnt here was that adding flex utility makes all the child elements get laid out in one direction. By default they all get placed in a single row. To change that row direction to a column instead, we can use the flex-col utility along with flex .

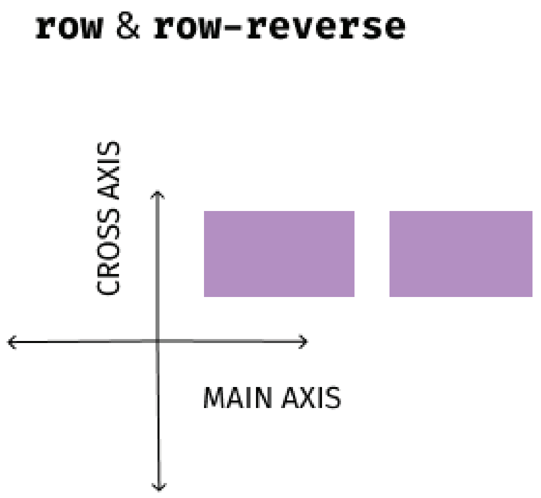
Tailwind Class	CSS Property & Value	Explanation
`flex-row`	`flex-direction: row;`	This is the default behaviour. All items are placed in a single row from left to right
`flex-col`	`flex-direction: column;`	All items are placed in a single column from top to bottom
`flex-row-reverse`	`flex-direction: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

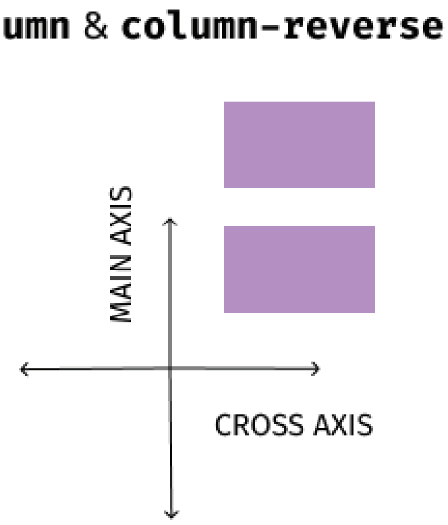
At first, it appears that using flex-col with flex is same as the normal flow of the web page. Without flexbox, this is how the elements are placed anyway. Then why do we need this? Like we just saw in Example 5a above, this is the best way to vertically align those two buttons in the center of the container. There are few more use cases of flex col that we will explore further.

Before that, I want you to notice one thing. In the above example, we used justify-center to center the items vertically. In Example 4a and Example 4b , we used items-center to center the items vertically! Now why is that?

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.



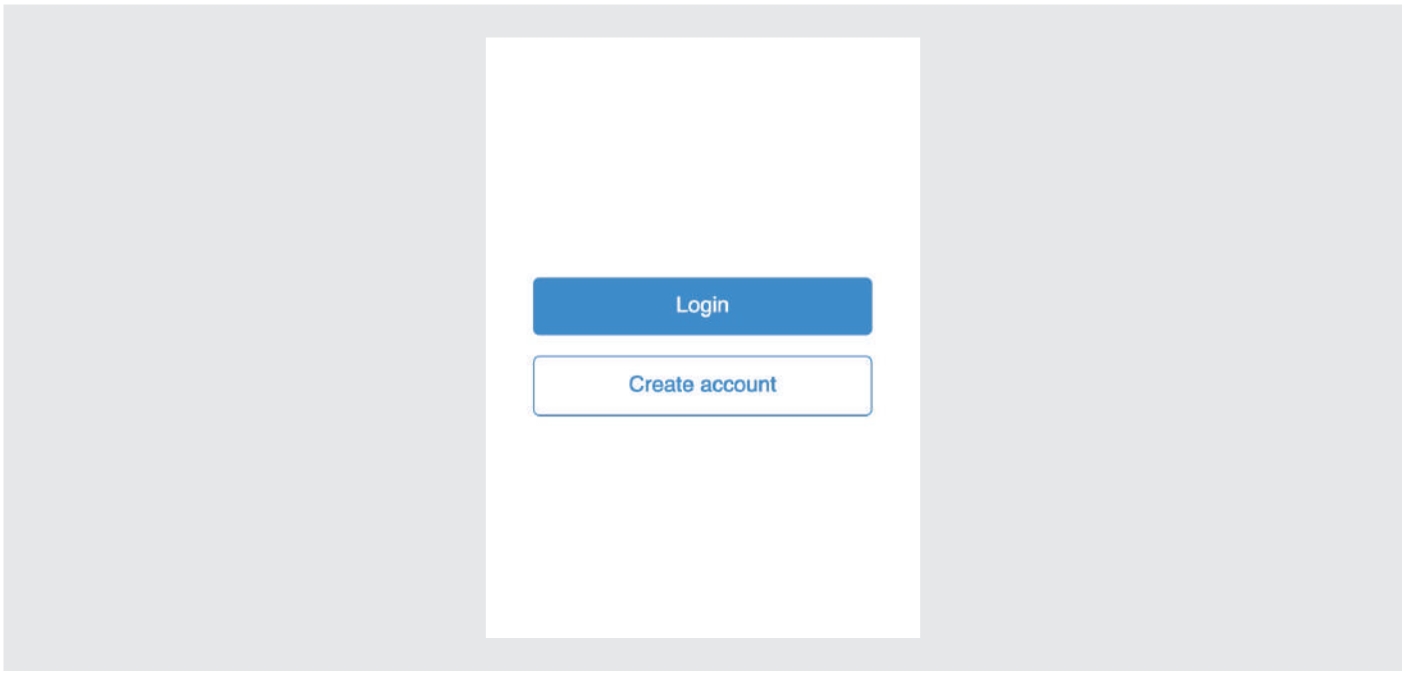


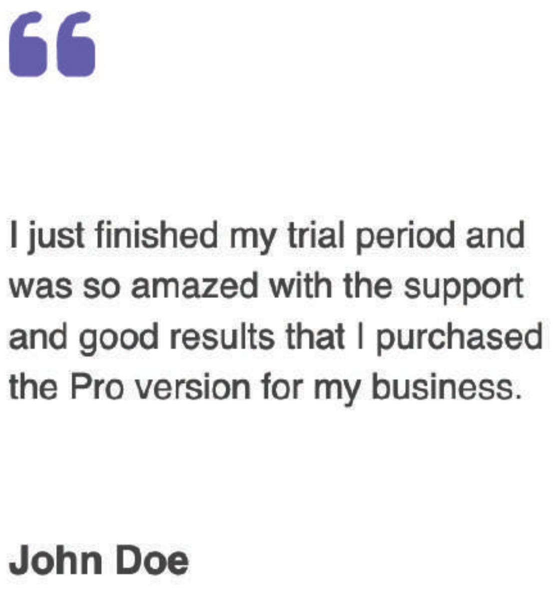
The justify-* utilities control spacing and alignment along the main axis, while the items-* utilities control alignment along the **cross axis.** In Example 5a, our flex direction is column . So vertically centering needs alignment along its main axis. That's why we need to use justify-center .

Example #5-B

Testimonial Card

Assume you have a testimonial card with fixed height. Within the card, there's a quote icon at the top, customer name at the bottom and the testimonial text at the middle. The testimonial text can vary in length, but needs to be equally spaced from the icon and the name.





I just finished my trial period and was so amazed with the support and good results that I purchased the Pro version for my business.

John Doe

Now use the flex-col class along with few more necessary classes and see if you can get the desired result.

Solution:

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

```html
<div class="card flex flex-col justify-between items-start">
  <img  ... >
  <p> ... </p>
  <span> ... </span>
</div>
```

Demo Here

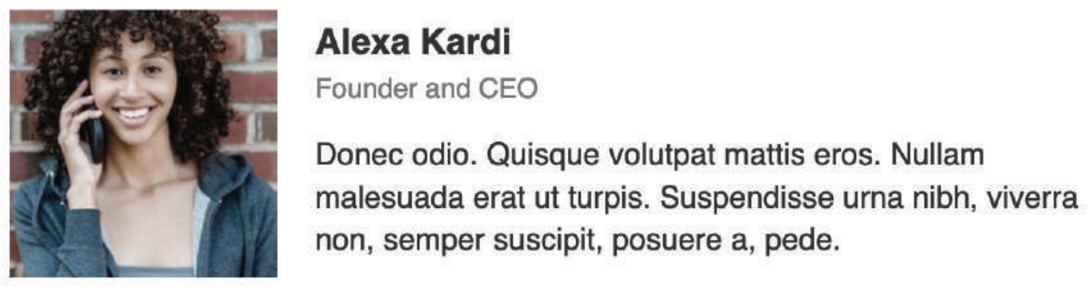
By default, the flex items stretch along the cross axis. So without the items-start class,the icon image stretches full width because that's the cross axis when column direction is used. The utility class justify-between is what makes the child items space out vertically as required.

Try adding more lines to the testimonial text or removing some lines. You will notice that the text still remains equally spaced from the icon and name, as required.

Example #5-C

Alternating List of Profiles

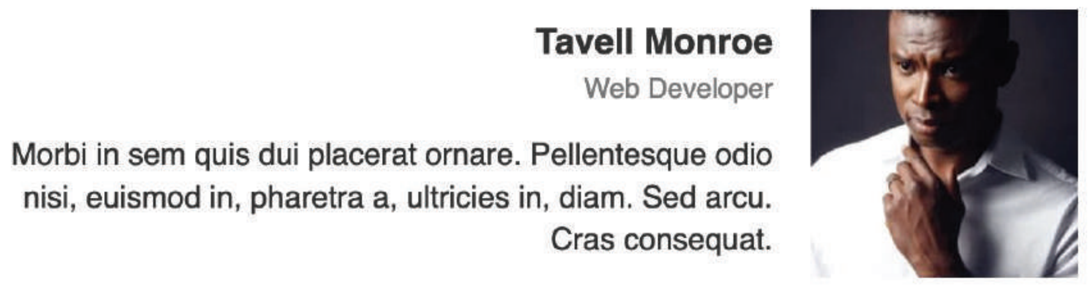
Let's say you have to list some profiles on your page. To break the monotony, you'd like to alternate the photos and text like this.



Alexa Kardi

Founder and CEO

Donec odio. Quisque volutpat mattis eros. Nullam malesuada erat ut turpis. Suspendisse urna nibh, viverra non, semper suscipit, posuere a, pede.



Tavell Monroe

Web Developer

Morbi in sem quis dui placerat ornare. Pellentesque odio nisi, euismod in, pharetra a, ultricies in, diam. Sed arcu. Cras consequat.

One way is to directly change the order in HTML.

```
<div class="profile">
  <img ... >
  <div> ... </div>
</div>
<div class="profile">
  <!-- Reverse the order -->
  <div> ... </div>
  <img ... >
</div>
```

But if you have a long list and suddenly you wish to insert another profile somewhere inbetween, you will have to again reverse the order in the markup for all the profiles that appear after that.

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

Solution:

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

Demo Here

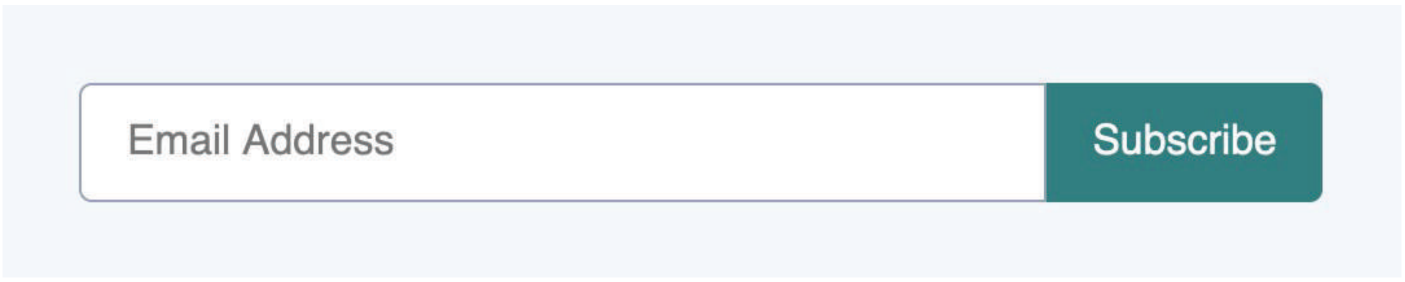
The even: prefix helps apply the row-reverse direction only the the even child elements. This solution is helpful when you are using frameworks like Vue, React or Laravel where you have the profiles data stored in objects and you display them using loops.

6. Flex Grow

Example #6-A

Inline Subscribe Form

Here is a subscribe form with a text input and a button displayed in a single row. So flexbox is the best solution, but how do you make the text input occupy all the available horizontal space of its parent container?



Try out and see how you can make the text input occupy the entire space available while the **Subscribe** button takes up only as much space as needed. (Don't take too long trying because we have a very simple utility class for this)

```
<div class="container flex">
  <input  ... >
  <button> ... </button>
</div>
```

We already have flex applied on .container . Now we need to add one class to the input element.

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

Demo Here

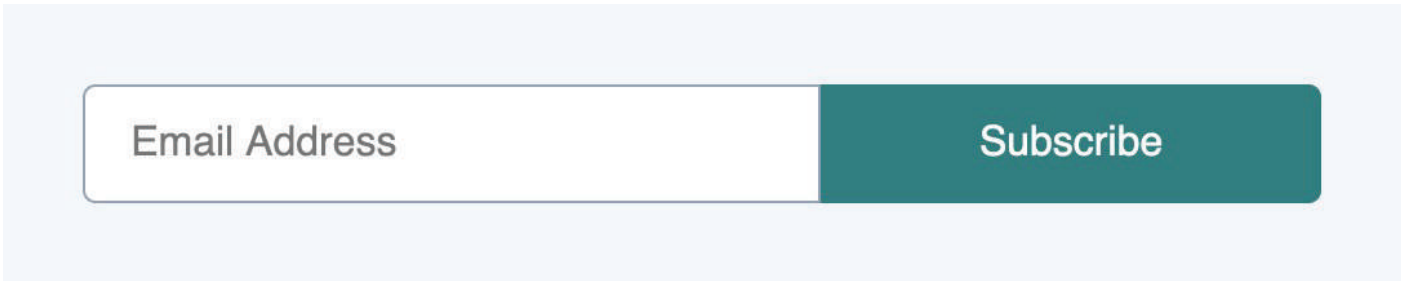
This just works! Resize the browser and it's fully responsive.

Notice that until now, we only added classes to the parent element - the flex container. The class flex-grow is the first one to be used on a child element - the flex item. Now let's learn more about these utilities.

Understanding Flex Grow

The default behaviour of a flex item is to occupy only as much space as needed by the content within. It doesn't "grow", because the default value of the CSS flex-grow property is 0. 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).

In the previous example, we added the rule flex-grow for the text input. This made the input field occupy all the left over width in the parent. What if we add the same rule to the button as well? Try for yourself in the same demo link above.



Notice how the button also tries to occupy some of the left over horizontal space. Also notice that we added a grow factor of 1 to both the items, but they don't have equal widths. This is where it's easy to get confused. Read the next part carefully.

When flex-grow is added to two flex items, the left over space is divided into two parts and added to the **initial** widths of those two items. Since the text input's initial width was more than that of the button, it occupies more space. 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

But if we want to set a higher flex-grow value, we can add it in the config file or use arbitrary values or add custom styles. What if we add flex-grow: 2 to the text input, but flex-grow: 1 to the button? This time, the left over space is divided into three equal parts. Two parts width is added to the text input and one part width to the button

If this is too confusing, just don't worry. In the next few examples, all of this will become clear. For now, just remember:

1. flex-grow is a flex item's utility (and not of flex container)
2. It can take any value greater than or equal to 0.
3. The default value is 0, hence the flex item does not grow by default

Example #6-B

Sticky Footer

Ever faced a situation where your main content is too small, making your footer appear somewhere in the middle of the page instead of at the bottom? The easiest solution to this is using flexbox for the whole layout, with column direction and adding flex-grow to the main content.

Main Content
Footer

```
<div class="container">
  <div class="main"> ... </div>
  <footer> ... </footer>
</div>
```

We need to first add a min-h-screen to the .container . Otherwise nothing will work. Then make it a flex container with flex and flex-col . At last, add flex-grow utility to the .main element.

Solution:

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

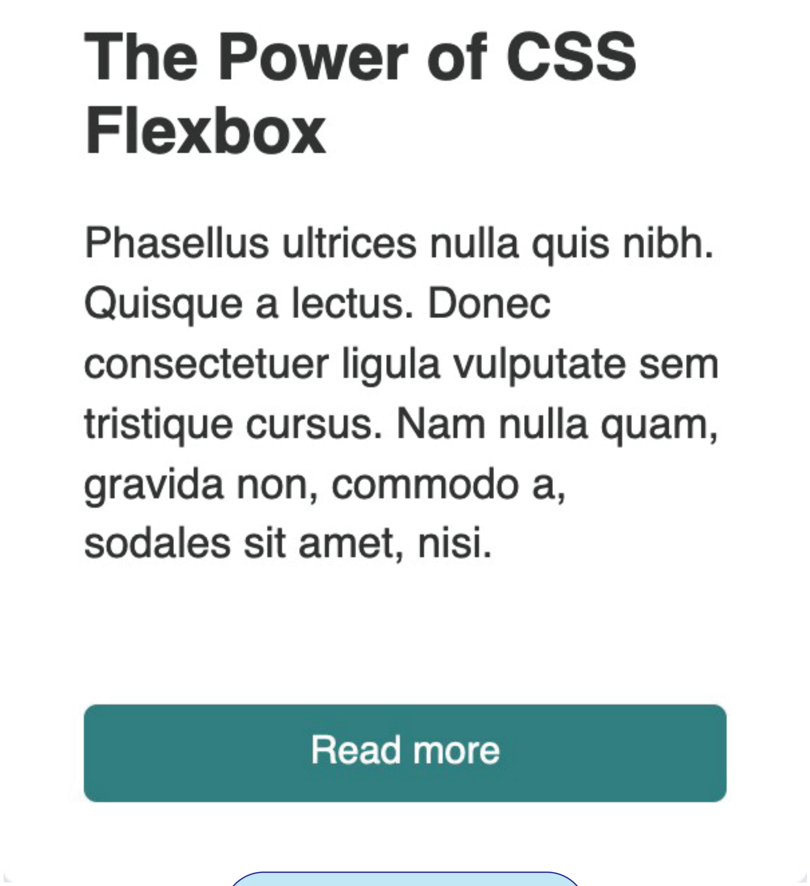
Demo Here

If the main content is long enough, the footer is at the bottom as usual. Which is why it's called a "Sticky Footer". Do check for yourself.

Example #6-C

Card with Header & Footer

This one is very similar to our previous example. 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 Power of CSS Flexbox

Phasellus ultrices nulla quis nibh. Quisque a lectus. Donec consectetuer ligula vulputate sem tristique cursus. Nam nulla quam, gravida non, commodo a, sodales sit amet, nisi.

