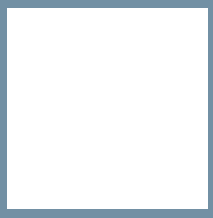
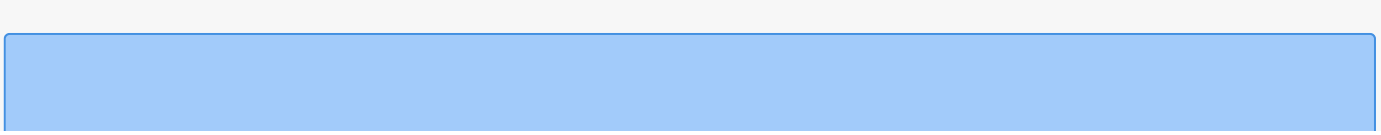
Breaking to a new row with flexbox



Here's the challenge: if you want to create a flexbox layout with several rows of items, how do you control which item ends up in which row? Presume you want to create a layout that looks something like this, with three stacked items and alternating full-width items:



A common way of controlling the positioning and size of flex items is to use `width` or `flex-basis`; if we set the fourth item to have a width of 100% it'll be positioned on its own row. But what if we don't want to or can't set the width of individual items, do we really need to? Or is there a way of just telling flexbox to line break at certain points?

There's no property that we can set on a flex item to make it break to a new row, but we can insert a collapsed row (you can think of it as a `<br>`) between two flex items two achieve something similar. In a gist:

```
/* Inserting this collapsed row between two flex items will make
 * the flex item that comes after it break to a new row */
.break {
  flex-basis: 100%;
  height: 0;
}
```

```
<div class="container">
  <div class="item"></div>
  <div class="break"></div> <!-- break -->
  <div class="item"></div>
</div>
```

Let's walk through some scenarios when you might want to use this, and look at some interesting layout techniques that it enables us to use.

Note that all of the code examples below requires and assumes that you have a flex container with `display: flex` and `flex-wrap: wrap` and that the flex items are added to that container:

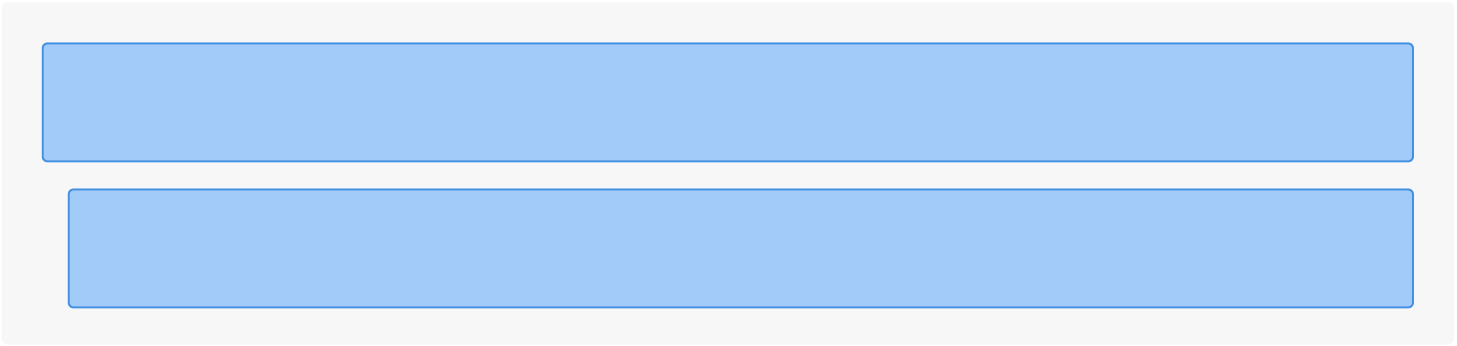
```
.container {
  display: flex;
  flex-wrap: wrap;
}
```

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

Inserting a line-breaking flex item

Using an element to break to a new flex row comes with an interesting effect: we can skip specifying the width of any item in our flex layout and rely completely on the line breaks to define the flow of our grid.

