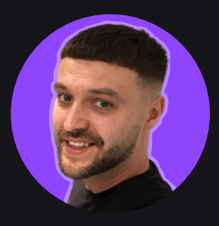
Learn CSS By Use Cases

ebook by Joe Harrison

Advanced

display: flex



Joe Harrison
@frontendjoe

display: flex

Advanced

Intro

Before the introduction of display: flex or flexbox, there were four layout modes – block, inline, table and positioned (position). On top of these modes, we used the float and clear properties to lay out our components.

Syntax

Flexbox (or the Flexible Box Layout Module) makes it easier to build website layouts. There are essentially two parts to working with flexbox – the flex container and the flex child (or item). Each part comes with it's own specific properties, most of which accept word values like center and space-between.

display: flex

Advanced

Special Power

We say flexbox is one dimensional, but there is a property called flex-wrap – which kind of makes it two dimensional too. Wrapping in flexbox can be very useful when building responsive websites.

Tips

I really like the gap property and I've used it consistently throughout my recent projects. If like me, you've probably relied on margin and padding to achieve gaps in your UI for quite a long time. The gap property will solve almost all of these problems using much less code – making your CSS cleaner in the process.

Flex Container Properties

Flex Container Properties

display: flex

Explanation

The flex container is a parent element that contains flex items (children).

Usage

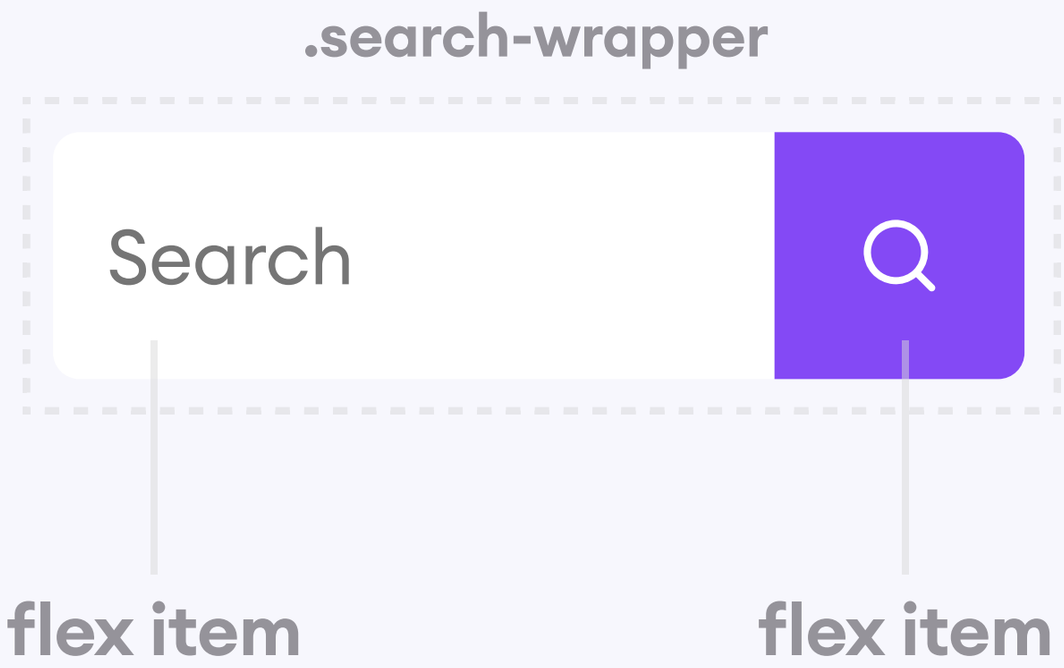
We declare a flex container by setting display: flex. This will place any elements inside, in a horizontal (row) or vertical (column) direction.