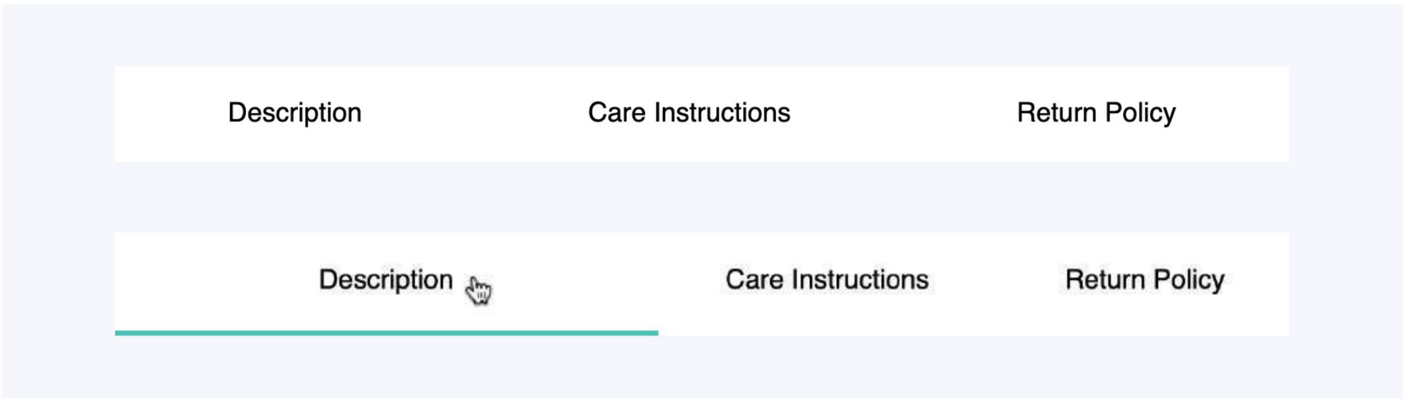
Read more

Demo Here

Example #6-D

Tabs Hover Effect

Here's an example of tabs that expand on hover. Each tab has a variable width depending on the text. Once hovered, the active tab expands while the other two shrink.



Can you try and achieve this by changing the flex-grow value on hover (using arbitrary styles with [])?

```html
<ul>
  <li> ... </li>
  <li> ... </li>
  <li> ... </li>
</ul>
```

Solution:

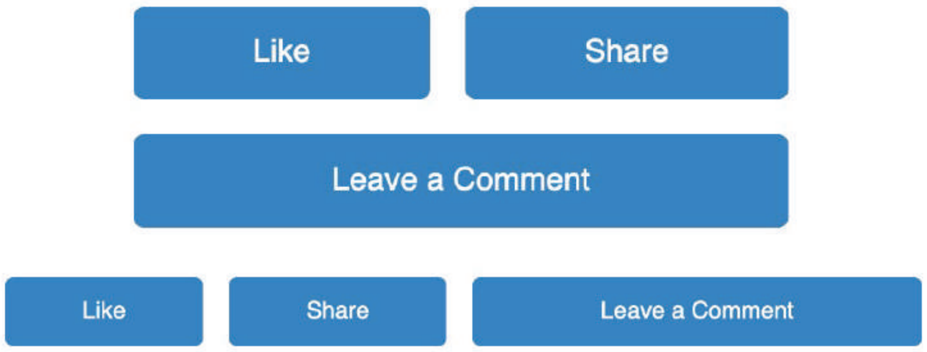
Initially, all the tabs are set to **flex-grow: 1** and on hover, we increase the value of flex-grow to any number depending on how wide you want the active tab to be.

```html
<ul class="flex">
  <li class="flex-grow hover:flex-grow-[3]"> ... </li>
  <li class="flex-grow hover:flex-grow-[3]"> ... </li>
  <li class="flex-grow hover:flex-grow-[3]"> ... </li>
</ul>
```

Example #6-E

Variable Width Responsive Buttons

Consider this example where you have three buttons below a blog post - "Like", "Share" and "Leave a Comment". You want them to occupy full width of the container and also want to give importance to the last button by giving it a larger width compared to the other two buttons.



And yes, 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

```
<div class="container">
  <button type="button"> ... </button>
  <button type="button"> ... </button>
  <button type="button"> ... </button>
</div>
```

Solution:

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.

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

Demo Here

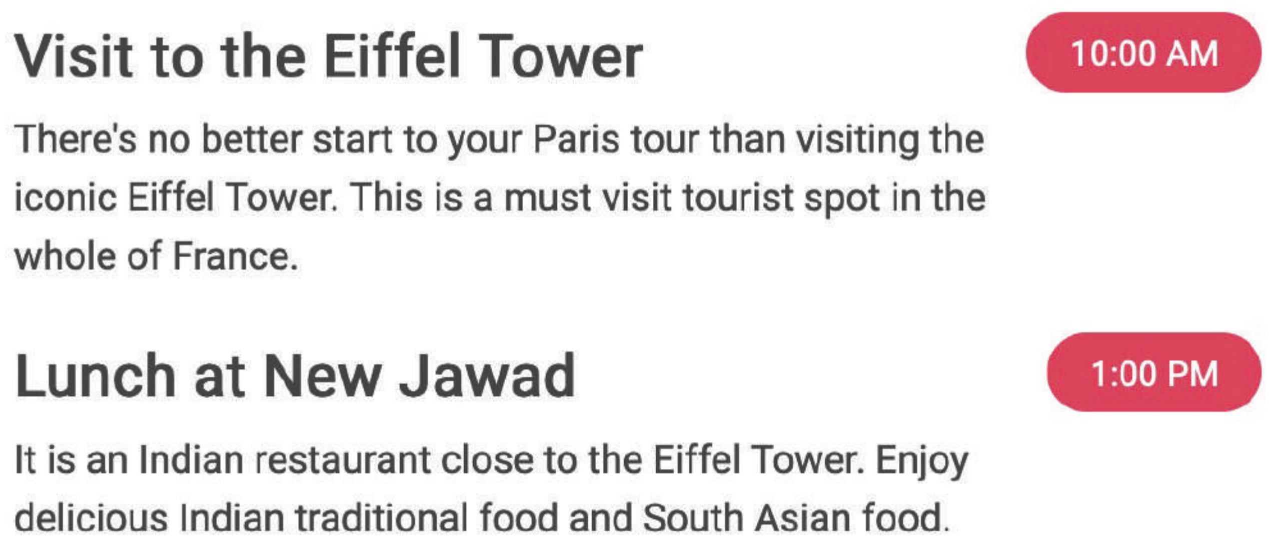
We will look at more examples that use flex-grow once we learn a couple more properties

7. Flex Shrink

Example #7-A

Itinerary

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



Visit to the Eiffel Tower

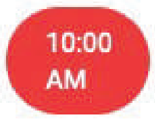
10:00 AM

There's no better start to your Paris tour than visiting the iconic Eiffel Tower. This is a must visit tourist spot in the whole of France.

Lunch at New Jawad

1:00 PM

It is an Indian restaurant close to the Eiffel Tower. Enjoy delicious Indian traditional food and South Asian food.

It looks simple and easy to achieve using flexbox, but because flexbox decides the width of each child item based on the content within, the time element gets a very little space making it appear in two lines like this 

A HTML solution to this is to wrap the time in <nobr> tags. But can you find a CSS solution to this problem?

Let's see how to get this working using another set of **flex item's** utility classes **flex-shrink.**

```html
<div class="container">
  <div> ... </div>
  <span>10:00 AM</span>
</div>
```

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

This will prevent the time span from shrinking

👉 Demo Here

You might immediately see that flex-shrink is somewhat opposite to flex-grow . Let's learn about these utilities in detail.

Understanding Flex Shrink

The default behaviour 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). Hence each item shrinks proportionate to its initial size. You don't have to get into the exact calculations. A larger element shrinks more than the smaller one by default. This is because, the default value of flex-shrink for each flex item is **1**.

In our previous example, we changed this value to **0** using the utility class flex-shrink-0 , hence preventing the item from shrinking. The other item shrinks to fit. 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

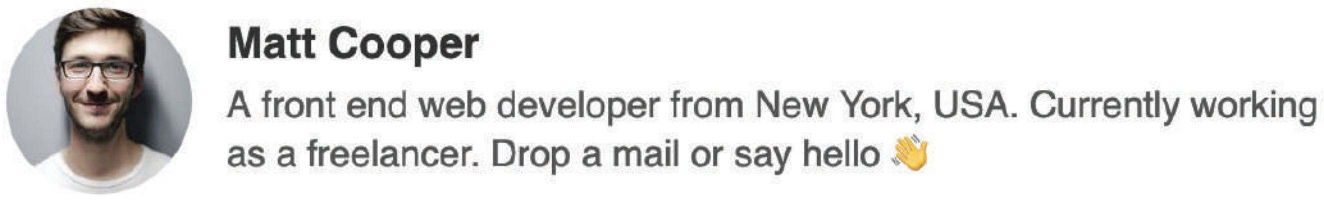
You will mostly never use a shrink factor other than 0 or 1. You would either want the element to shrink or not. So, you only need to remember these:

1. flex-shrink is a flex item's property
2. It can take any value greater than or equal to 0.
3. The default value is 1, hence the flex item shrinks by default regardless of the specified width .

Example #7-B

Profile Card - Large

We saw a small profile card in Example 4b. Since that's small, it's responsive as it is. But 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 👋

Can you make the image not shrink?

Yes, one solution is to change the width to min-width . It works for this example, but sometimes we might not know the exact width. Hence its best to use flex-shrink .

```
<div class="profile">
  <img ... >
  <div> ... </div>
</div>
```

Solution:

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

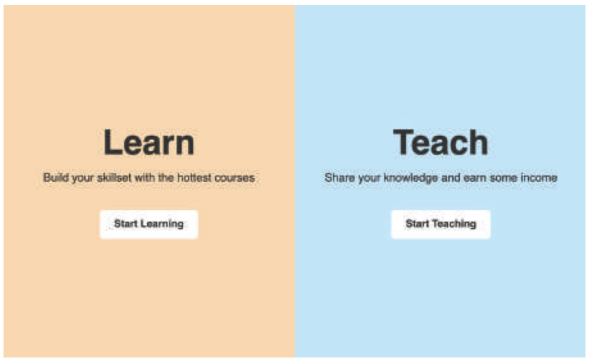
👉 Demo Here

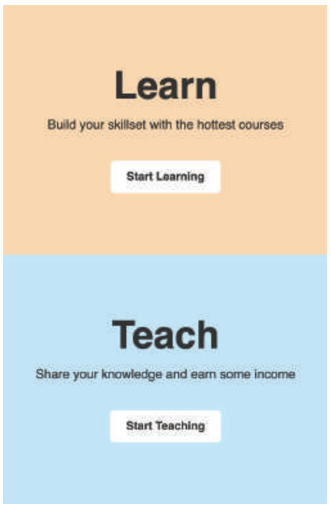
8. Flex Basis

Example #8-A

Split Screen Display

Here's a simplified example of a landing page with a split screen display occupying full screen. On large screens, the page is split up horizontally and on smaller screens, it's split up vertically


Learn
Build your skillset with the hottest courses
Start Learning
Teach
Share your knowledge and earn some income
Start Teaching


Learn
Build your skillset with the hottest courses
Start Learning
Teach
Share your knowledge and earn some income
Start Teaching

With the concepts you've learned so far, I'm sure you can make this work. Since Tailwind CSS uses mobile first approach, you need to first use flex-col to split the screen vertically for small screens along with height utilities for half height. For wider screens, you can use the md: prefix and change to flex-row and using width utilities, split the screen horizontally.

```
<div class="container">
  <div class="split"> ... </div>
  <div class="split"> ... </div>
</div>
```

The .container element should be full screen. That's done with w-full and h-screen . Also, the container's flex direction is set to column on smaller screens and row on large screens. So this is achieved with flex flex-col md:flex-row .

Then, you can add a h-1/2 to the flex items on small screens. And on large screens, w-1/2 and change the height back to h-full .

Possible Solution 1

Here's a simplified example of a landing page with a split screen display occupying full screen. On large screens, the page is split up horizontally and on smaller screens, it's split up vertically

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

Or, you might can add a flex-grow to both the flex items.

Possible Solution 2

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

Both the solutions work for this example. Solution 1 is long but works all the time. Solution 2 is short but might not work for different cases (Example, one split screen contains a large image).

Better Solution

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

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

Demo Here

So flex-basis property for a flex item is similar to width for flex-row direction and similar to height for flex-col direction.

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 . Like we already saw in the previous example, 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.

When you set the flex-basis of an item to 100px for example, the item first occupies 100px . And then,

1. If there's more space available AND flex-grow is greater than 0, the item grows to occupy more than 100px
OR
2. If there's less space AND flex-shrink is greater than 0, the item shrinks to occupy lesser than 100px.

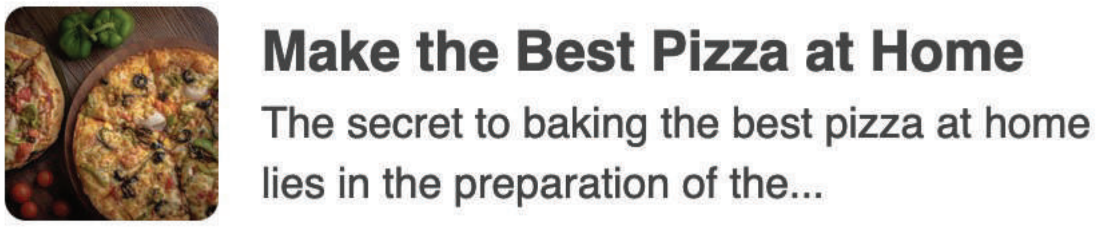
By default, the value of flex-basis is auto , which means the size is auto-calculated based on the width or height utilities.
Once the basis-* utilities are added to the upcoming version in Tailwind CSS, you can use it similar to the width and height utilities. Some of the example values are here:

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

Example #8-B

Blog Post Display

Here's a blog post display example very similar to Example 7b of a large profile card. It has an image on the left and long text on the right.


Make the Best Pizza at Home

The secret to baking the best pizza at home lies in the preparation of the...

Can you make the image not shrink?

Yes, one solution is to change the width to min-width . It works for this example, but sometimes we might not know the exact width. Hence its best to use flex-shrink .

```
<div class="container">
  <div>
    <img  ... >
  </div>
  <div> ... </div>
</div>
```

Solution:

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

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

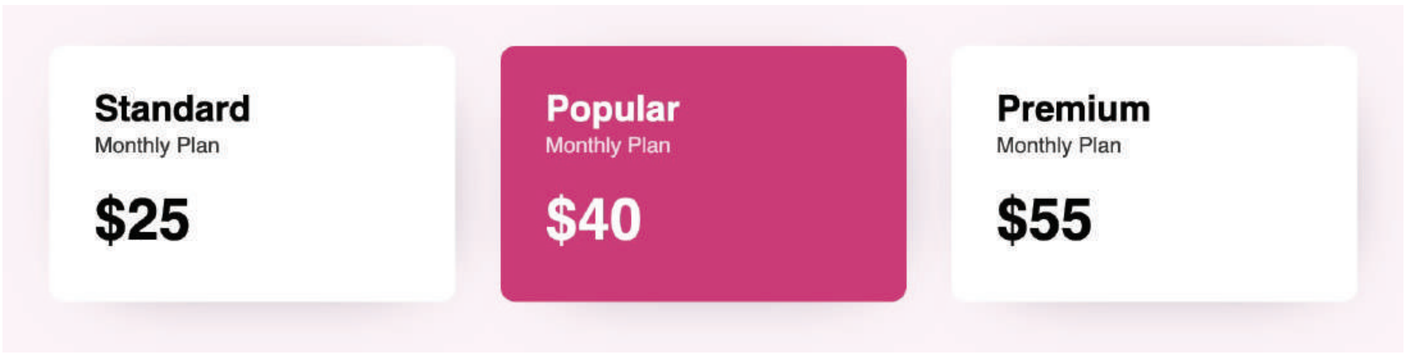
```
@layer utilities {
  .basis-20 {
    flex-basis: 5rem;
  }
}
```

Demo Here

Example #8-C

Pricing Plans

Three equally sized blocks with margins in between is a very common pattern. With all the concepts we just learnt, this example doesn't look very hard now.



```html
<div class="container">
  <div class="plan"> ... </div>
  <div class="plan"> ... </div>
  <div class="plan"> ... </div>
</div>
```

Possible Solution

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

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

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

```css
@layer utilities {
  .basis-1\/3 {
    flex-basis: 33.333333%;
  }
}
```

Demo Here

Note that though we used basis-1/3 , the final width of each column is less than 33% of the parent because of the margins between the flex items. Each item shrinks by default to fit into the container.

Now there's a better solution to this:

Better Solution

```
<div class="container">
  <div class="plan mx-4 basis-0 flex-grow flex-shrink"> ... </div>
  <div class="plan mx-4 basis-0 flex-grow flex-shrink"> ... </div>
  <div class="plan mx-4 basis-0 flex-grow flex-shrink"> ... </div>
</div>
```

Demo Here

This is better because if you add four blocks or two blocks instead of three, you don't have to change the basis-* value. Try removing one of the plans or adding another. They all take up equal space.

Also, in the previous solution, we don't have flex-shrink and flex-grow specified. They have their default values. It is always encouraged to set all 3 properties to avoid any kind of confusion. Very soon we will see how all these three utilities can be combined into just one.

Spaces between the blocks

One thing to note is that we have added margin to each item to create margins in between and around the blocks. This creates some margins around the blocks too, and not just between them. The best solution hence is to use the gap utilities on the flex container. But the gap CSS property doesn't have a good browser support for flexbox yet, at the time of writing this. I encourage you to check for browser support and use it accordingly. I will talk more about this property in the Grid section of this book.

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. In the previous example, you can replace all those three CSS classes with a single class.

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

Try replacing those 3 classes with just flex-1 in the previous example and notice that everything works the same.