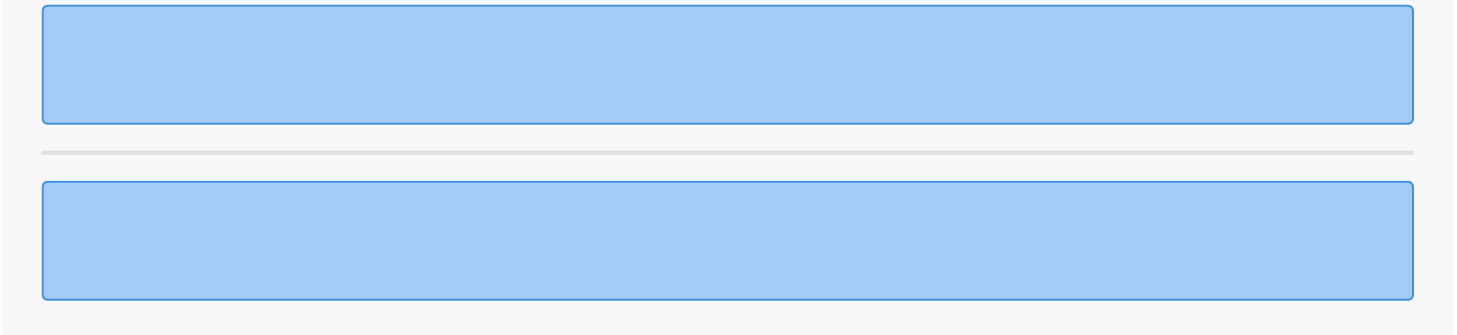
Let's start with a simple example. Say that we have two items shown side by side (these are set to grow with `flex-grow: 1`, and they have no defined `width` or `flex-basis`):



We can insert a line breaking element between the items to make them both take up 100% of the available space:

```
<div class="item">...</div>
<div class="break"></div> <!-- break to a new row -->
<div class="item">...</div>
```

This produces a layout with two vertically stacked full-width items (I've added a border to the `.break` element to illustrate its position and behavior):



How does this work? Since we've said that `.break` should take up 100% of the width of the container (because we set `flex-basis: 100%`), the breaking flex item needs to sit on its own row to accomplish that. It can't share a row with the first item so it will break to a new row, which will leave the first item alone on one row. The first item will then grow to fill the remaining space (since we set `flex-grow: 1`). The same logic applies to the second item.

We can use this technique to compose the layout at the top of the post by breaking before and after every fourth item:

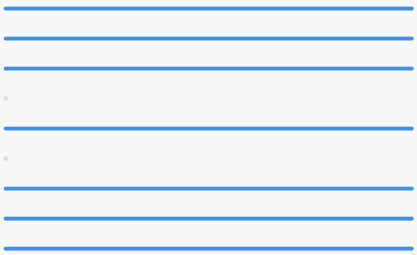
```
<div class="item">...</div>
<div class="item">...</div>
<div class="item">...</div>
<div class="break"></div> <!-- break -->
<div class="item">...</div>
<div class="break"></div> <!-- break -->
<div class="item">...</div>
<div class="item">...</div>
<div class="item">...</div>
```

This will produce the layout at the top of the blog post. Essentially an item won't break to a new row unless we insert the line-breaking element:

Again, we didn't need to specify the width on any of those items. The same technique will work for columns if

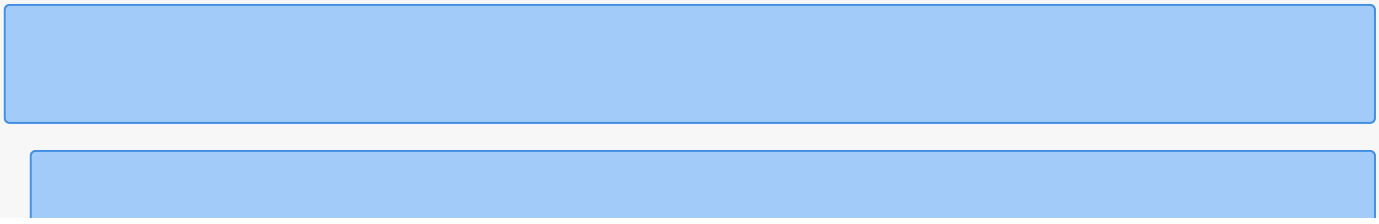
we have a flex container with `flex-direction: column`, and set the `width` (rather than `height`) to `0` for our breaking element:

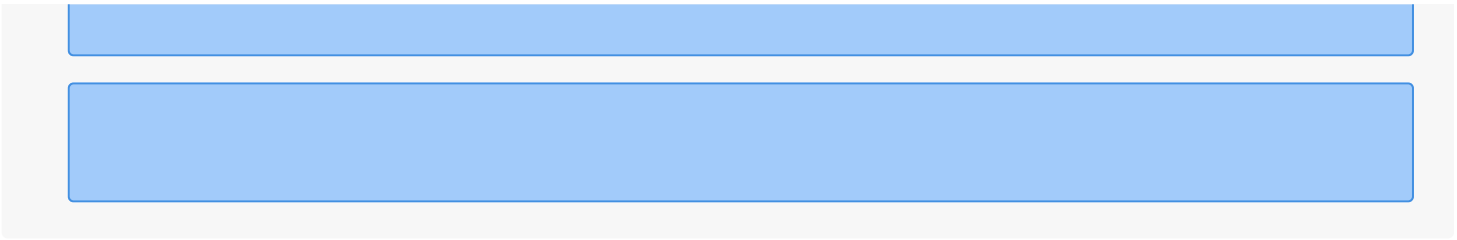
```
/* Use a collapsed column to break to a new column */
.break-column {
  flex-basis: 100%;
  width: 0;
}
```



This approach of using line breaking elements to define a layout definitely adds some bloat and noise to our HTML, but it can be a powerful tool when used in the right way. We can, for example, use it to build a masonry layout with CSS only, and position the breaks dynamically with the `order` property. We can also break to a new row without having to modify the width of any content item, and we can rely solely on `flex-grow` to distribute space in a grid layout.

Suppose that we're looking to create this layout:



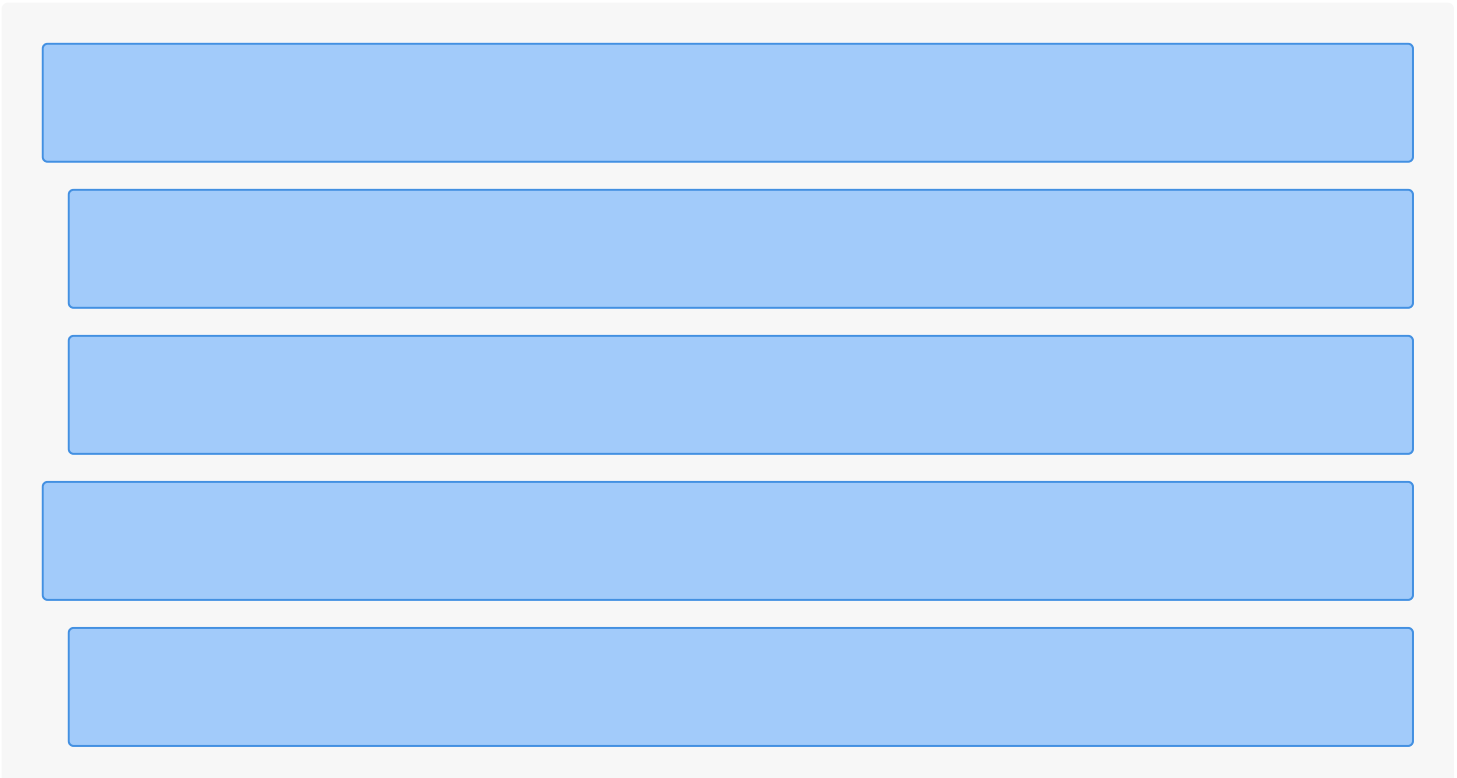


And assume that we want to do so by setting different values of `flex-grow` to distribute the space (rather than using `flex-basis` or `width`, which you'd have to recalculate as soon