Paradigm Shift

In flexbox, the majority of positional styling for child elements is controlled by the parent. Flex items themselves do come with their own properties, which can override the parent.

display: flex

Place flex items side by side




```
.search-wrapper { display: flex; }
```

By default, **flex-direction** is **row** (horizontal)

flex-direction

Stack flex items vertically


```
.search-wrapper {
  display: flex;
  flex-direction: column;
}
```

Default value is **row** - other values include **row-reverse** and **column-reverse**

row-reverse

Useful in responsive design

Flex will only be enabled on desktop

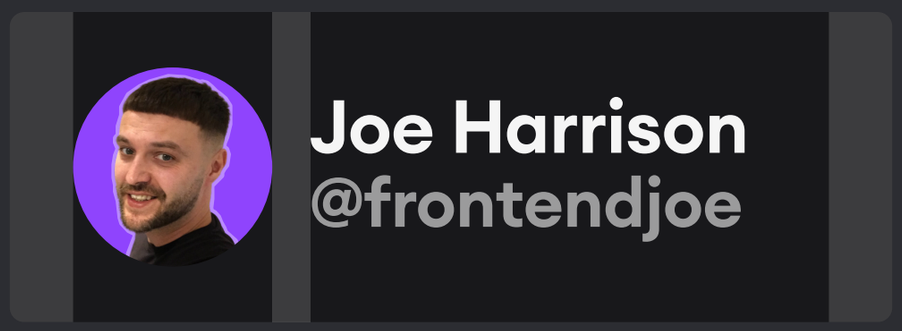
```
@media (min-width: 48rem) {
   .article {
      display: flex;
      flex-direction: row-reverse;
   }
}
```

Reverse the now flex items

Can also be achieved with the **order** flex item property

gap

Gap between flex items



gap will also work when we change **flex-direction**

```
.card {
  display: flex;
  gap: 0.75rem;
  padding: 0 1rem;
}
```

gap should be set in **rem** or **px**

gap will only add gaps in between flex items,
so we still need **padding** for outer spacing

flex-wrap

Wrap flex items

By default, items will be placed left aligned (**justify-content: flex-start**) as we'll learn next

```
.list {
  display: flex;
  flex-wrap: wrap;
  width: 25rem;
}
```

.list will require **width** if not defined by it's container

justify-content

Place flex items horizontally (x)

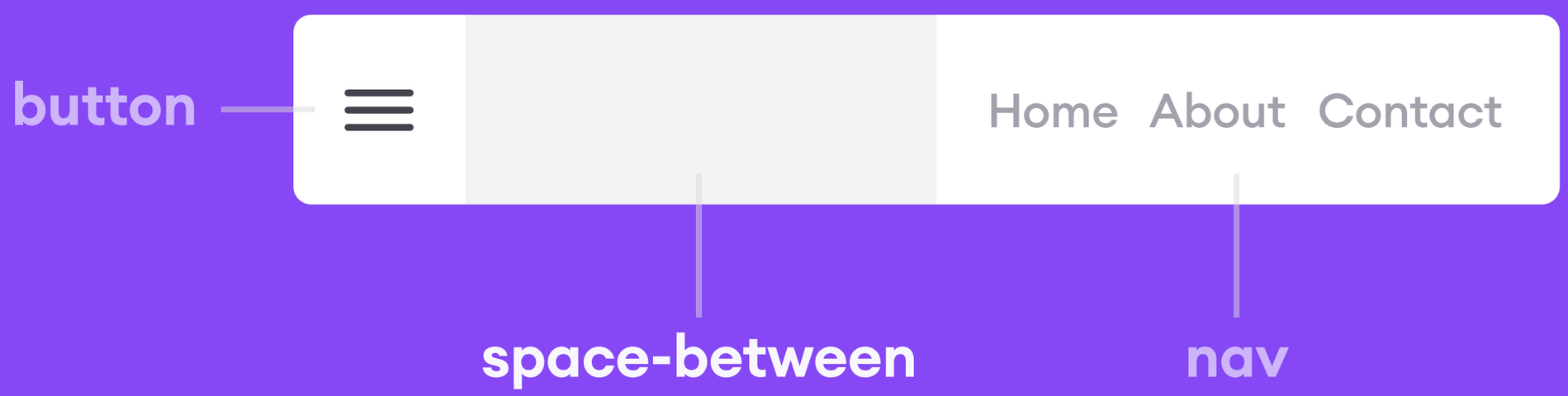
items will be placed in the **center**

```
.list {
  display: flex;
  flex-wrap: wrap;
  width: 25rem;
}
```

justify-content applies to the **X** axis

space-between

Apply space between flex items




```
.toolbar {
  display: flex;
  justify-content: space-between;
  width: 100%;
}
```

.toolbar will require **width** if not defined by it's container

space-evenly

Apply space before, between & after


```
.ratings {
  display: flex;
  justify-content: space-evenly;
  width: 16rem;
}
```