Understanding Flex

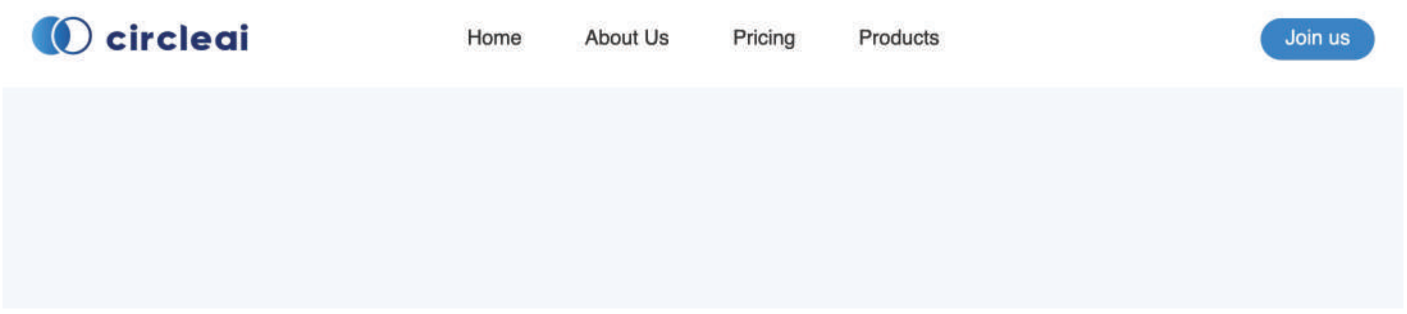
The flex-* utility classes control how flex items both grow and shrink along with specifying an initial size. In Tailwind CSS, we have four of these utility classes that cover most of the use cases.

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.

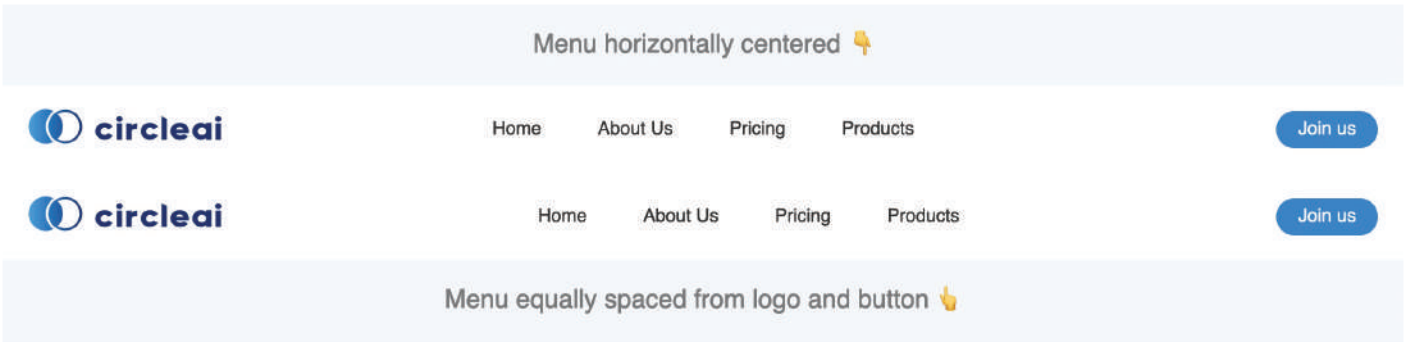
Example #9-A

Navigation Bar with Centered Menu

We often come across navigation bars with a logo on the left, one or two buttons on the right and multiple menu links absolutely centered horizontally. Though it looks simple, it's not straightforward to implement.



Notice how the menu is at the exact center of the entire navbar. The distance from menu to logo and menu to button are not equal. Hence using justify-between is not sufficient to achieve this. Look at the difference



```html
<header>
  <a>
    <img ... >
  </a>
  <ul> ... </ul>
  <span>
    <button> ... </button>
  </span>
</header>
```

Solution

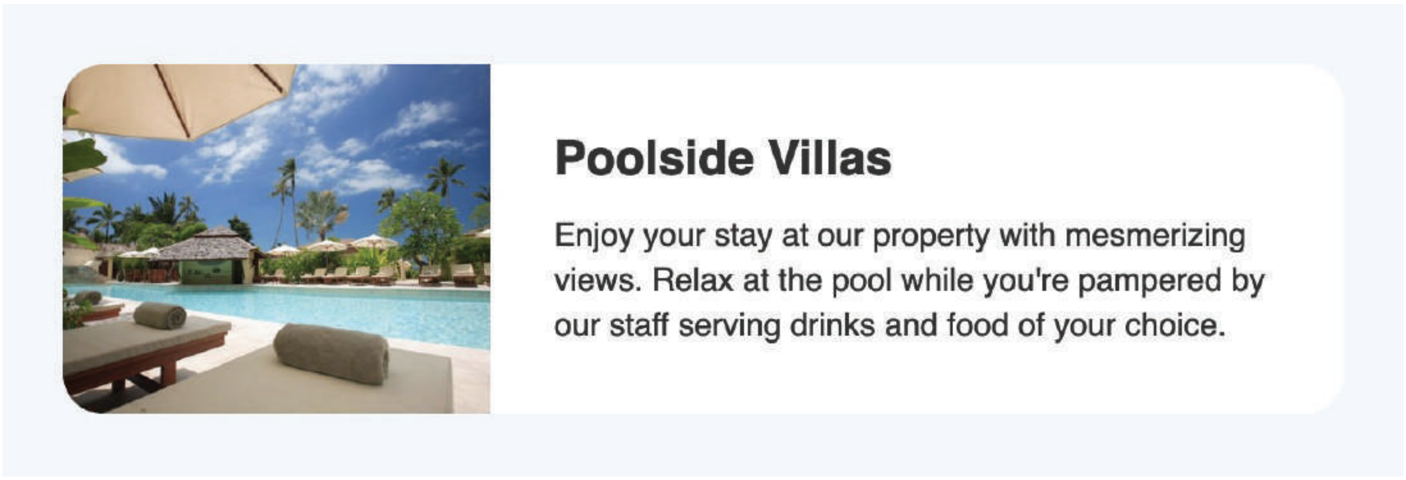
```html
<header class="flex justify-between items-center">
  <a class="flex-1">
    <img ... >
  </a>
  <ul> ... </ul>
  <span class="flex-1 text-right">
    <button> ... </button>
  </span>
</header>
```

👉 Demo Here

Example #9-B

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. The text block is twice the width of the image and the component is flexible.



Poolside Villas

Enjoy your stay at our property with mesmerizing views. Relax at the pool while you're pampered by our staff serving drinks and food of your choice.

See if you can get this working using flex-* utility with an arbitrary value.

```
<div class="container">
  <img ... >
  <div class="details"> ... </div>
</div>
```

Solution

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

For the img , we have flex-1 which is flex: 1 or flex: 1 1 0% .

For the div , we use flex-[2] which translates to flex: 2 or flex: 2 1 0% and hence the div occupies twice the width of the image.

Along with flex-1 for the img , we also need **w-full** and object-cover to fit the image in the set space without changing the aspect ratio.

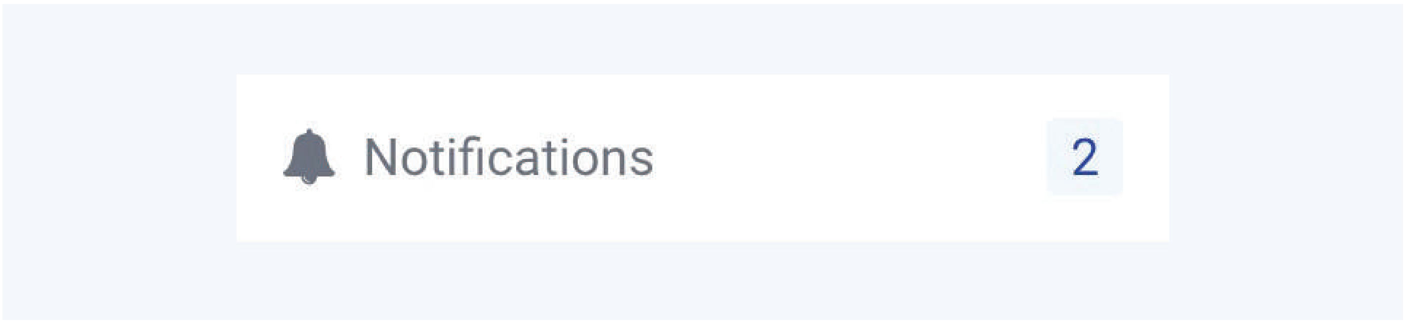
Demo Here

10. Auto Margins

Example #10-A

Notifications Menu Item

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



```html
<div class="container">
  <i> … </i>
  <span class="text">Notifications</span>
  <span class="count">2</span>
</div>
```

If you can wrap the icon and text within another div , we can achieve this look by using justify-between . But can you get the same look without editing this HTML?

One way to achieve this is by adding **flex-grow** to the .text element. Here's another solution.

Solution

```html
<div class="container flex align-center">
  <i> … </i>
  <span>Notifications</span>
  <span class="count ml-auto">2</span>
</div>
```

We have added ml-auto which is margin-left: auto to the count element to simply push it to the right.

👉 Demo Here

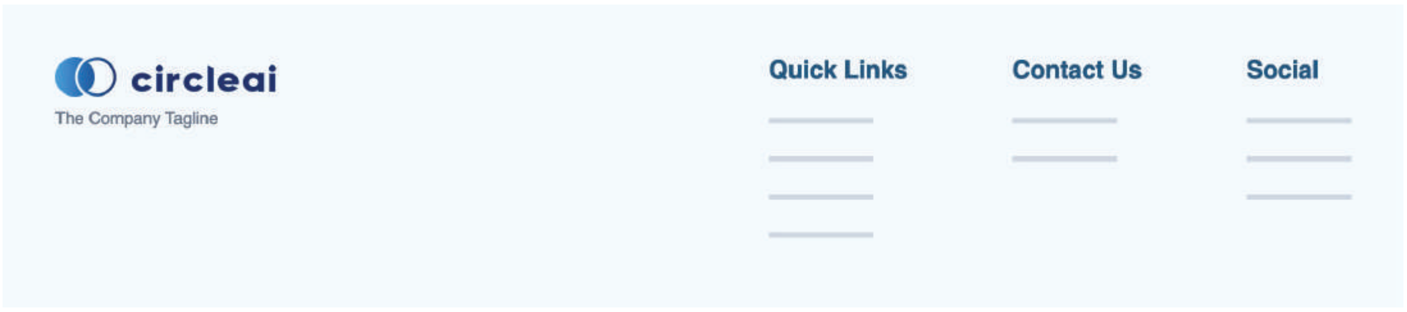
The margin utilities can be used with flex items to extend margins to occupy the extra space. So, in the above example, the left margin occupies all the extra space on the left, pushing the element to the right.

If you can recall, we used m-auto to center a single flex item within its container in Example 4e - Solution 2. This works the same way.

Example #10-B

Footer with Multiple Columns

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



Based on what you saw in the previous example, can you get this result using auto margins?

```html
<footer>
  <div class="footer-col"> ... </div>
  <div class="footer-col"> ... </div>
  <div class="footer-col"> ... </div>
  <div class="footer-col"> ... </div>
</footer>
```

Solution

We need to "push" the 2nd column to right

```html
<footer class="flex">
  <div class="footer-col"> ... </div>
  <div class="footer-col ml-auto"> ... </div>
  <div class="footer-col"> ... </div>
  <div class="footer-col"> ... </div>
</footer>
```

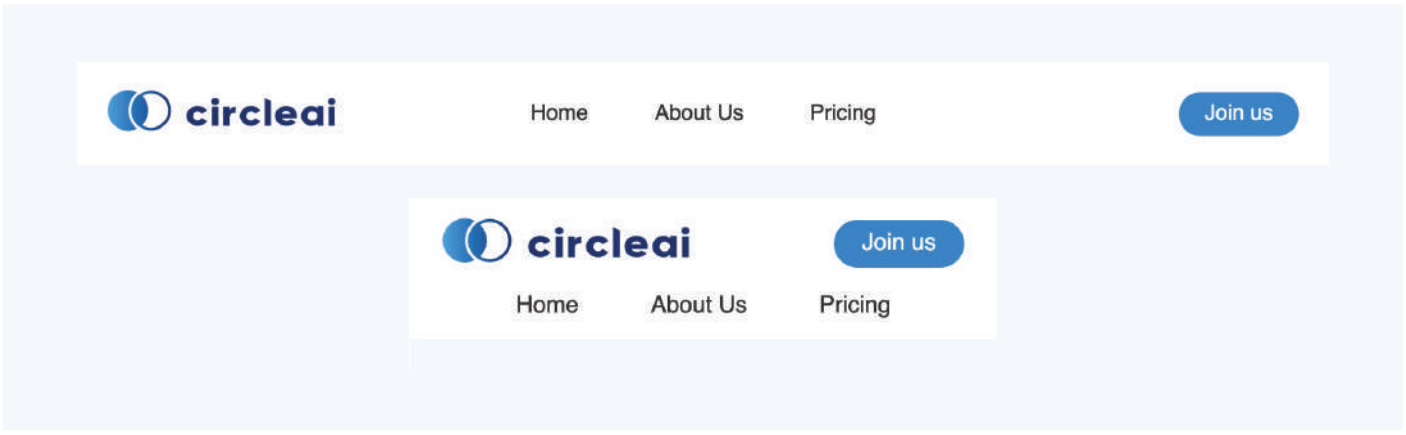
👉 Demo Here

11. Order

Example #11-A

Responsive Navigation Bar

Let's look at our Example 9a once again and make it responsive now. 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.



```html
<header>
  <a>
    <img ... >
  </a>
  <ul> ... </ul>
  <span>
    <button> ... </button>
  </span>
</header>
```

Solution

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

Demo Here

The only new utility here is the order utility

Understanding Order

The order property is also used on a flex item. 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.

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.

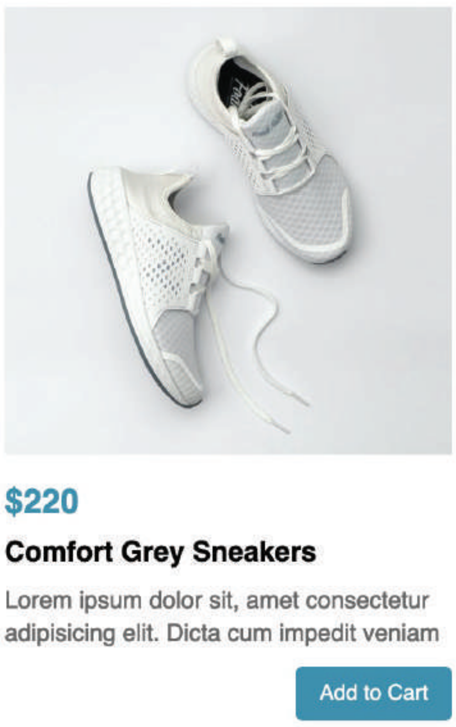
Hence when we set order-last to the ul element in the example above, only that element is removed from the normal flow and placed at the end. And for large screens, we change it back to normal flow using order-none .

12. Align Self

Example #12-A

Product Display

This is a card component using flex-col . 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.


$220

Comfort Grey Sneakers

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Dicta cum impedit veniam

Solution

```
<div class="container flex flex-col">
  <img  ...  >
  <span> ... </span>
  <h3> ... </h3>
  <p> ... </p>
  <button class="self-end"> ... </button>
</div>
```

Demo Here

We are using the utility class self-end , which translates to align-self: flex-end . This makes only the button align to the end along the cross axis. Remember, main axis and cross axis?

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. Note that:

1. self-* classes are applied to a flex item, whereas items-* are applied to a flex container
2. self-* takes effect only on the item it's applied to, whereas items-* works for all the flex items within the container.

The available utility classes are also similar to items-*.

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)*

Example #12-B

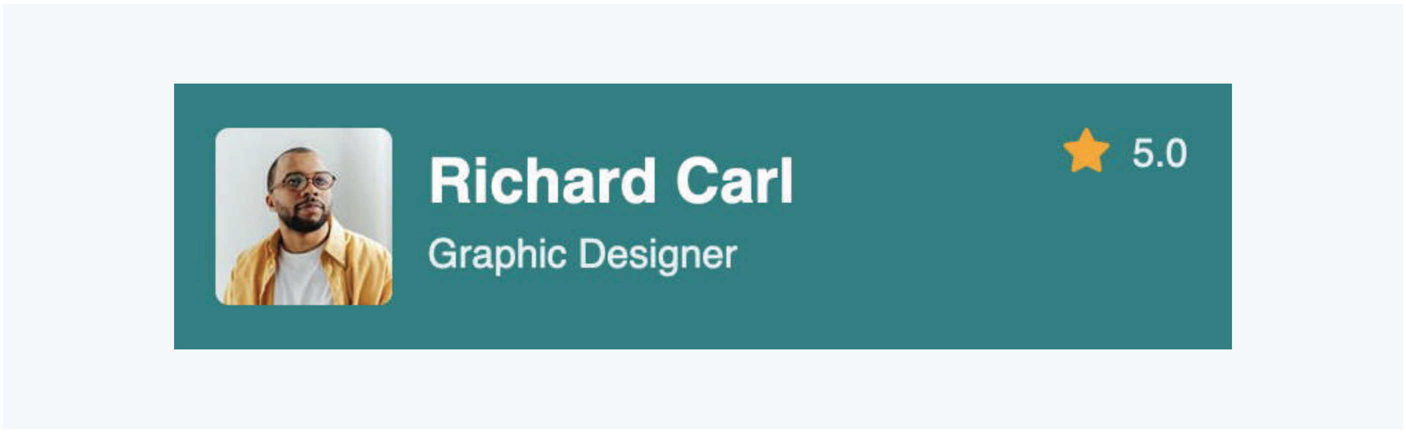
Profile with Rating

This is a small variation to the profile card we saw in Example 4b. 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.

Example #12-B

Profile with Rating

This is a small variation to the profile card we saw in Example 4b. 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.



```
<div class="container">
  <img ... >
  <div> ... </div>
  <div class="rating"> ... </div>
</div>
```

Solution

You need two Tailwind classes for the rating div - one for pushing it to the right (alignment along main axis) and another for aligning it at the top (alignment along cross axis).

```
<div class="container flex items-center">
  <img ... >
  <div> ... </div>
  <div class="rating ml-auto self-start"> ... </div>
</div>
```

Demo Here

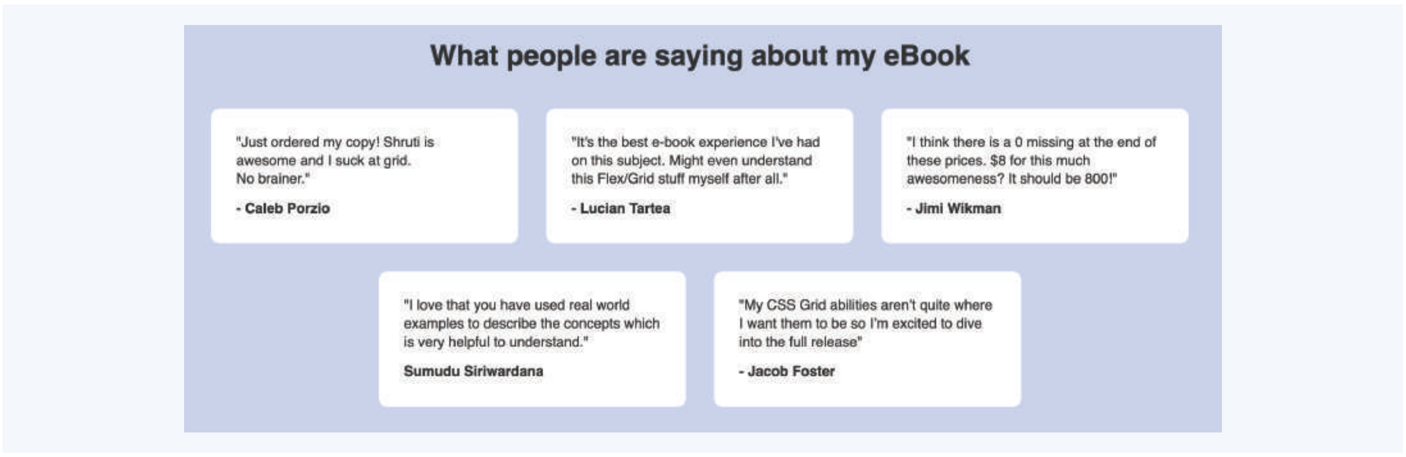
This example might help you understand why we have a CSS property align-self (self-* utility classes) but no property like justify-self because we can simply use auto margins to space out or align a single item along the main axis.