as you added or removed items):

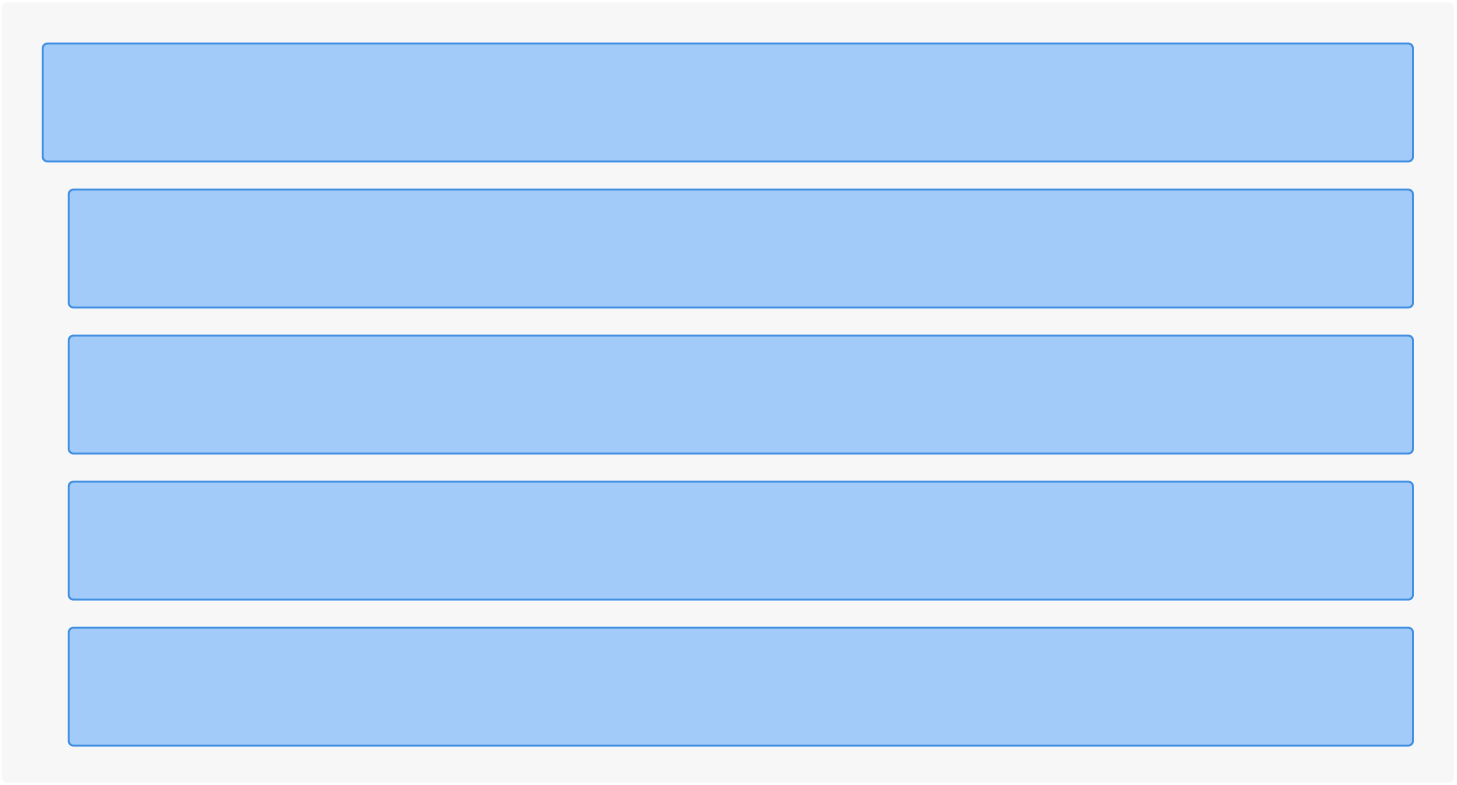
```
.item { flex-grow: 1; }
.item-wide { flex-grow: 3; }
```

```
<div class="item"></div>
<div class="item-wide"></div>
<div class="item"></div>
```