.ratings will require **width** if not defined by it’s container

align-items

Place flex items vertically (y)

flex-start

MORE INFO ›

center

MORE INFO ›

flex-end

MORE INFO ›

flex-start, **center** and **flex-end** all work with **justify-content** too

Applies to the **Y** axis

```
.button {
  display: flex;
  align-items: center;
  height: 3rem;
  padding: 0 1rem;
}
```

align-items works great with **width** and **padding**

Learn CSS By Use Cases

ebook by Joe Harrison

Flex Item Properties

Flex Item Properties

display: flex

Explanation

A flex item is a direct child element of a flex container. Therefore, any element declared immediately inside it, automatically becomes flexible or flexed.

Special Power

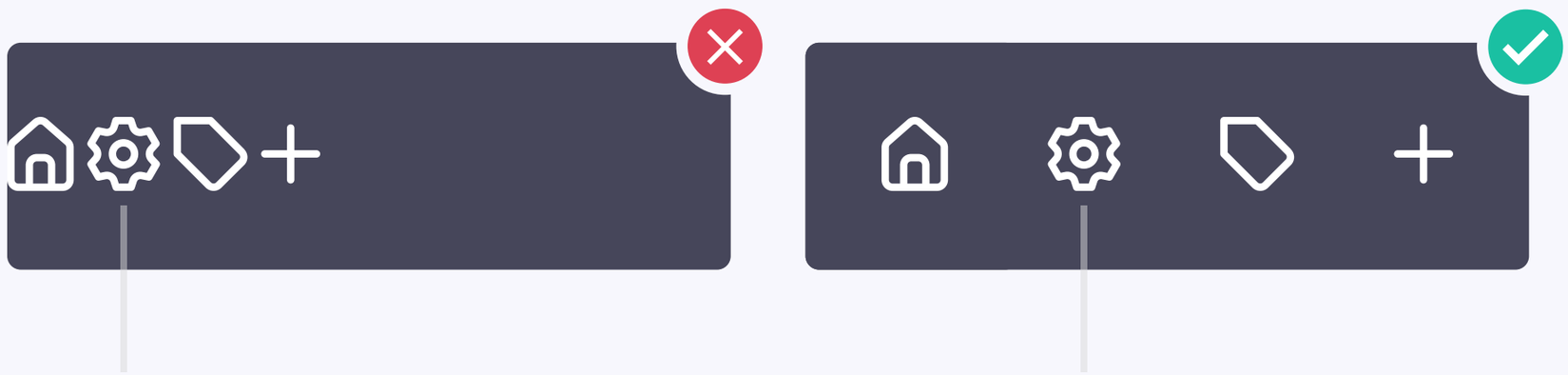
I've found that flex item properties are most useful when it comes to responsive edge cases.

Tips

I keep things simple when it comes to flex item properties. I do the majority of my child element sizing by declaring width, height and margin.

flex-grow

How elements fill space



flex-grow
default is 0

.nav-item

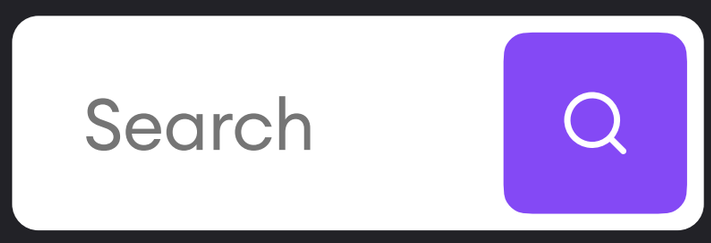
.nav requires **width** if not set by it's container

```
.nav {
  display: flex;
  width: 12rem;
}

.nav-item { flex-grow: 1; }
```