13. Align Content

Example #13-A

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.



What people are saying about my eBook

"Just ordered my copy! Shruti is awesome and I suck at grid. No brainer."
- Caleb Porzio

"It's the best e-book experience I've had on this subject. Might even understand this Flex/Grid stuff myself after all."
- Lucian Tartea

"I think there is a 0 missing at the end of these prices. $8 for this much awesomeness? It should be 800!"
- Jimi Wikman

"I love that you have used real world examples to describe the concepts which is very helpful to understand."
Sumudu Siriwardana

"My CSS Grid abilities aren't quite where I want them to be so I'm excited to dive into the full release"
- Jacob Foster

You might think this is what items-center does. But no. That works only for single row flex items.

```html
<div class="container">
  <div class="testimonial"> … </div>
  <!—— Four more testimonial divs ——>
</div>
```

One of the options is to wrap all the .testimonial elements in another div and center align that div vertically. But that's an unnecessary addition of an element to the DOM.

Solution

```html
<div class="container flex flex-wrap justify-center content-center">
  …
</div>
```

Demo Here

All we need to do is use the content-center utility class instead of items-center .

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

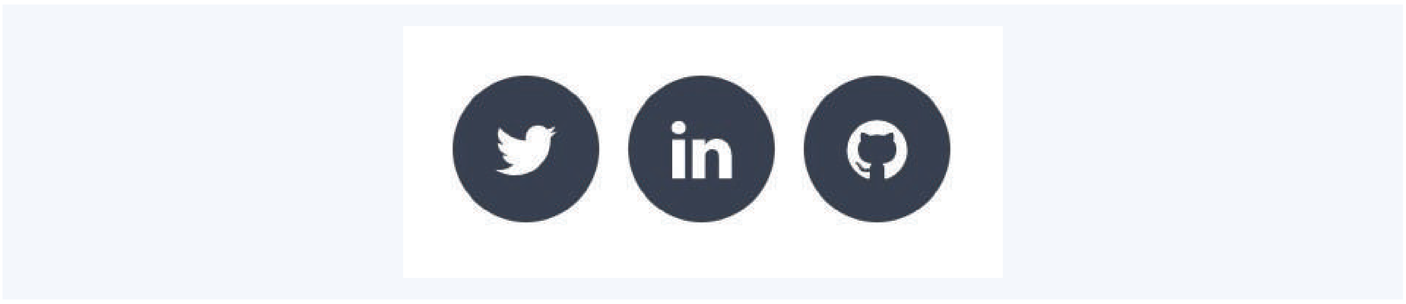
Note: This property is very rarely used. So it's totally okay if you cannot remember it

14. Inline Flex

Example #14-A

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.



To center each icon within the circle, we can add flex to the circle and center using justify-center and items-center . But that leaves us with the circles stacked one below the other, instead of next to each other

```html
<a class="icon flex justify-center items-center" href="#">
  <i class="fa fa-twitter"></i>
</a>
<a class="icon flex justify-center items-center" href="#">
  <i class="fa fa-linkedin"></i>
</a>
<a class="icon flex justify-center items-center" href="#">
  <i class="fa fa-github"></i>
</a>
```

Solution

Now simply change flex to inline-flex

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

Demo Here

Understanding Inline Flex

All the utilities that we have seen so far, either applied on flex container or on flex items, affect the **flex items** in one or other way - by changing their dimensions, position or alignment within the container. But 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

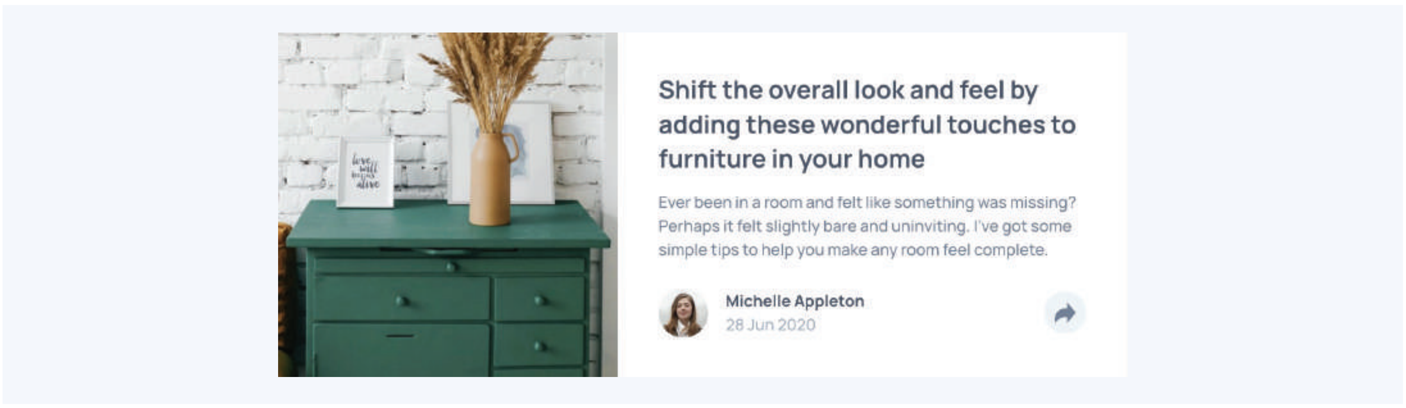
That's how we make the icons appear next to each other (inline) in our previous example where each icon itself is a flex container.

15. Comprehensive Examples for Flexbox

Example #15-A

Article Preview

This article preview component is a challenge from the Frontend Mentor website.



Shift the overall look and feel by adding these wonderful touches to furniture in your home

Ever been in a room and felt like something was missing? Perhaps it felt slightly bare and uninviting. I've got some simple tips to help you make any room feel complete.

Michelle Appleton
28 Jun 2020

Following the mobile-first approach, make this component responsive, by setting the outer flex container's direction to column and for larger screens, change it to row .

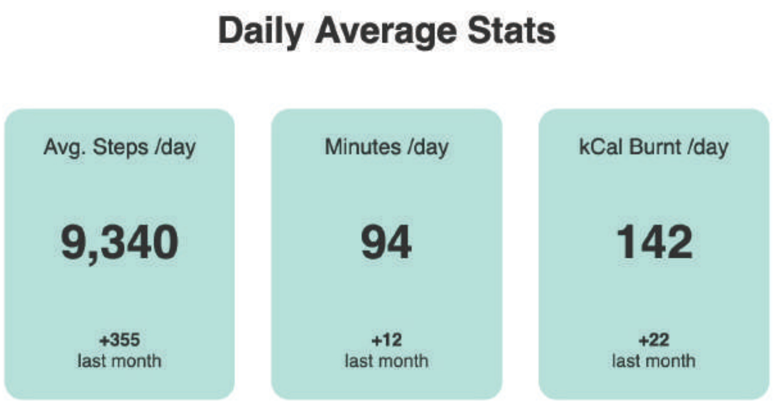
HINT: Use the utilities flex , flex-col , align-items , ml-auto and arbitrary values for flex-* to achieve the 40% and 60% widths.

Demo Here

Example #15-B

Fitness Report

Here's a simple fitness report component to test your newly gained flexbox skills



Daily Average Stats

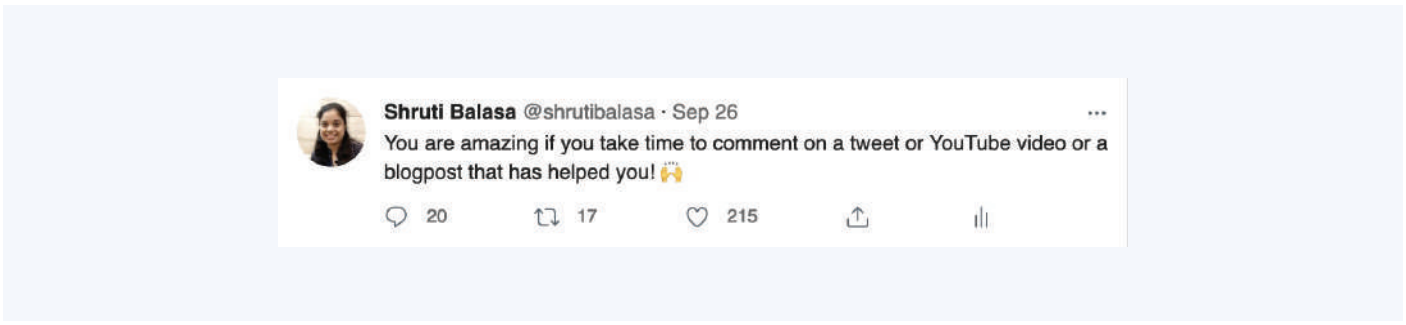
Avg. Steps /day	Minutes /day	kCal Burnt /day
9,340	94	142
+355 last month	+12 last month	+22 last month

Demo Here

Example #15-C

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).



Shruti Balasa @shrutibalasa · Sep 26 ···
You are amazing if you take time to comment on a tweet or YouTube video or a blogpost that has helped you! 🙌

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",

HINT: Use the utilities flex , justify-* , items-* and auto margins.

👉 Demo Here

Whoa! This really completes almost everything you can do with CSS flexbox. Take some time to digest all this, practice some more with other components and layouts you observe. And then come back to learn CSS Grid.

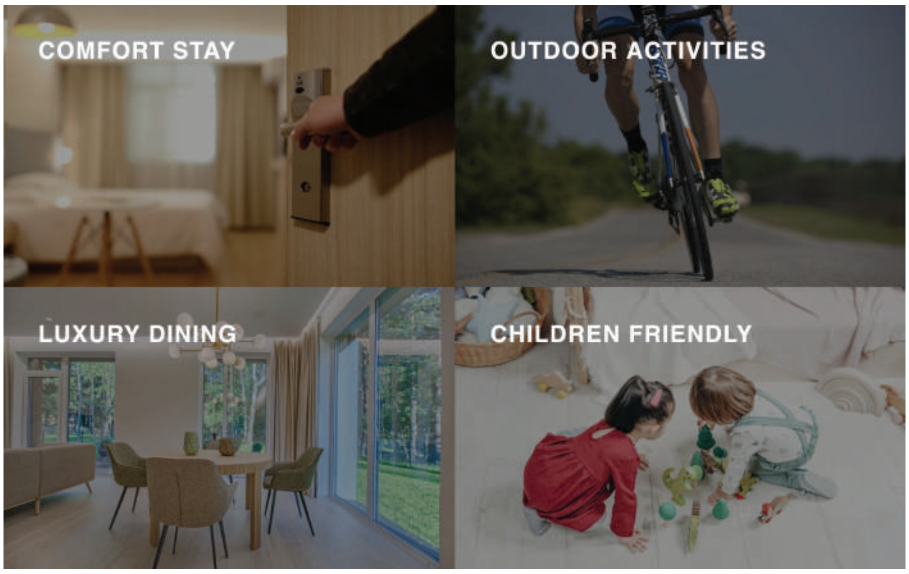
Caution: Grid is slightly more complex than flexbox! Be prepared.

16. Display Grid & Grid Template Columns

Example #16-A

Full Page Gallery

Let's say you want to create a gallery page for a resort, listing all the albums in a grid fashion occupying full screen like this.



This is surely possible using float , table or flex . But if you want a simpler solution, grid is the best option. If you already have an idea about grid or if you wish it to try this layout any other way, feel free it to give it a shot.

```
<div class="container min-h-screen">
  <div class="item">…</div>
  <!-- Three more items -->
</div>
```

Solution

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

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

Demo Here

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. Similar to flexbox, we can control the the size, alignment, placement and order of these elements using grid. Here again, we need at least two elements - a parent element called **grid container** and at least one child element called **grid item**.

In our above example, adding **grid** class makes the .container element a grid container.

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

Understanding Grid Template Columns

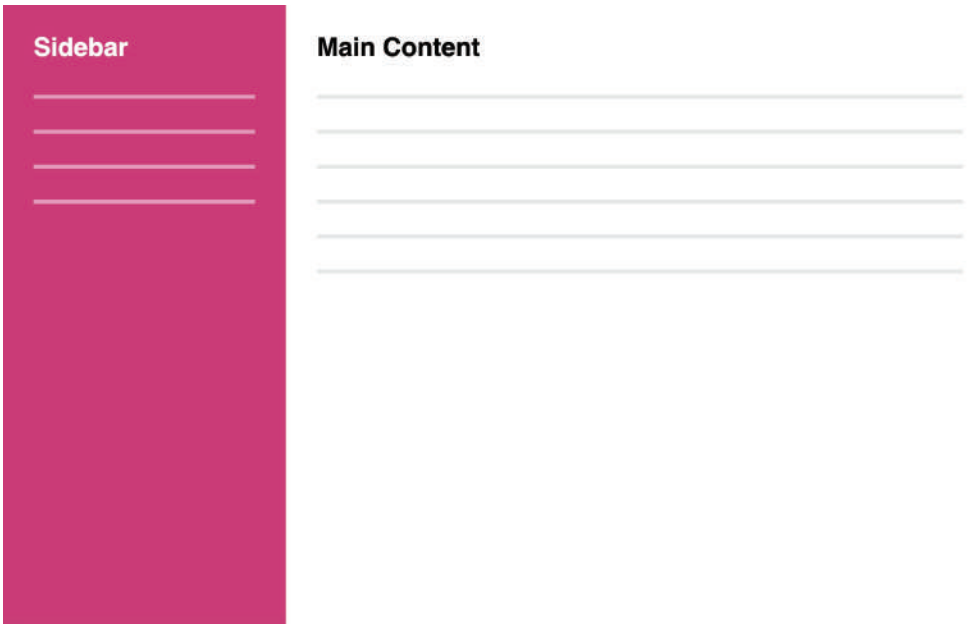
The utilities grid-cols-* is used to specify how many columns you need and of what size each. Majority of the use cases for grid require creating equal width columns, and hence Tailwind provides these utility classes where you can create one to twelve columns of equal width.

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

Example #16-B

Layout with Sidebar

This is a common layout with a sidebar on left and main content on the right. There are multiple ways of achieving this, but grid makes it simplest.



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

Solution

We need two columns here - .sidebar with a fixed width and .main that takes up the remaining space. This is possible with an fr unit in grid-template-columns . Here's the Tailwind solution:

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

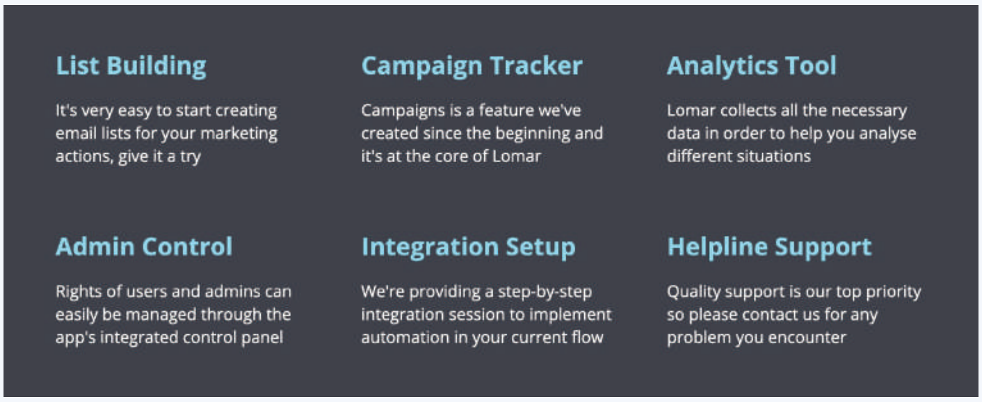
Along with grid class, we added grid-cols-[22rem,1fr] which translates to gridtem-plate-columns: 22rem 1fr . So we get two columns - first one with fixed 22rem width and second one which occupies the remaining space

Demo Here

Example #16-C

Services Grid

This is a classic example of grid - listing services or features in a grid format with equal width columns.



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 in order to help you analyse different situations

Admin Control

Rights of users and admins can easily be managed through the app's integrated control panel

Integration Setup

We're providing a step-by-step integration session to implement automation in your current flow

Helpline Support

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

Solution

We need three columns of equal width. In Tailwind, we have already seen how simple it is to do the same.

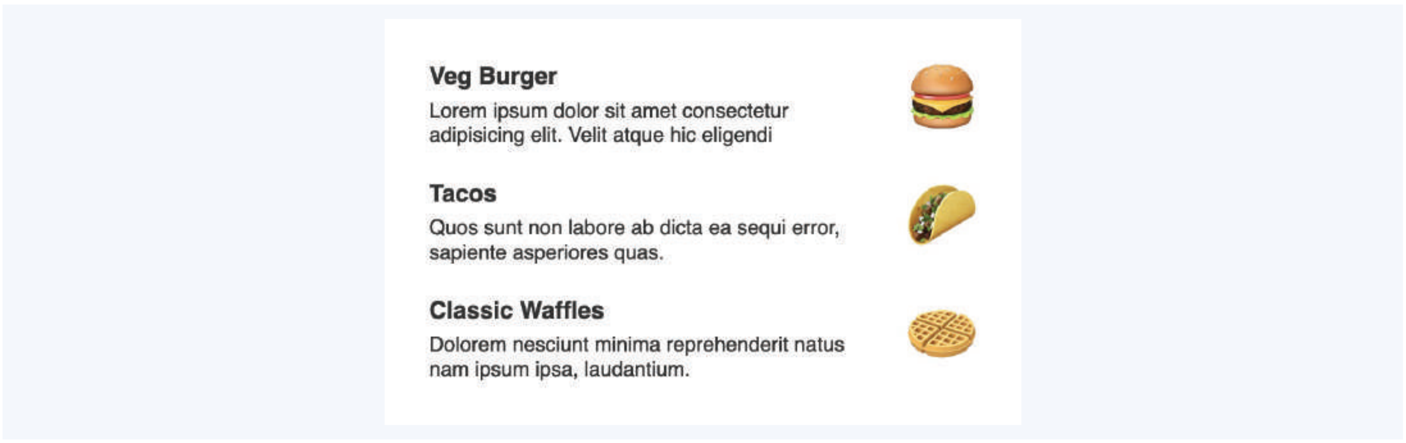
```html
<div class="container grid grid-cols-3">
  <div class="item">...</div>
  <!-- Five more items here -->
</div>
```

👉 Demo Here

Example #16-D

Quick Bites Menu

Usually restaurant menus are displayed in a grid fashion. Here's one such example with he item name and description on the left and a picture on the right.