If we then want to add another row of items below that row:



We wouldn't be able to do so without resorting to setting `flex-basis` or `width` on at least some of the items (or creating a nested flexbox layout with one flex item for every row). If all of the items just have different values of `flex-grow` nothing would make them break to a new row, they'd all just squeeze in on one row together:

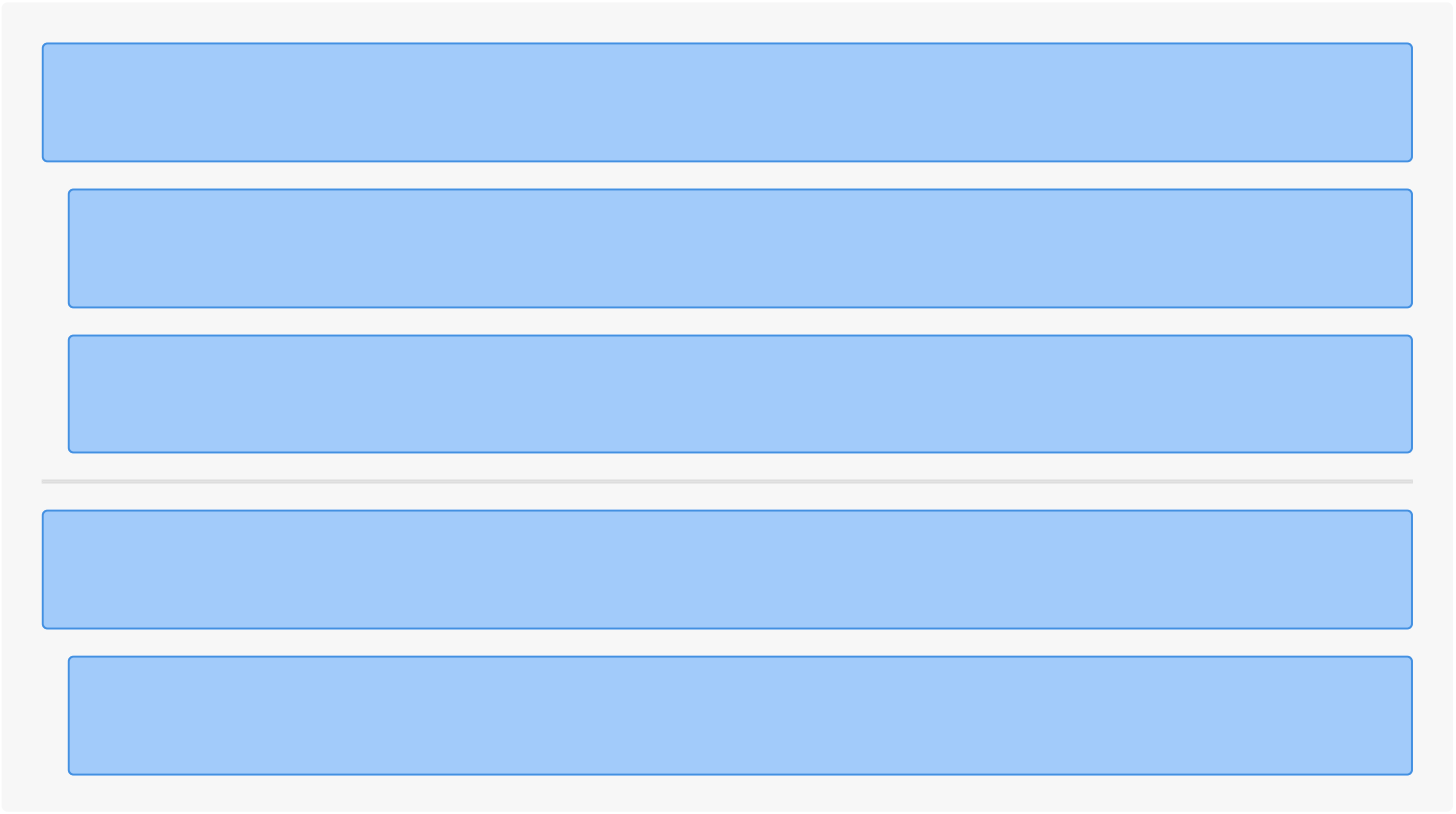


Cozy, but not what we're after. If we insert a breaking element, however, we can construct this layout by distributing all space with `flex-grow`:

```
.item { flex-grow: 1; }
.item-wide { flex-grow: 3; }
```

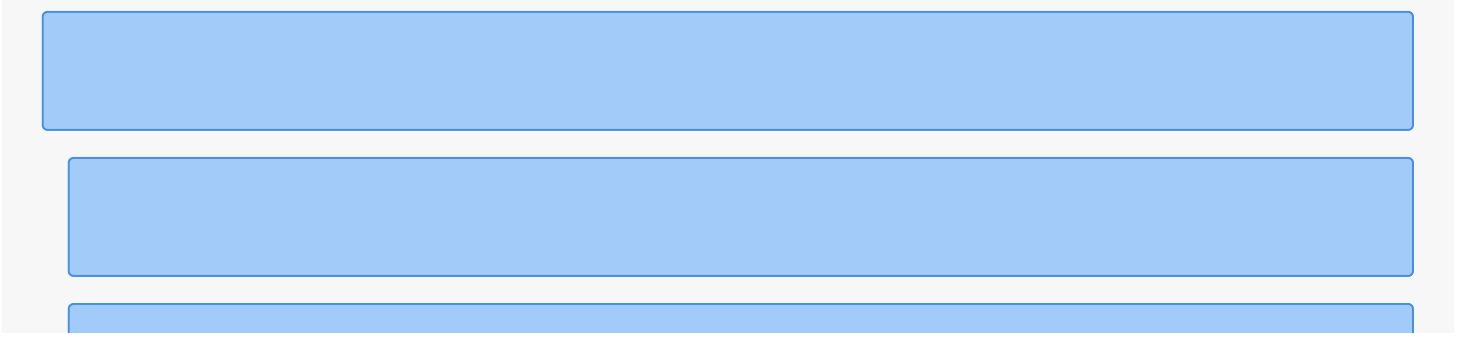
```
<div class="item"></div>
<div class="item-wide"></div>
<div class="item"></div>
<div class="break"></div> <!-- break -->
<div class="item"></div>
<div class="item"></div>
```