Will grow to **1 fraction** of it's container **width**

flex-shrink

Size elements proportionally



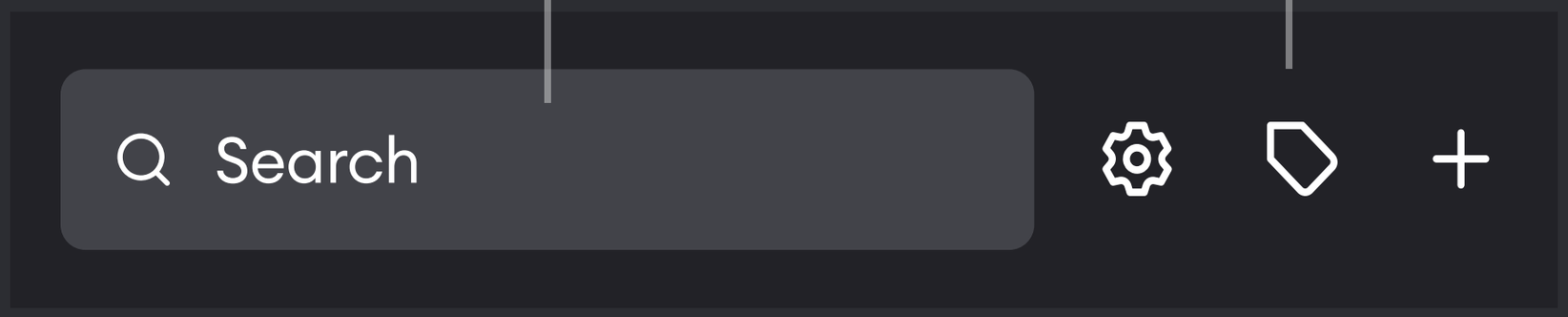

```
.wrapper { display: flex; }

.input {
  flex-grow: 2;
  flex-shrink: 2;
}

.button {
  flex-grow: 1;
  flex-shrink: 1;
}
```

For me, I find 90% of the time - the desired effect for these two properties is to always to have them the same

flex-basis

Set fixed size of a flex item

.search (will grow/shrink) **.menu** (fixed)



.nav will require **width** if not defined by it's container

```
.nav { display: flex; }

.search {
  flex-grow: 1;
  flex-shrink: 1;
}

.menu {
  flex-basis: auto;
  width: 10rem;
}
```

auto means that it will look at the element's **height** & **width** properties. We don't need to set it as it's the default value

We can add **min-** and **max-** properties to fully override the sizing of **flexbox**

flex

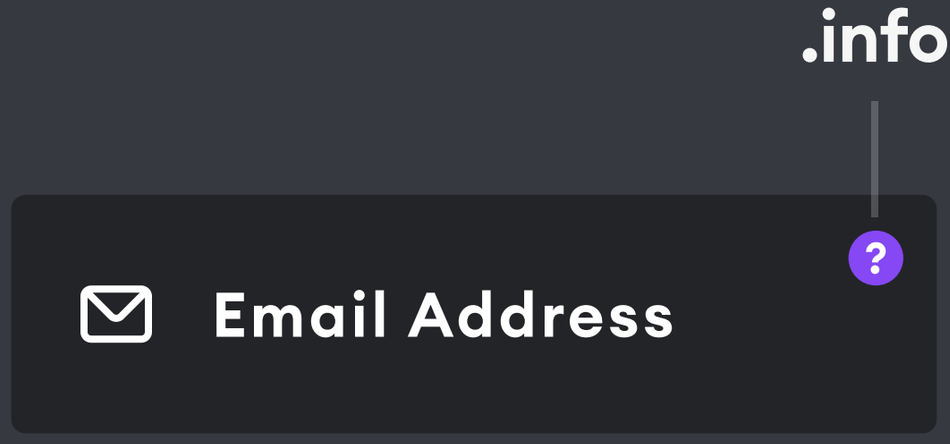
Shorthand for flex items

```
/* Longhand */
.flex-item {
  flex-grow: 8;
  flex-shrink: 4;
  flex-basis: auto;
}

/* Shorthand */
.flex-item {
  flex: 8 4 auto;
}
```

The downside to this shorthand is that we have to **set all three values** for it to work in the majority of browsers

align-self

Override container styling



Align all items vertically

Override **align-items** with own value

```
.wrapper {
  display: flex;
  align-items: center;
  height: 2.5rem;
}

.info {
  align-self: flex-start;
}
```

display: flex

Properties

Knowledge Gained

- 🏆 Flexbox makes it easier to build website layouts using rows or columns
- 🏆 A flex container is a parent element that contains flex items (children)
- 🏆 A flex item is a direct child element of a flex container
- 🏆 The majority of positional styling in flexbox is defined by the container
- 🏆 Flex items can override container settings

Joe Harrison
@frontendjoe