Veg Burger
Lorem ipsum dolor sit amet consectetur adipisicing elit. Velit atque hic eligendi

Tacos
Quos sunt non labore ab dicta ea sequi error, sapiente asperiores quas.

Classic Waffles
Dolorem nesciunt minima reprehenderit natus nam ipsum ipsa, laudantium.

Solution

We need two columns of unequal width, so we can use grid-cols-* with arbitrary values to specify two values separated by commas. As you might have guessed, the first value is 1fr . You can specify a fixed width for the second column. But what's even better is to use the keyword auto . This keyword lets the content of items decide the size of the column / row.

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

Demo Here

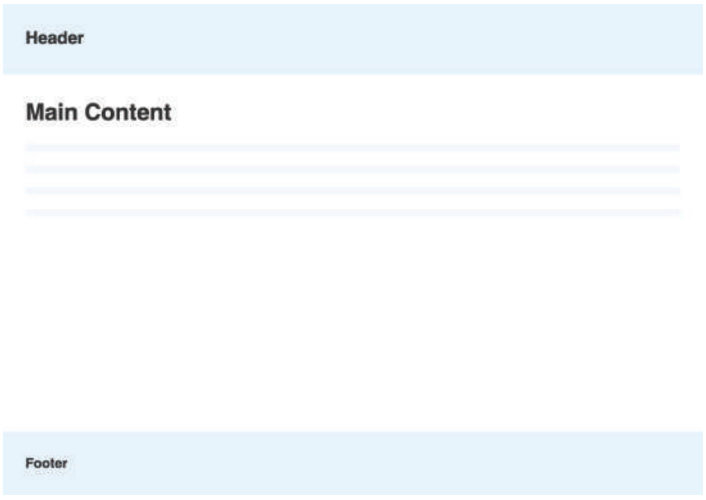
So far we used fixed units, percentage values, fr units and the auto keyword for specifying grid-cols-* arbitrary values apart from the Tailwind utilities available. There are few more options that we'll cover under the Advanced Grid Template Values topic.

17. Grid Template Rows

Example #17-A

Sticky Footer with Grid

We already saw how to make a sticky footer using flex. Here's a similar example with an additional header element. If you can recall, we used flex-col and flex-grow to create this.



You might think this is what items-center does. But no. That works only for single row flex items.

```
<div class="container min-h-screen">
  <header> ... </header>
  <div class="main"> ... </div>
  <footer> ... </footer>
</div>
```

Solution

Look at the example and observe that we have a single column but multiple rows. This can be specified using grid-rows-* instead of grid-cols-* . We need 3 rows - first and third rows with auto height and the second row occupying all the remaining height.

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

👉 Demo Here

Note: This demo works as long as you have only three elements header , footer and .main . If you add an additional element at the same level, it breaks. Whatever additional content you want, you need to add it within the .main div.

Understanding Grid Template Rows

The property grid-template-rows in CSS is used to specify how many rows you need and of what size each. Similar to grid-template-columns the height of rows can be specified in % , px , rem or any valid value for height separated by spaces. The number of individual values you specify will be the number of rows created.

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

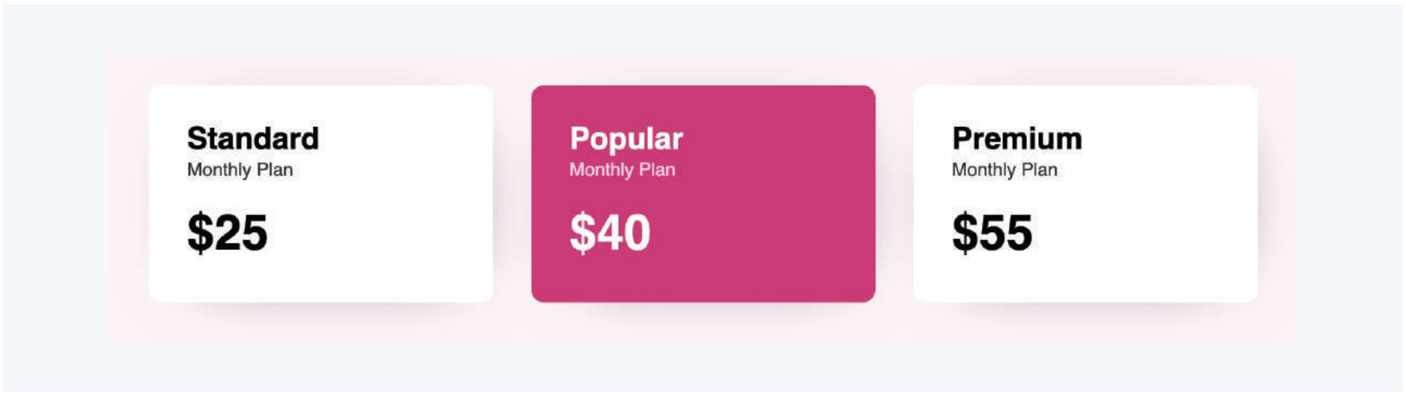
We will also look at more examples using grid-rows-* after covering a few more concepts. For now, I'm sure you have a basic idea. And of course, you can use both grid-col-* and grid-rows-* at the same time.

18. Gap

Example #18-A

Pricing Plans with Grid

Let's look at the same pricing plans example once again, this time using grid. By now you know how to create three equal sized blocks (or equal sized columns) with grid. But let's also look at how to add those spaces between those blocks.



```html
<div class="container">
  <div class="plan"> ... </div>
  <div class="plan"> ... </div>
  <div class="plan"> ... </div>
</div>
```

Solution

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

Demo Here

Understanding Column Gap

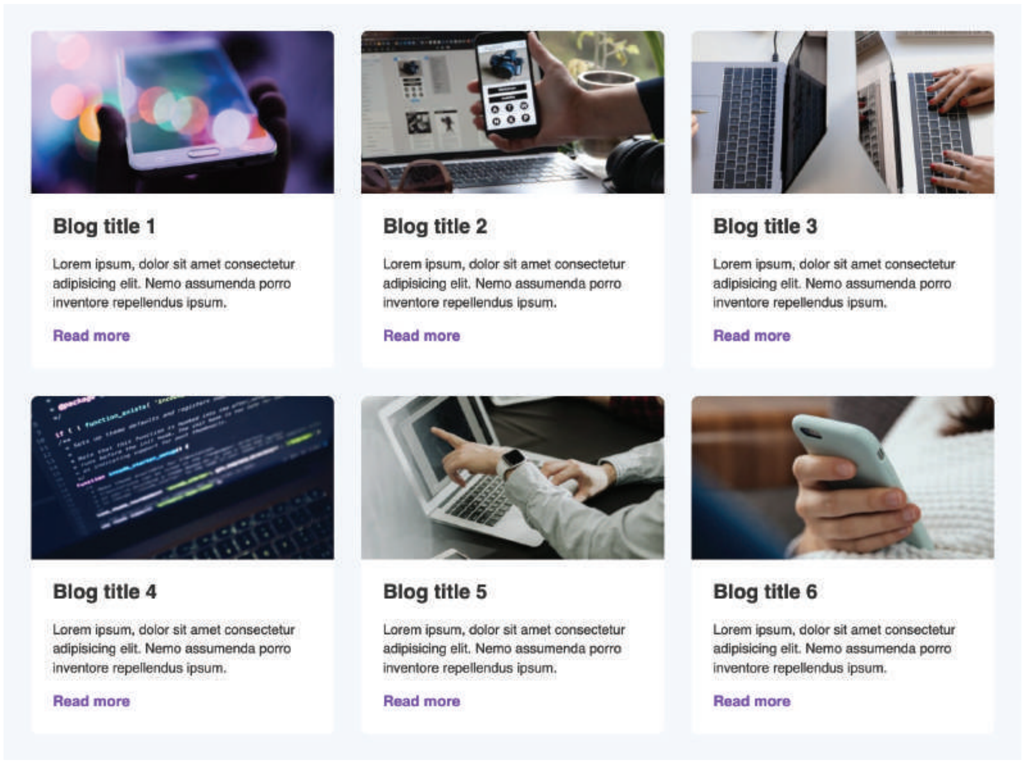
The gap-x-* utilities set the size of the horizontal gap (also known as gutters) between columns. Like I mentioned earlier in this book, this property can be used with flexbox too, but doesn't have good browser support yet. With grid however, it is better supported. Some of the common utilities for gap-x-* are here:

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

Example #18-B

Blog Posts Display

This is a classic use case for grid - display of blog post cards in a grid format with horizontal and vertical spacing between each card. Let's create this layout using grid and also make it responsive.



Blog title 1

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Blog title 2

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Blog title 3

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Blog title 4

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Blog title 5

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Blog title 6

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Check the below link to see how this layout is created with Grid and made responsive using the mobile first approach. Now see if you can add some horizontal and vertical spacing between the items using gap-x-* and a similar property gap-y-* .

```html
<div class="container">
  <div class="item">…</div>
  <!-- Five more item cards -->
</div>
```

Responsive Solution

Using the mobile first approach, we just add a grid class at first. This automatically creates one column. At sm breakpoint, we change that to two columns using grid-cols-2 and at md breakpoint, we change it to three columns using grid-cols-3 . We also add spacing of 2rem between columns and rows with gap-x-8 and gap-y-8 .

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

Demo Here

Better Solution

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

Understanding Row Gap

The gap-y-* utilities set the size of the vertical gap (also known as gutters) between rows. Again, this property can be used with flexbox too, but doesn't have good browser support yet. With grid however, it is better supported. The available utilities are similar to that of gap-x-* .

Understanding Gap

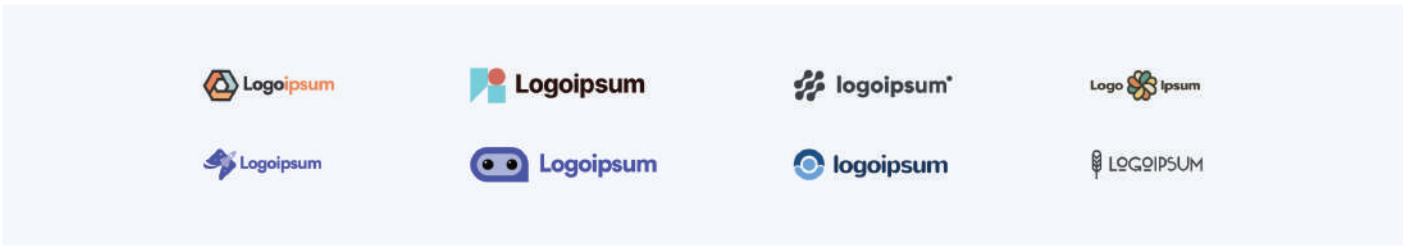
The utility gap is used to set the same spacing between rows and columns at once. The available utilities are similar to that of gap-x-* and gap-y-* . For all the available values.

19. Justify Content

Example #19-A

Featured Logos in a Grid

We have seen a similar example of logos with flexbox. But with flex, you can only align logos in one direction. You cannot control the alignment in the other direction.



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

Solution

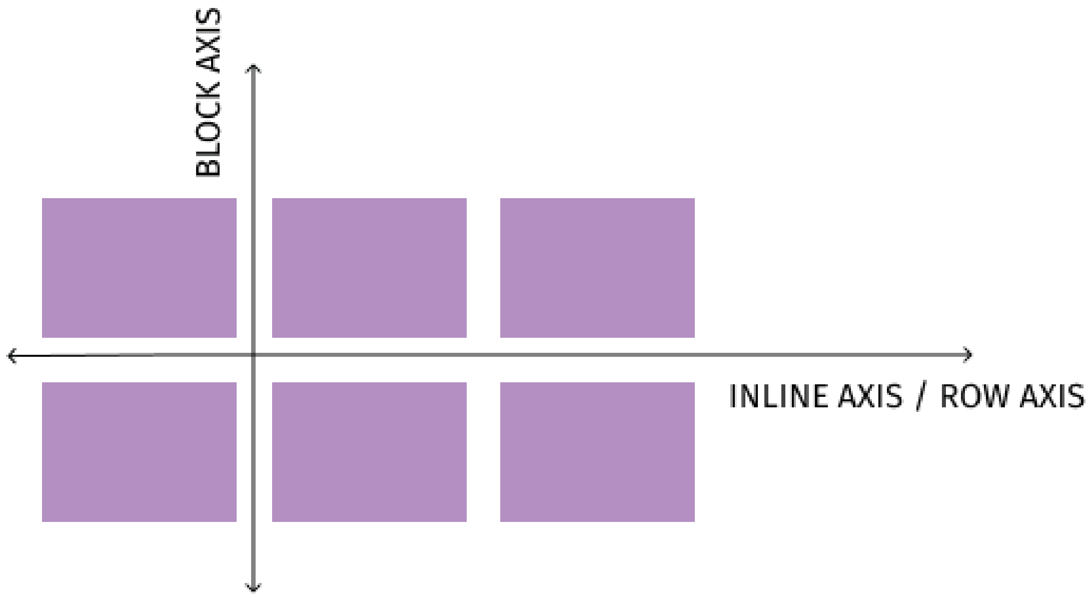
We need one more utility class now - justify-between to space out the columns. This is similar to what we did with flexbox.

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

Demo Here

Understanding Justify Content in Grid

Before we talk about this property with reference to Grid, you need to know two more terms. 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**.

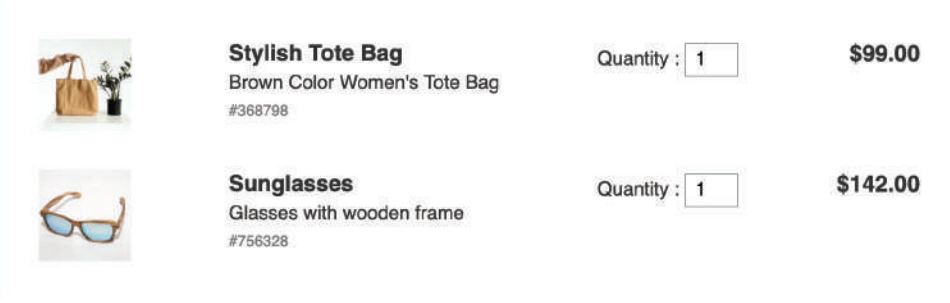


Tailwind Class	CSS Property & Value	Explanation
`justify-start`	`justify-content: flex-start;`	All columns are placed at the beginning of the container
`justify-end`	`justify-content: flex-end;`	All columns are placed at the end of the container
`justify-center`	`justify-content: center;`	All columns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

Example #19-B

Shopping Cart Summary

Here's another great example for CSS Grid - a shopping cart summary with each row containing an image, product description, quantity and price. You can use justify-* here too, to space out the columns.


Stylish Tote Bag
Brown Color Women's Tote Bag
#368798
Quantity : 1
$99.00

Sunglasses
Glasses with wooden frame
#756328
Quantity : 1
$142.00

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

Solution

Along with using justify-between property, we also need to add a text-right to the .price element to align the text in the last column to right.

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

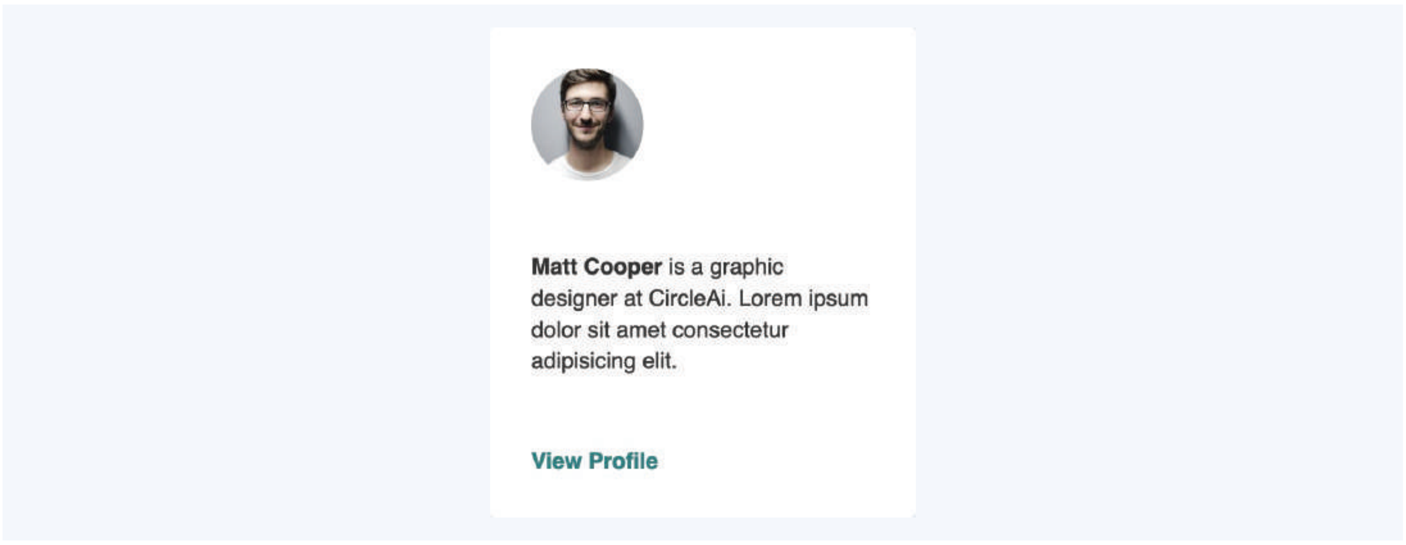
👉 Demo Here

20. Align Content

Example #20-A

Profile Card with Bio & Link

Assume you need a profile card with a fixed height containing the profile picture, short bio and a link - all three elements spaced out equally.This can be achieved using flexbox too,but the solution with grid is one utility class lesser.



Matt Cooper is a graphic designer at CircleAi. Lorem ipsum dolor sit amet consectetur adipisicing elit.

```
<div class="card h-96 grid">
  <img ...  />
  <p> ... </p>
  <a> ... </a>
</div>
```

Solution

For this one, we need only one column, so adding grid to the .card element automatically creates a grid with one column and three rows with auto heights. Now we just need to control the vertical spacing using content-* utilities.