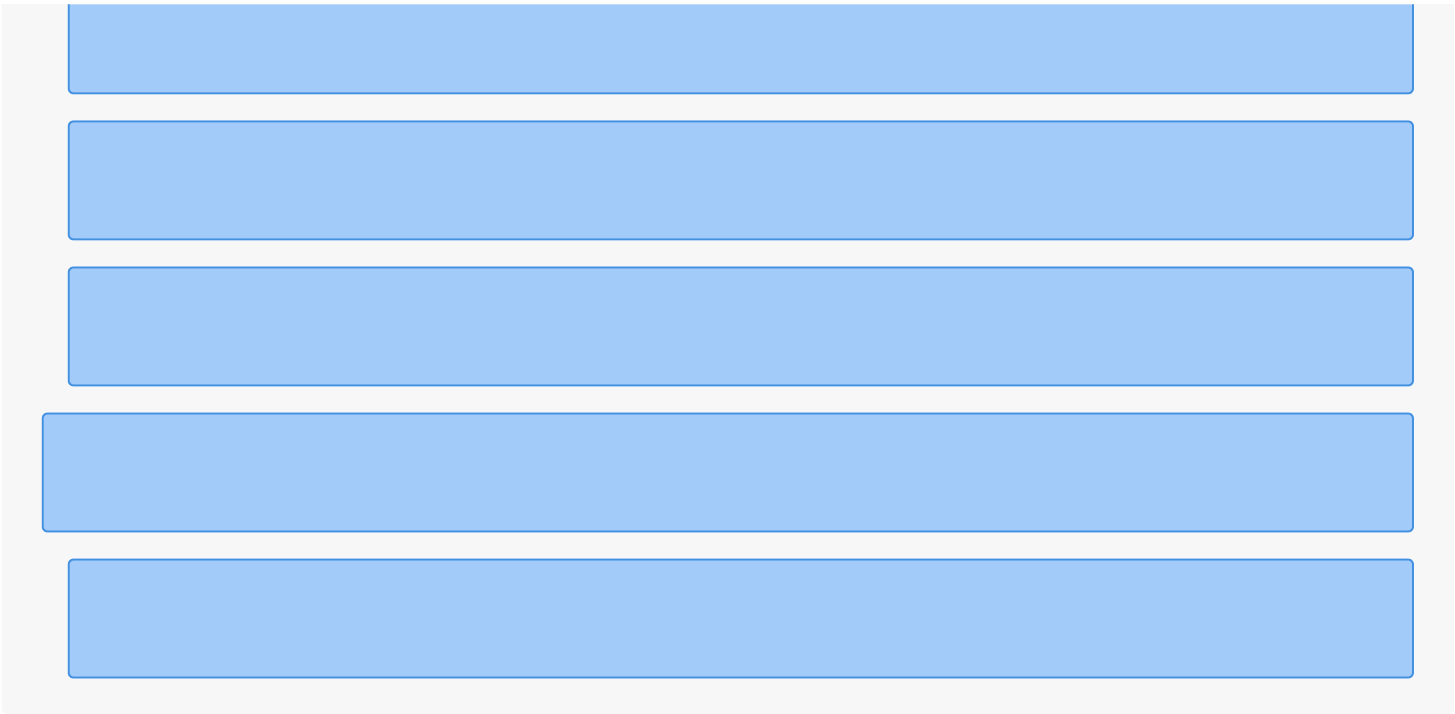
Yielding the desired layout, with all sizes defined proportionally through `flex-grow`:



And if there's a scenario when we need five items in the first row we don't have to change any of the CSS to make that work, we can just add those items before the line break:

```
<div class="item"></div>
<div class="item"></div>
<div class="item-wide"></div>
<div class="item"></div>
<div class="item"></div>
<div class="break"></div> <!-- break -->
<div class="item"></div>
<div class="item"></div>
```





All that you need to add to your CSS to use line-breaking elements are these two classes (the only difference between the two classes is that `width` (and not `height`) needs to be set to `0` for the element to collapse when used in a column layout):

```
/* Inserting a collapsed row between two flex items will make
 * the flex item that comes after it break to a new row */
.break {
  flex-basis: 100%;
  height: 0;
}

/* Use a collapsed column to break to a new column */
.break-column {
  flex-basis: 100%;
  width: 0;
}
```

You could certainly achieve the same effect or similar effects by nesting flexboxes and having one flex item for every row, and in many cases just utilizing `flex-basis`, `width`, or the content within the flex items is probably the preferred way of controlling the flow of items in a flexbox layout. But inserting line-breaking flex items is approachable and easy to grok, it works, and the technique comes with some unique characteristics that may come in handy.

PUBLISHED IN TUTORIAL • CSS | 29 APR 2019

Say hi.

hello@
tobiasahlin.com

I love to design and make things. I speak, teach, and consult at tech companies and startups, e.g. Spotify, Minecraft, Volvo, and Hyper Island. Say hi!

Tobias Bjerrome Ahlin
Stockholm, Sweden