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

Demo Here

Understanding Align Content in Grid

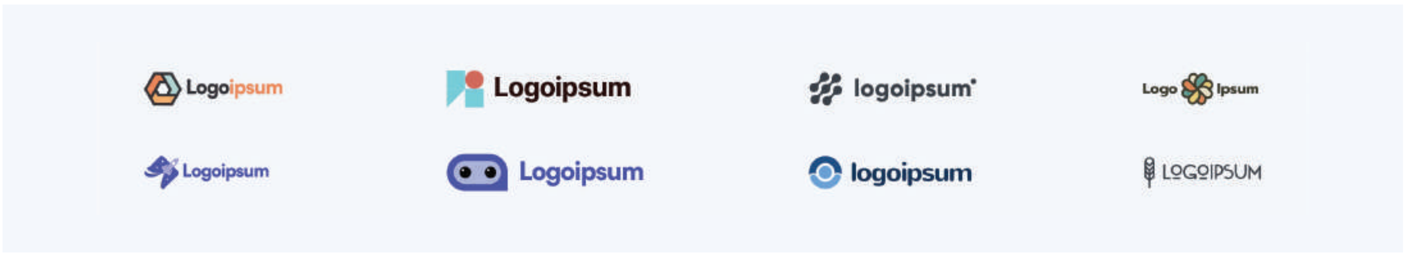
We have seen content-* utilities with respect to flexbox to control spacing between multiple lines of wrapped items along the cross axis. In Grid, these are used to control spacing between the rows. content-between is one of the available utilities we just used. The available utilities are similar to that of justify-* .

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 center
`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

Example #20-B

Featured Logos Center of Page

In Example 19a, we looked at displaying logos in a grid and spacing them horizontally with justify-between . Now what if we want both the rows to be at the center of the page vertically?



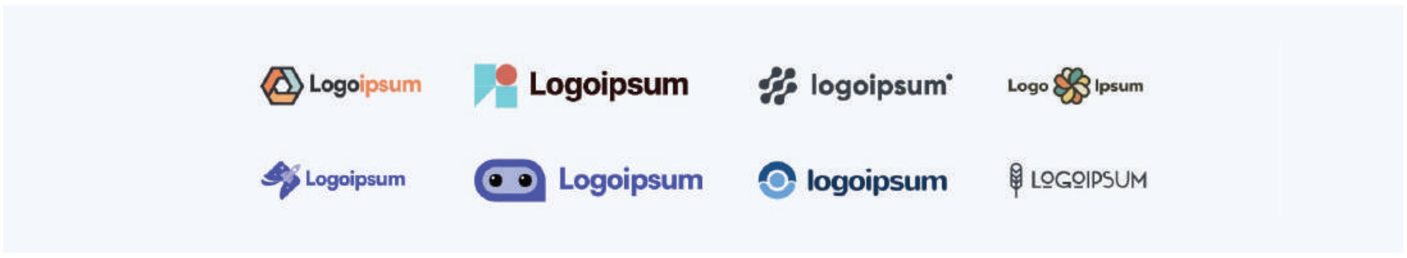
Solution

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

Demo Here

A small variation

Now assume you don't want the logos to space out horizontally, instead you want them to be centered horizontally too.



Solution

For this, you can use justify-center instead of justify-between . So now we have:

```html
<div class="container ... justify-center content-center">
  ...
</div>
```

Better Solution

```html
<div class="container ... place-content-center">
  ...
</div>
```

We have combined justify-center and content-center into place-content-center .

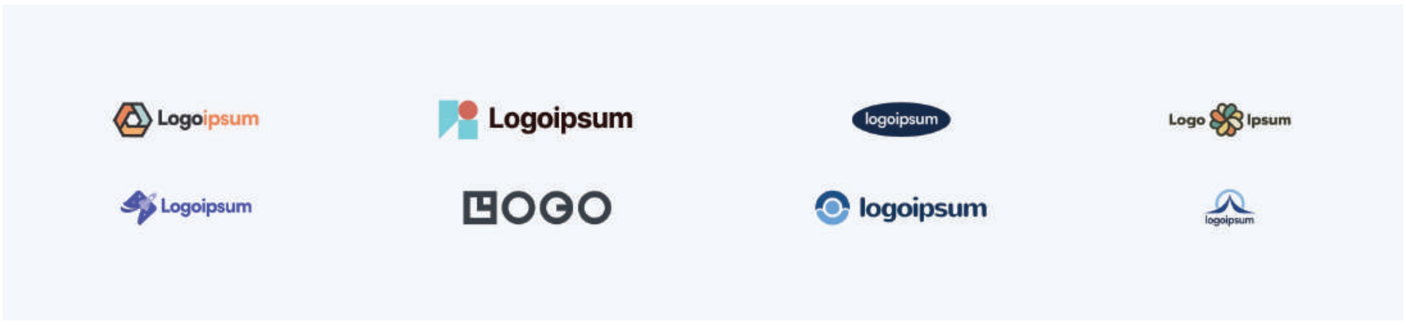
The place-content-* utilities allows you to control the spacing of grid items along both the block and inline axes at once. But this is possible only when you want the placement of rows and columns to be the same. In the previous example, we wanted the rows AND columns to be placed at the center of the grid container. Hence we could use the place-content* utilities. The available utilities are similar to that of justify-* and content-* .

21. Justify Items

Example #21-A

Featured Logos of Different Widths

We're back with the same example with yet another variation. Previously all the logos we used were approximately of the same dimensions, hence it looked good.



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

Solution

There's one new utility class to our rescue - justify-items-* .

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

Demo Here

Understanding Justify Items

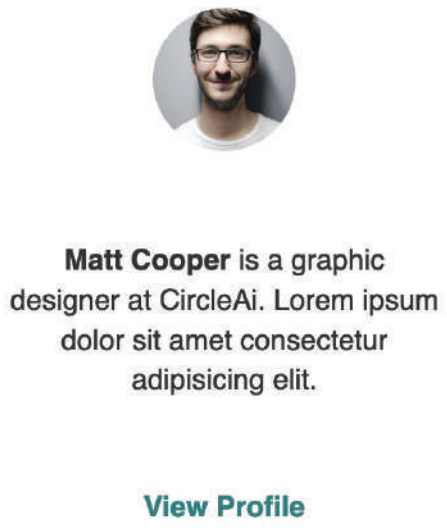
The justify-items-* utilities allows us to **horizontally** align the content within the columns, while the previous utilities justify-* allows us to control spacing of the entire columns. The available utilities in Tailwind are:

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

Example #21-B

Profile Card with Bio & Link Centered

We are revisiting Example 20a, this time making everything center aligned horizontally.



Matt Cooper is a graphic designer at CircleAi. Lorem ipsum dolor sit amet consectetur adipisicing elit.

View Profile

Solution

There's one new utility class to our rescue - justify-items-* .

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

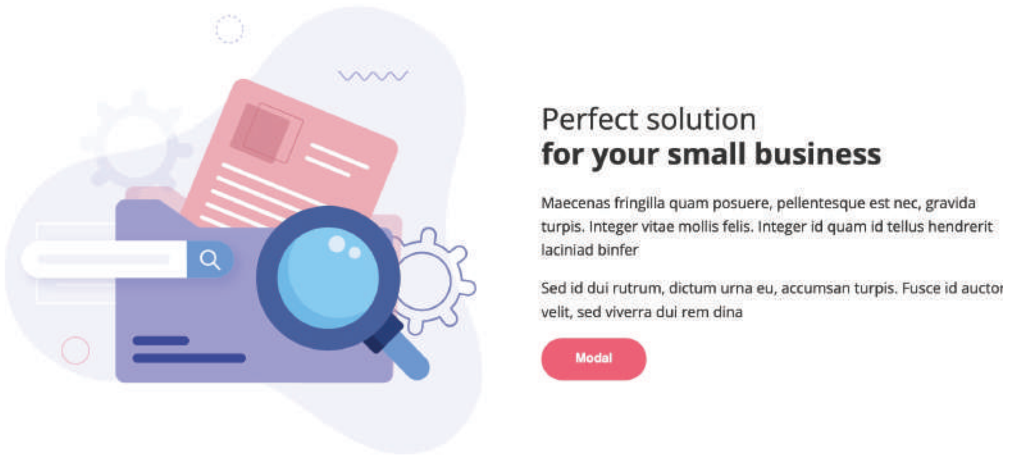
Demo Here

22. Align Items

Example #22-A

Image and Text Section

One very common section in web pages is an image on the left half and text on right half of the page. You need the text and image to be perfectly center aligned vertically for all large screen sizes. Grid is great for something like this.



Perfect solution
for your small business

Maecenas fringilla quam posuere, pellentesque est nec, gravida turpis. Integer vitae mollis felis. Integer id quam id tellus hendrerit laciniad binfer

Sed id dui rutrum, dictum urna eu, accumsan turpis. Fusce id auctor velit, sed viverra dui rem dina

Solution

Using grid-cols-2 we have created two equal sized columns. Using gap , we have added some spacing between them. To center align the image and text vertically in the page, we need to use the items-* utility, very similar to flexbox.

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

Demo Here

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. We have already seen the available utilities when we covered this concept with respect to flexbox.
Here they are again, for reference.

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)*

Example #22-A

Image and Text Section

Let's look at the same example once again. So far, we used a fixed height h-10 for all the logos. Now we'll remove that and instead add a max-w-[10rem] (Check the CSS tab).

Logoipsum
Logoipsum
logoipsum
Logo Ipsum
Logoipsum
LOGO
logoipsum
logoipsum

Solution

We can use the utility items-center to vertically center the taller and shorter logos within each row

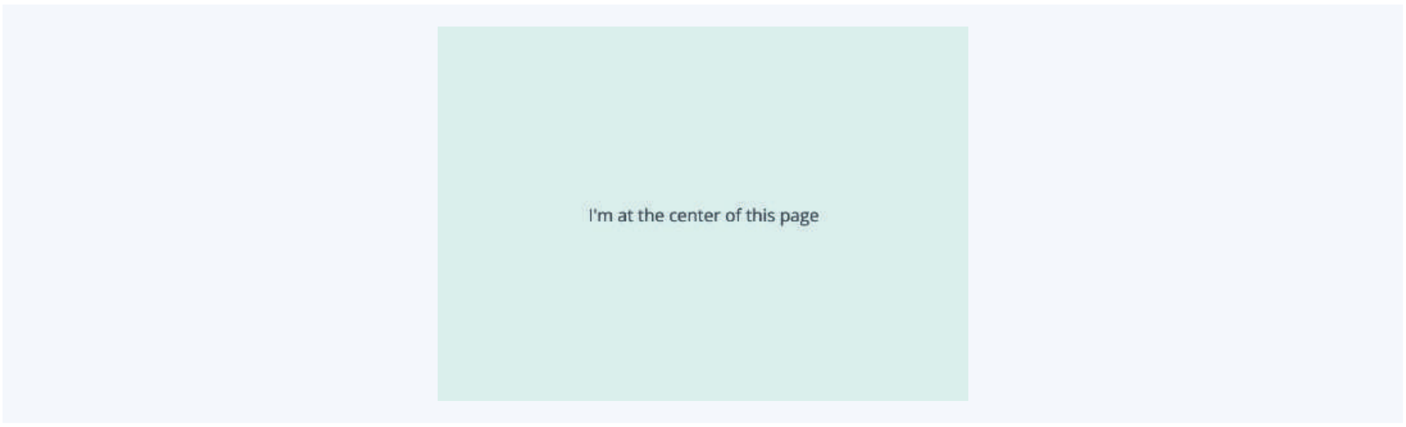
```
<div class="container ... justify-between justify-items-center items-center">
  ...
</div>
```

23. Place Items

Example #23-A

Center a div using Grid

We have already seen how easy it is to center a div using flexbox. With grid, it's one lesser utility class.



Solution

You can either use the previous two utilities justify-items-center and items-center along with grid . Or you can combine both these using the place-items-* utilities.

```html
<div class="container grid place-items-center">
  <div class="item">
    ...
  </div>
</div>
```

Demo Here

Understanding Place Items

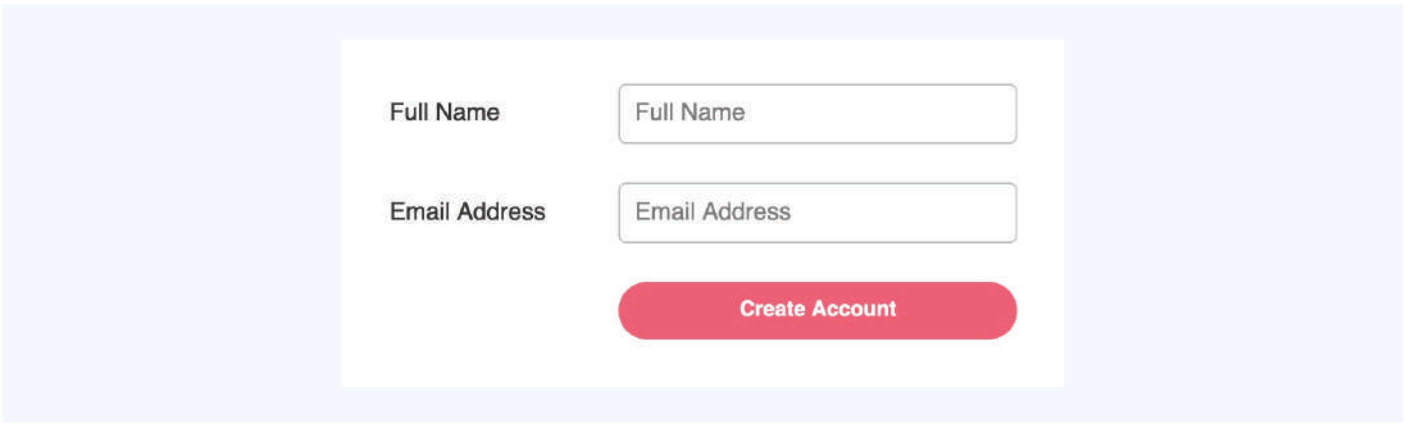
The place-items-* utilities allows you to align the items horizontally within columns and vertically within rows at once. But this is possible only when you want the same alignment in both directions. In the previous example, we wanted the item to be at the center horizontally and vertically. Hence we could use the place-items-* utilities. The available utilities are similar to that of justify-items-* and items-* .

24. Grid Column Start, End & Span

Example #24-A

Horizontal Form

Creating forms and making them responsive is so much more easier with grid than any other tool. Here's the simplest example of a horizontal form with labels on the left, inputs on the right and a button on the right too.



Solution

Of course, one solution is to add a dummy element in HTML before the button, but that's definitely not a good practice. And there's a simple utility class available for this:

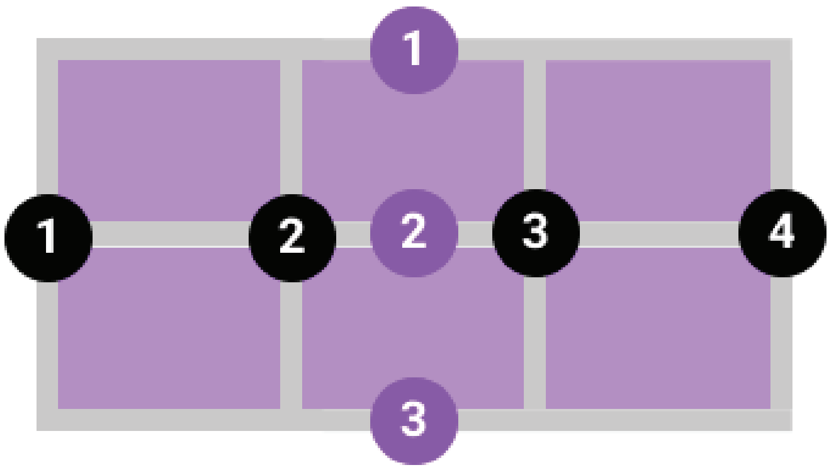
```html
<form class=" ... ">
  ...
  <button class="col-start-2"  ... >Create Account</button>
</form>
```

Demo Here

We have used col-start-2 on the **grid** item to change the column it appears in. Let's learn about this.

Understanding Column Start

Before we learn more about the col-start-* utilities, we need to learn about grid lines. When you define a grid using grid-cols-* and/or grid-rows-* , grid lines are created. These are nothing but the lines between and around the columns and rows. This picture explains how the column lines and row lines are numbered.



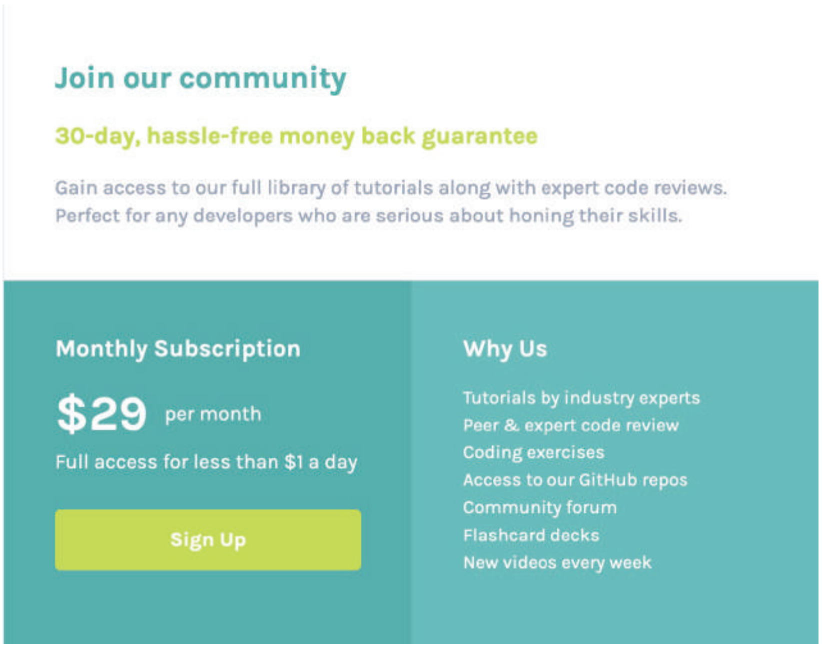
It's important to remember that the line numbers start from 1 and not 0

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

Example #24-B

Single Price Grid Component

This grid component is a challenge from the Frontend Mentor website. This is responsive with the mobile version having just one column with all three items one below the other



Join our community

30-day, hassle-free money back guarantee

Gain access to our full library of tutorials along with expert code reviews.
Perfect for any developers who are serious about honing their skills.

Monthly Subscription

$29 per month

Full access for less than $1 a day

Sign Up

Why Us

Tutorials by industry experts
Peer & expert code review
Coding exercises
Access to our GitHub repos
Community forum
Flashcard decks
New videos every week

Solution

Above the `sm breakpoint, we need to make the .component-header start at column line 1 and end at column line 3.

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

Demo Here

Let's learn about col-end-* utilities and few more ways of getting the same result.

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-* . In CSS, you can also use a negative integer for grid-column-end , in which case it starts counting the lines in reverse, starting from -1 .

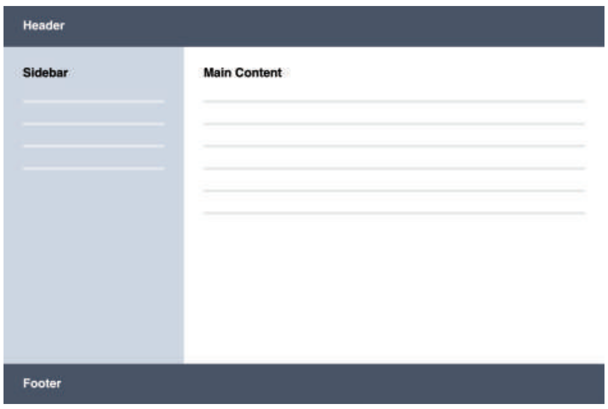
Understanding Column Span

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 upto col-span-12 along with a helpfult which makes the grid item span across all the columns in the grid.

Example #24-C

Page Layout with Grid

Let's look another layout that combines the above two with a header, sidebar, main content and a footer.



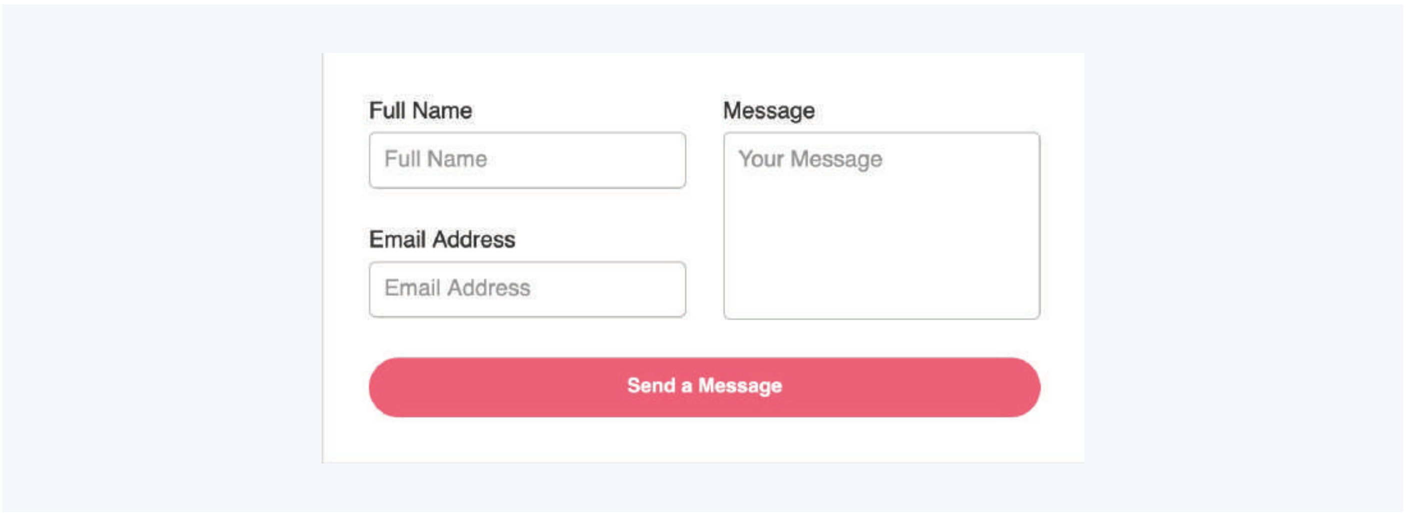
Demo Here

25. Grid Row

Example #25-A

Contact Form

Let's look a contact form with a couple of fields in the first column and one field in the second column. Previously we saw grid items spanning across columns, but here one grid item spans across rows too. The concept is the same.



Solution

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

Demo Here

We have already learned about the col-start utilities. Now we're using two new utilities row-start-* and row-end-* .

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.

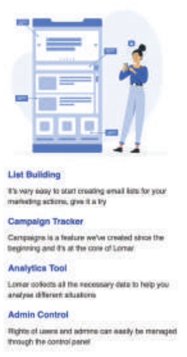
Understanding Row Span

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 upto row-span-12 along with a helpful row-span-full which makes the grid item span across all the columns in the grid.

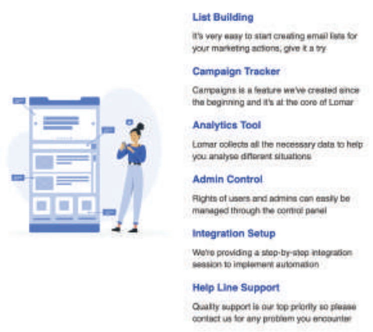
Example #25-B

Responsive Services Section

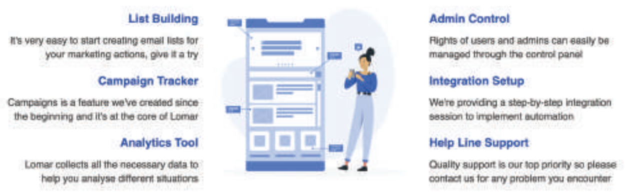
Mobile layout has one grid column, tablet layout has two grid columns and desktop layout has three grid columns. But more importantly, the grid items' placements change. This is quite simple now using col-start , col-end , row-start and row-end .



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

Solution

Following mobile-first approach, we start with one single column, change to two columns at sm breakpoint and to three columns at md breakpoint

lines along with the column line. Here, I have used `row-span-2`, but you can also use `row-end-3`.

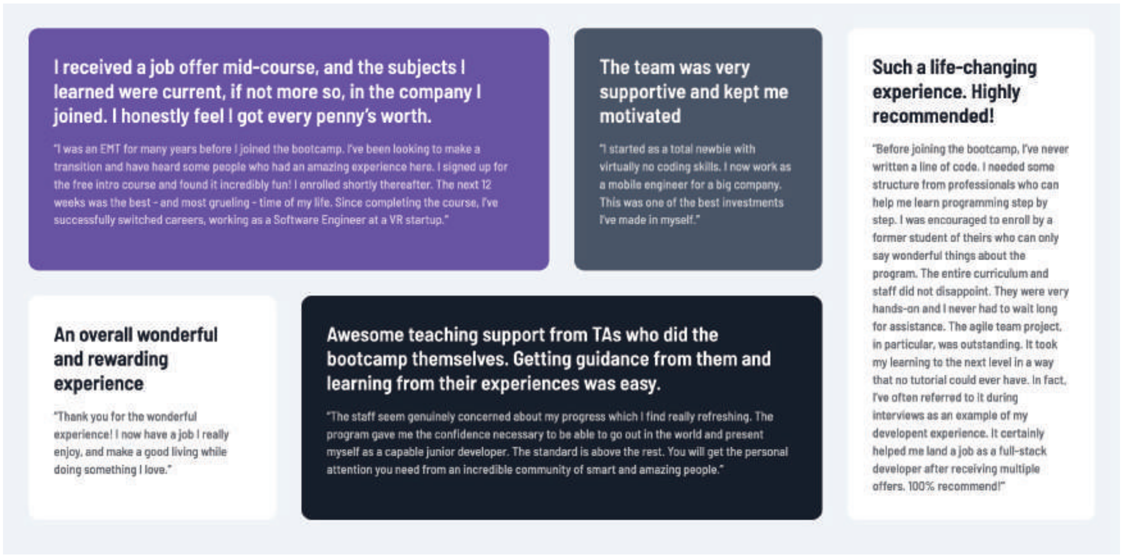
▸ Working Demo

👉 Demo Here

Example #25-C

Testimonials Grid Section

Here's another example from Frontend Mentor website, but with removed avatars and names for simplicity. Go for the mobile first approach with just one column and above lg breakpoint , try and achieve this layout.



I received a job offer mid-course, and the subjects I learned were current, if not more so, in the company I joined. I honestly feel I got every penny's worth.

"I was an EMT for many years before I joined the bootcamp. I've been looking to make a transition and have heard some people who had an amazing experience here. I signed up for the free intro course and found it incredibly fun! I enrolled shortly thereafter. The next 12 weeks was the best - and most grueling - time of my life. Since completing the course, I've successfully switched careers, working as a Software Engineer at a VR startup."

The team was very supportive and kept me motivated

"I started as a total newbie with virtually no coding skills. I now work as a mobile engineer for a big company. This was one of the best investments I've made in myself."

Such a life-changing experience. Highly recommended!

"Before joining the bootcamp, I've never written a line of code. I needed some structure from professionals who can help me learn programming step by step. I was encouraged to enroll by a former student of theirs who can only say wonderful things about the program. The entire curriculum and staff did not disappoint. They were very hands-on and I never had to wait long for assistance. The agile team project, in particular, was outstanding. It took my learning to the next level in a way that no tutorial could ever have. In fact, I've often referred to it during interviews as an example of my developent experience. It certainly helped me land a job as a full-stack developer after receiving multiple offers. 100% recommend!"

An overall wonderful and rewarding experience

"Thank you for the wonderful experience! I now have a job I really enjoy, and make a good living while doing something I love."

Awesome teaching support from TAs who did the bootcamp themselves. Getting guidance from them and learning from their experiences was easy.

"The staff seem genuinely concerned about my progress which I find really refreshing. The program gave me the confidence necessary to be able to go out in the world and present myself as a capable junior developer. The standard is above the rest. You will get the personal attention you need from an incredible community of smart and amazing people."

Solution

On mobile, you just have to apply grid and gap-8 to section and everything just works. Above the lg breakpoint, you need to create four columns and almost every grid item needs to be positioned using the row and column lines.

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

Most important is positioning of that .white-long div. You need to mention both the row lines along with the column line. Here, I have used row-span-2 , but you can also use row-end-3 .

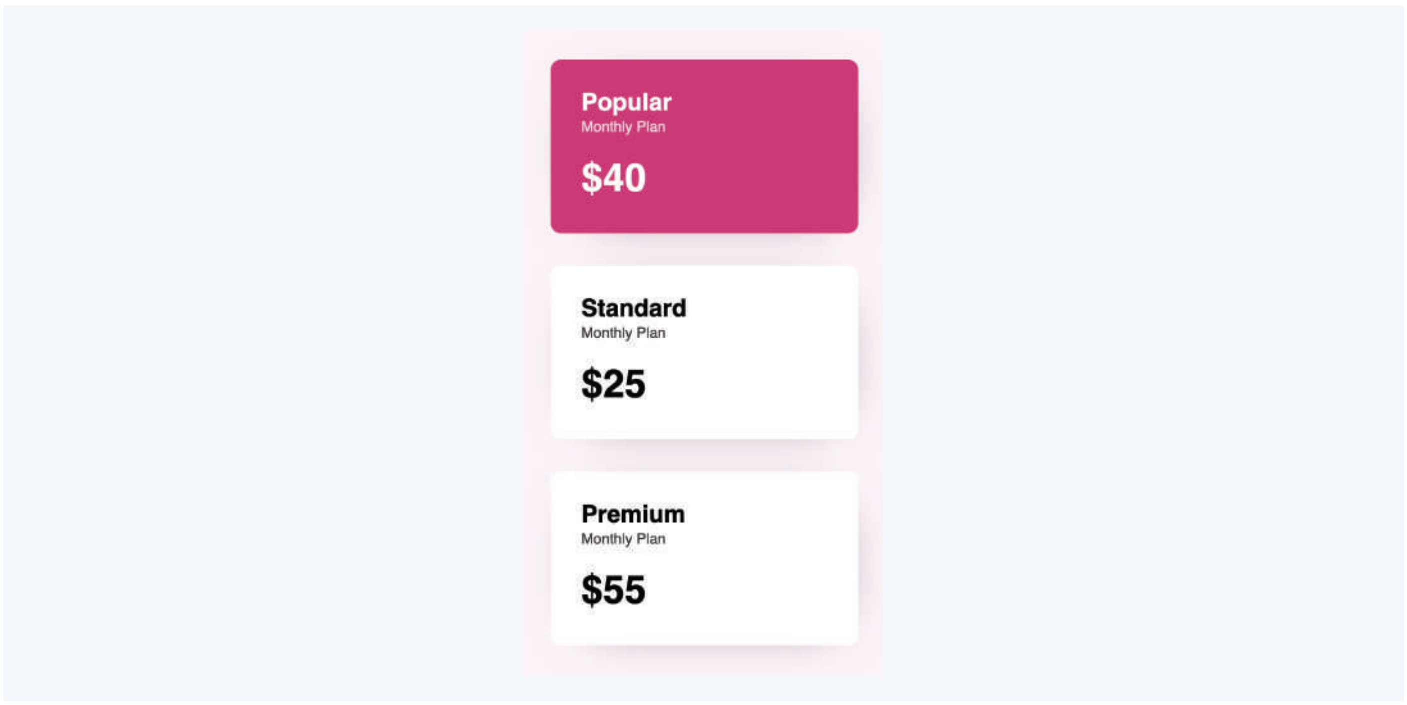
👉 Demo Here

26. Order

Example #26-A

Responsive Pricing Plans

Let's look at the Pricing Plans Example again and make it responsive with one change. On mobile screens, we place the **Popular** plan first, followed by **Standard** and **Premium** while keeping the order same on desktop.



Solution

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

Demo Here

Understanding Order in Grid

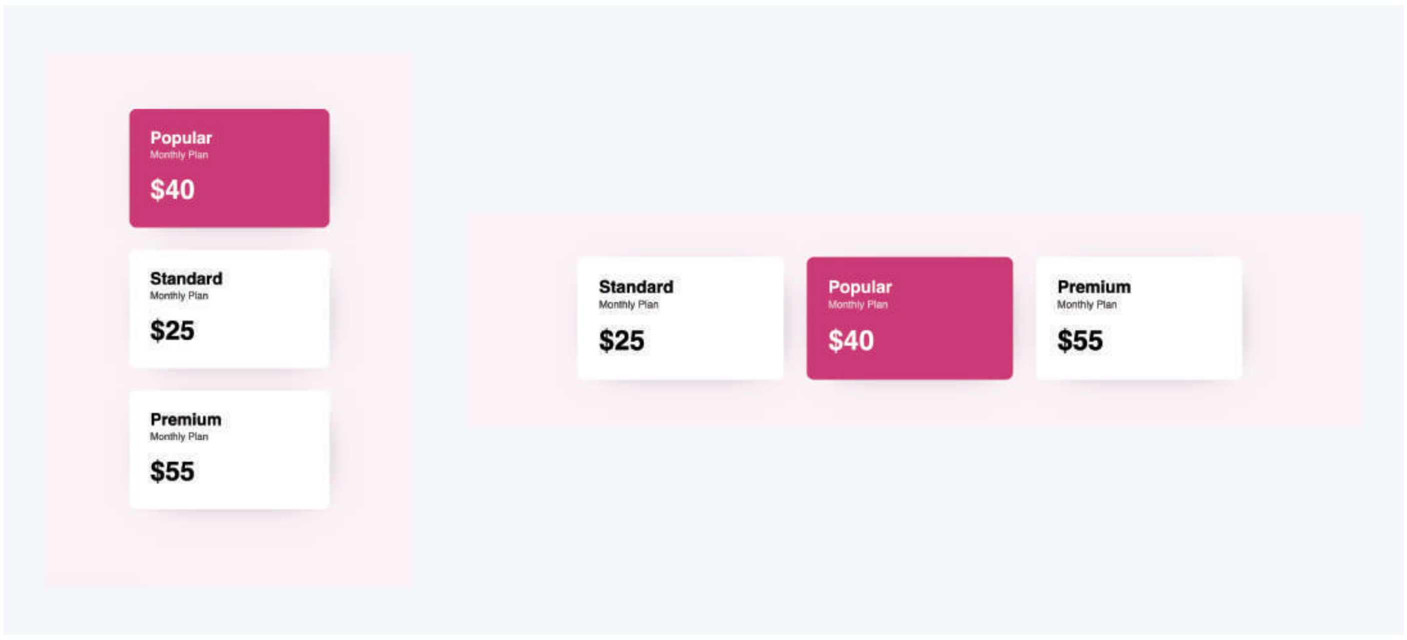
The same order utilities that we saw with respect to flexbox can be used for grid items too. The value can be any number - positive or negative. The items with greater order value appear later on the web page compared to the items with lesser value irrespective of their appearance in the markup.

27. Advanced Grid Template Values

Example #27-A

Pricing Plans with Size Limits

In the pricing plans example we saw earlier, you might have noticed that the columns stretch full width of the container even on mobile screens.



Solution

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

Demo Here

Understanding minmax()

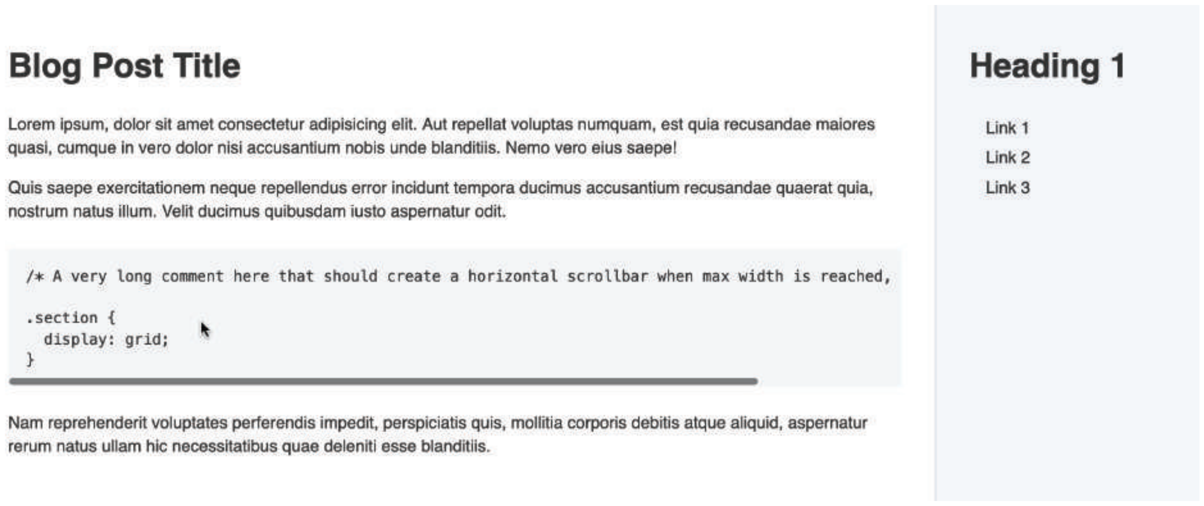
The **minmax()** function takes in two parameters - min and max. It specifies a size range greater than or equal to min and less than or equal to max. Both these values can be any length values in **px** , **%** , **rem** or even values like 1fr , min-content or max-content .

Example #27-B

Blog Post Page with Code Snippet

grid-cols-[22rem,1fr]

It's a simple solution and it usually works. But consider this blog post page example with a
similar page layout. Here we need to display a code snippet using a pre tag. And code
snippets can sometimes have long lines of code or comments. 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.



Blog Post Title

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Aut repellat voluptas numquam, est quia recusandae maiores quasi, cumque in vero dolor nisi accusantium nobis unde blanditiis. Nemo vero eius saepe!

Quis saepe exercitationem neque repellendus error incidunt tempora ducimus accusantium recusandae quaerat quia, nostrum natus illum. Velit ducimus quibusdam iusto aspernatur odit.

```
/* A very long comment here that should create a horizontal scrollbar when max width is reached,

.section {
  display: grid;
}
```

Nam reprehenderit voluptates perferendis impedit, perspiciatis quis, mollitia corporis debitis atque aliquid, aspernatur rerum natus ullam hic necessitatibus quae deleniti esse blanditiis.

Heading 1

Link 1
Link 2
Link 3

Solution

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

Demo Here

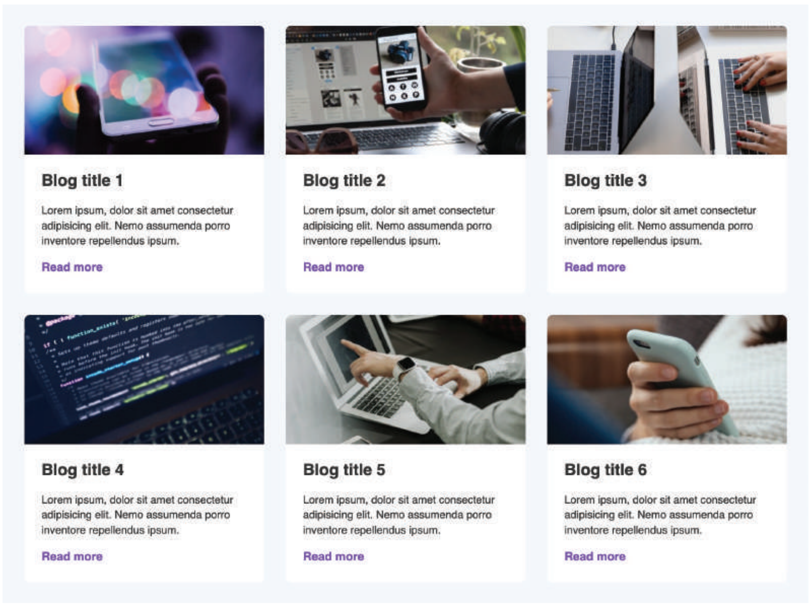
Now that the range is 0 to 1fr , it works fine. If none of this makes sense, just remember one thing - minmax(0, 1fr) is always a better option than 1fr . Which is why, if you look at the Tailwind classes grid-cols-* , their equivalent CSS values are:

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));`

Example #27-C

Responsive Grid without Media Queries

Remember how we made the blog posts display responsive by adding media queries at two breakpoints to change the number of columns? Well, we actually don't need to do that. 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.



Blog title 1

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Blog title 2

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Blog title 3

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Blog title 4

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Blog title 5

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Blog title 6

Lorem ipsum, dolor sit amet consectetur adipisicing elit. Nemo assumenda porro inventore repellendus ipsum.

Read more

Solution

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

👉 Demo Here

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, but not enough to accommodate another full column, that space will be distributed among the other columns, making sure we aren't left with any empty space at the end of the row.

Solution

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

👉 Demo Here

Understanding auto-fill

This is very similar to auto-fit . In our previous example where there are more than 5 blog posts, you will not be able to notice any difference at all. Try for yourself. Both the keywords give us the same result. But when there are fewer items and more space to fill in items:

* 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.

If this is not clear, this CSS Tricks article - auto-fill vs auto-fit explains it really well. Now use this method to make these examples responsive without using media queries:

1. Featured Logos in a Grid
2. Responsive Pricing Plans

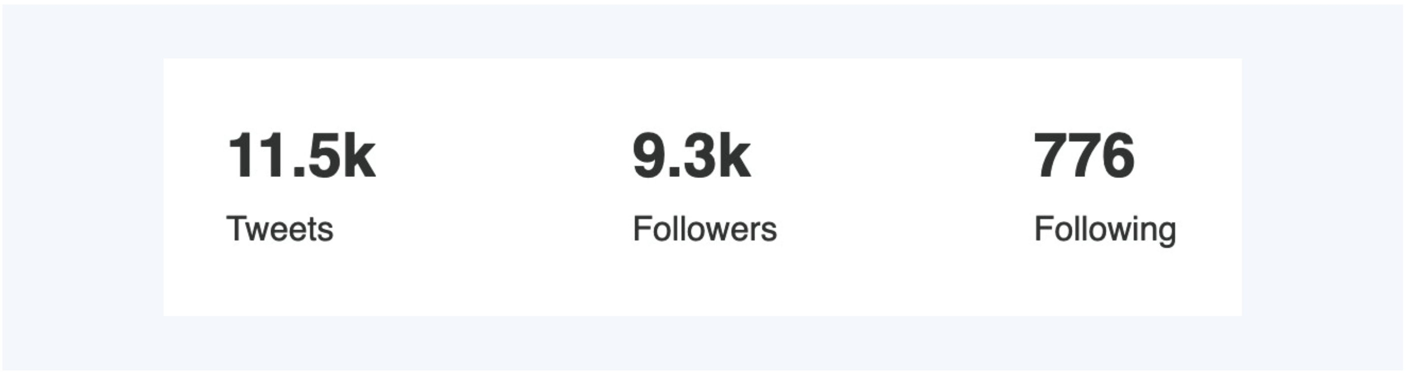
You decide whether to use **auto-fit** or auto-fill .

28. Grid Auto Flow

Example #28-A

Analytics Section

Here is a simple section that shows analytics with numbers and labels. Usually we would create 3 separate div elements for this and each div would contain a number and label. But we can avoid those additional divs using grid.



11.5k
Tweets

9.3k
Followers

776
Following

Solution

Clearly we need three columns and two rows for this. But when we create a 3 x 2 grid, items start getting placed one by one filling first row and then move to second row. We need to change this default flow, to fill the column first instead of row, by adding the grid-flow-col utility class:

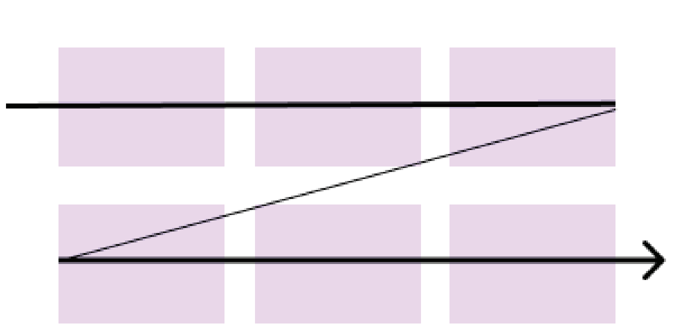
```
<section class="grid grid-rows-[auto,auto] grid-flow-col justify-between">
  ...
</section>
```

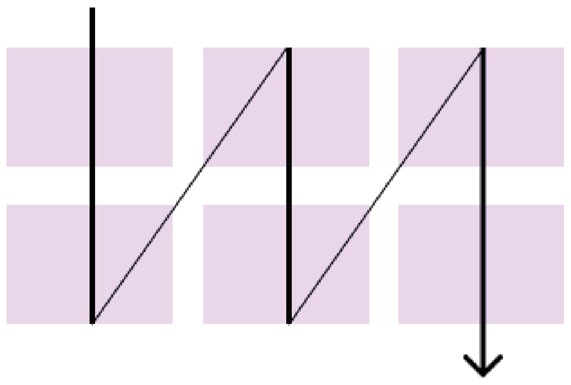
👉 Demo Here

Understanding Grid Auto Flow

The CSS property grid-auto-flow specifies the flow in which the grid items get placed into the rows and columns. By default, the flow is row . Which means, items start getting placed one by one filling first row and then keep adding more rows.

The Tailwind equivalent for this property with value row is grid-flow-row (The default) and for column is grid-flow-col (Which we used in our previous example).


grid-auto-flow: row


grid-auto-flow: column

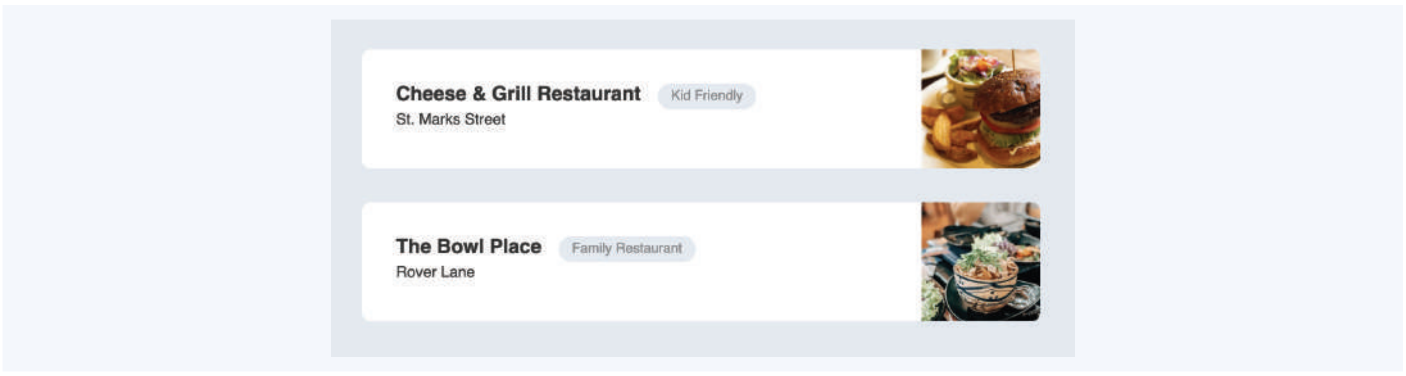
Tailwind Class	CSS Property & Value	Explanation
`grid-flow-row`	`grid-auto-flow: row;`	The items get placed one by one filling one row after the other
`grid-flow-col`	`grid-auto-flow: column;`	The items get placed one by one filling one column after the other

29. Justify Self & Align Self

Example #29-A

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.



Cheese & Grill Restaurant Kid Friendly
St. Marks Street

The Bowl Place Family Restaurant
Rover Lane

Solution

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

Demo Here

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).

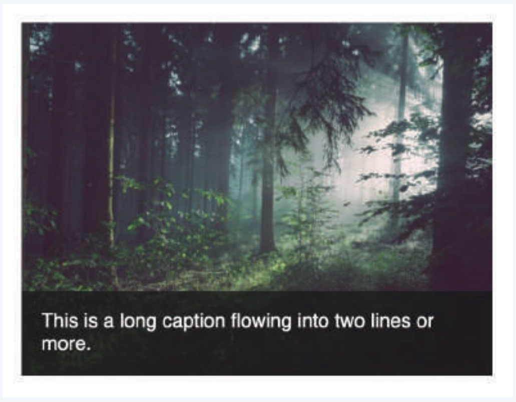
The utilities self-* is also used on a grid item. When the content of the item is shorter than the height of the row, we can use this property to control the alignment along the block axis (vertical direction).

The available utilities are similar to that of justify-items-* and items-* .

Example #29-B

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. Usually this is done with absolute positioning, but there's one problem there. When the image is too small (on mobile screen) and the caption cannot fit in the dimensions, a part of the text gets hidden. But if we implement this with grid, the image expands to fit the content within.



This is a long caption flowing into two lines or more.

Solution

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

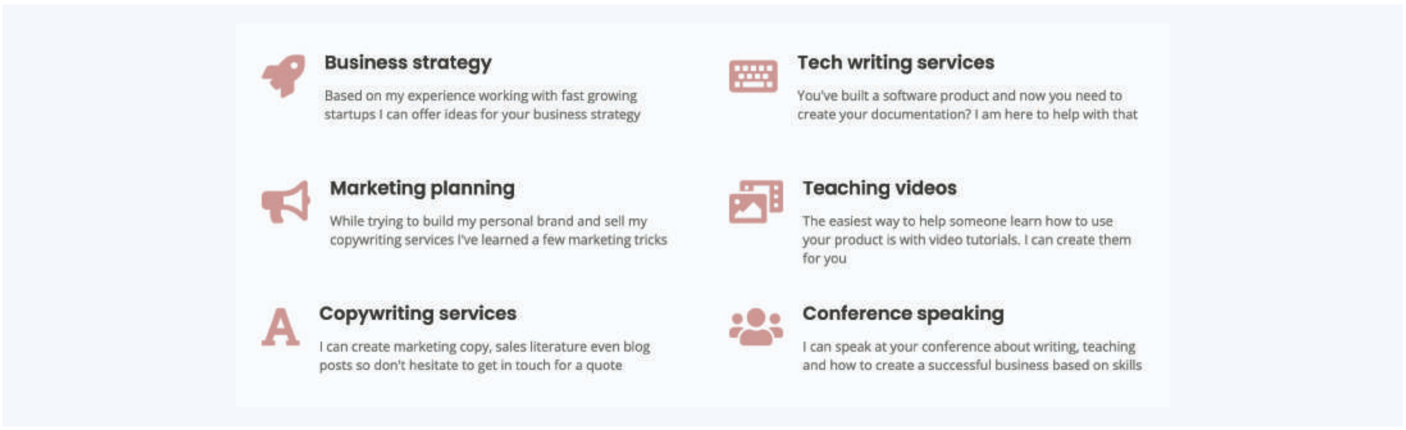
👉 Demo Here

30. Comprehensive Examples for Grid & Flexbox

Example #30-A

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 a great example of flexbox within agrid layout.



Business strategy
Based on my experience working with fast growing startups I can offer ideas for your business strategy

Marketing planning
While trying to build my personal brand and sell my copywriting services I've learned a few marketing tricks

Copywriting services
I can create marketing copy, sales literature even blog posts so don't hesitate to get in touch for a quote

Tech writing services
You've built a software product and now you need to create your documentation? I am here to help with that

Teaching videos
The easiest way to help someone learn how to use your product is with video tutorials. I can create them for you

Conference speaking
I can speak at your conference about writing, teaching and how to create a successful business based on skills

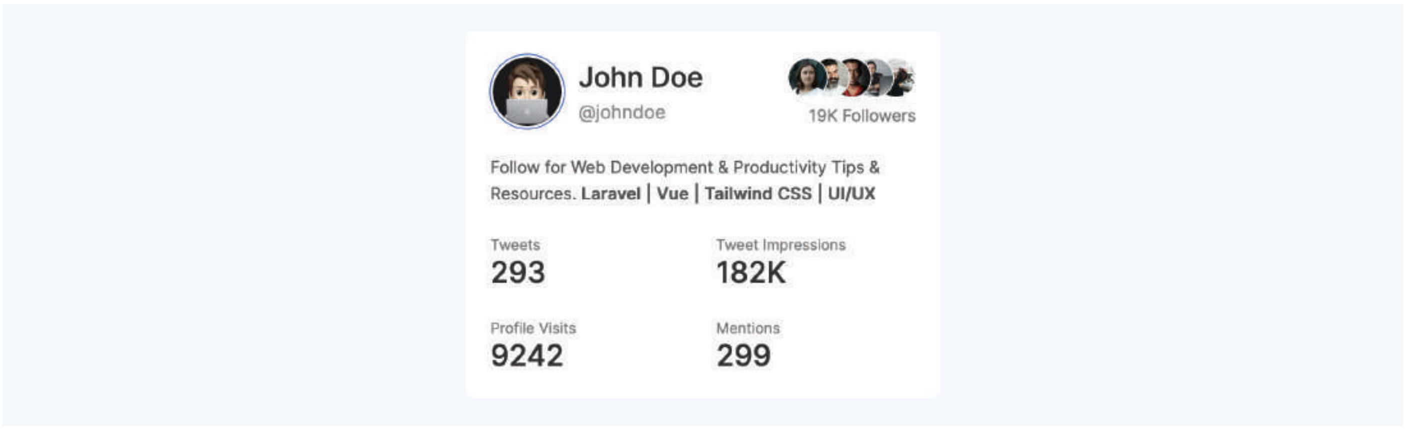
HINT : Use the utilities grid-rows-* and grid-flow-col for the grid. And make each grid item a flex container.

Demo Here

Example #30-B

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.



John Doe
@johndoe
19K Followers

Follow for Web Development & Productivity Tips & Resources. **Laravel** | **Vue** | **Tailwind CSS** | **UI/UX**

Tweets
293

Tweet Impressions
182K

Profile Visits
9242

Mentions
299

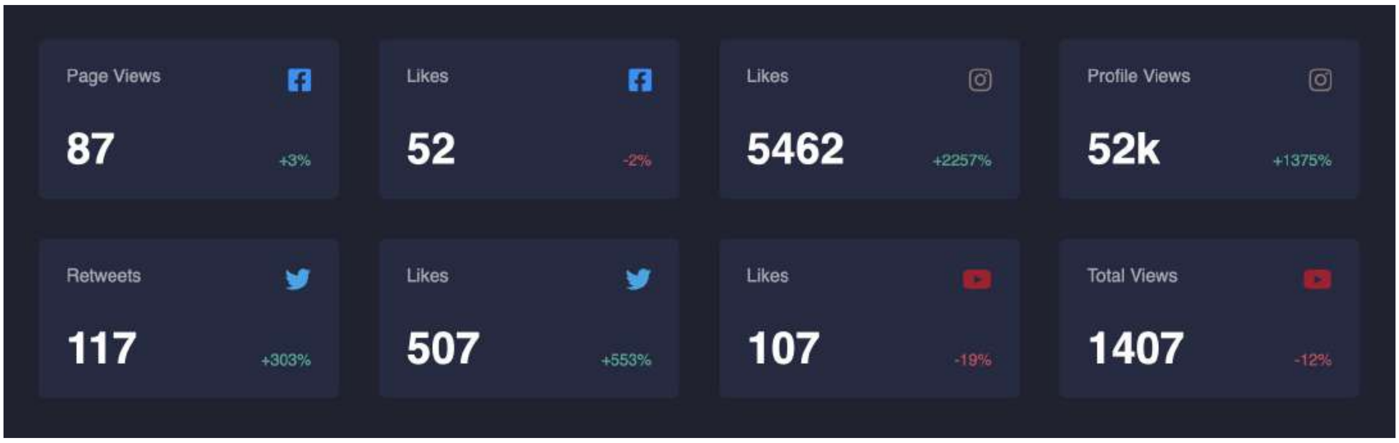
HINT : Use items-* and auto margins within flexbox. Use row-reverse direction for the followers' images. For the grid, simply use grid-col-* and gap .

Demo Here

Example #30-C

Social Media Dashboard

This example is part of the social media dashboard challenge. It's a brilliant example for grid within grid.



HINT : Use the utilities grid-cols-* , justify-* , content-* , justify-self -*, self-* and gap .

Demo Here