Methods for Aligning Flex Items along the Main Axis

As stated in the question:

> To align flex items along the main axis there is one property: `justify-content`
>
> To align flex items along the cross axis there are three properties: `align-content` , `align-items` and `align-self` .

The question then asks:

> Why are there no `justify-items` and `justify-self` properties?

One answer may be: *Because they're not necessary.*

The flexbox specification provides *two* methods for aligning flex items along the main axis:

1. The `justify-content` keyword property, and
2. `auto` margins

justify-content

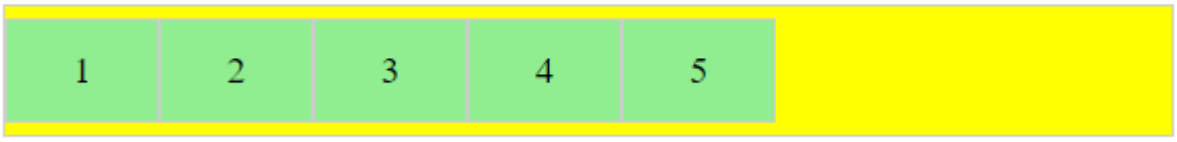
The `justify-content` property aligns flex items along the main axis of the flex container.

It is applied to the flex container but only affects flex items.

There are five alignment options:

- **`flex-start`** ~ Flex items are packed toward the start of the line.

```
justify-content: flex-start;
```



- **`flex-end`** ~ Flex items are packed toward the end of the line.

```
justify-content: flex-end;
```

- **`center`** ~ Flex items are packed toward the center of the line.

```
justify-content: center;
```

- **`space-between`** ~ Flex items are evenly spaced, with the first item aligned to one edge of the container and the last item aligned to the opposite edge. The edges used by the first and last items depends on `flex-direction` and writing mode (`ltr` or `rtl`).

```
justify-content: space-between;
```

- **`space-around`** ~ Same as `space-between` except with half-size spaces on both ends.

```
justify-content: space-around;
```

Auto Margins

With `auto` margins, flex items can be centered, spaced away or packed into sub-groups.

Unlike `justify-content` , which is applied to the flex container, `auto` margins go on flex items.

They work by consuming all free space in the specified direction.

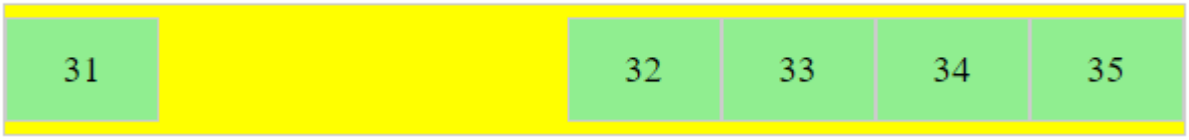
Align group of flex items to the right, but first item to the left

Scenario from the question:

> - making a group of flex items align-right (`justify-content: flex-end`) but have the first item align left (`justify-self: flex-start`)
>
>   *Consider a header section with a group of nav items and a logo. With `justify-self` the logo could be aligned left while the nav items stay far right, and the whole thing adjusts smoothly ("flexes") to different screen sizes.*

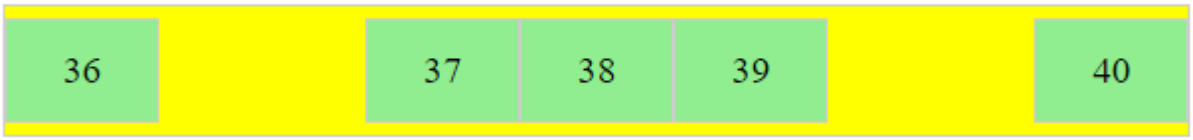
```
.box26 { margin-right: auto; }
```

```
.box32 { margin-left: auto; }
```



Other useful scenarios:

```
.box36 { margin-right: auto; } .box40 { margin-left: auto;}
```




```
.box41, .box44 { margin-left: auto; } .box42, .box45 { margin-right: auto;}
```

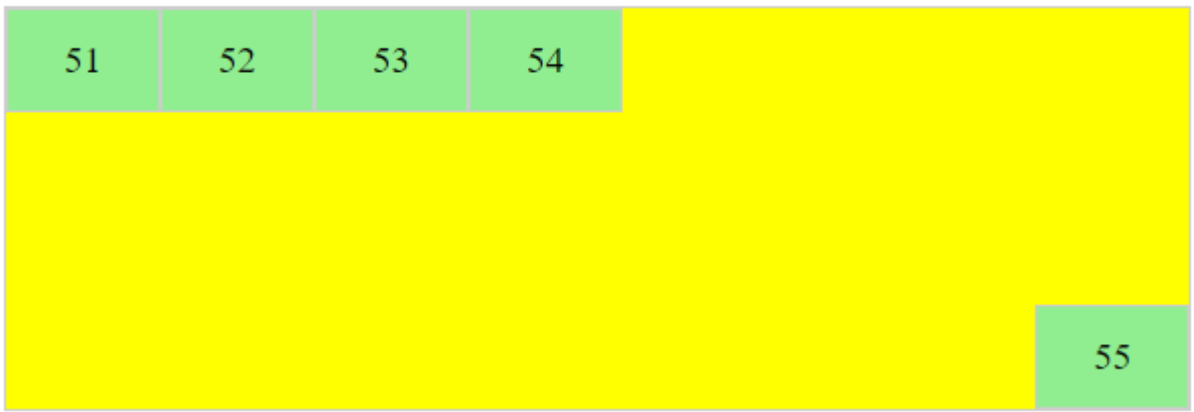

```
.box49 { margin-left: auto; }
```

Place a flex item in the corner

Scenario from the question:

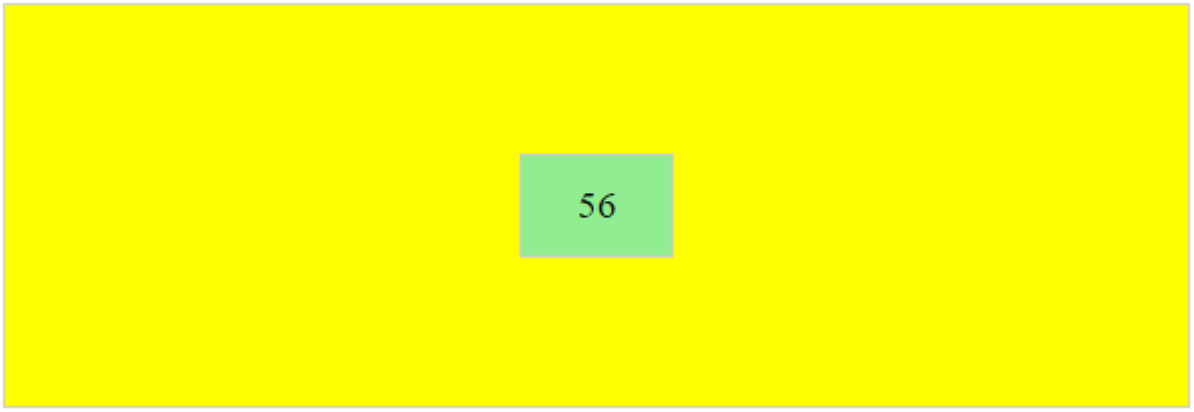
> - placing a flex item in a corner `.box { align-self: flex-end; justify-self: flex-end; }`

```
.box55 { align-self: flex-end; margin-left: auto; }
```



Center a flex item vertically and horizontally

```
.box56 { margin: auto; }
```



`margin: auto` is an alternative to `justify-content: center` and `align-items: center`.

Instead of this code on the flex container:

```
.container {
    justify-content: center;
    align-items: center;
}
```

You can use this on the flex item:

```
.box56 {
    margin: auto;
}
```

This alternative is useful when **centering a flex item that overflows the container**.

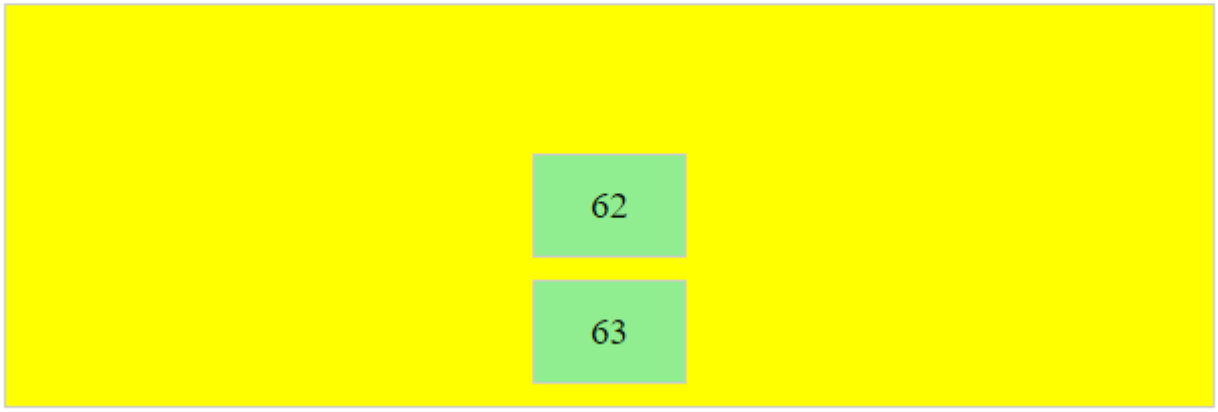
Center a flex item, and center a second flex item between the first and the edge

A flex container aligns flex items by distributing free space.

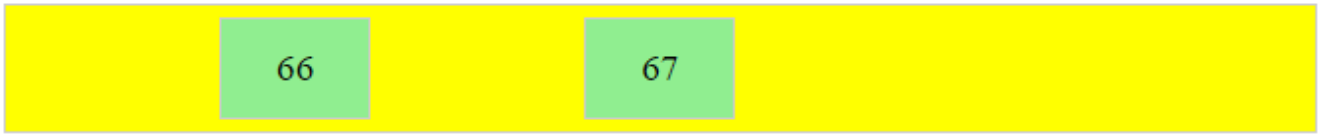
Hence, in order to create *equal balance*, so that a middle item can be centered in the container with a single item alongside, a counterbalance must be introduced.

In the examples below, invisible third flex items (boxes 61 & 68) are introduced to balance out the "real" items (box 63 & 66).

```
.container { flex-direction: column; align-items: center; }
.box61     { margin-top: auto; visibility: hidden; }
.box62     { margin-top: auto; }
.box63     { margin-top: auto; margin-bottom: auto; }
```




```
.box66     { margin-left: auto; }
.box67     { margin-left: auto; }
.box68     { margin-left: auto; margin-right: auto; visibility: hidden;  }
```



Of course, this method is nothing great in terms of semantics.

Alternatively, you can use a pseudo-element instead of an actual DOM element. Or you can use absolute positioning. All three methods are covered here: **Center and bottom-align flex items**

NOTE: The examples above will only work – in terms of true centering – when the outermost items are equal height/width. When flex items are different lengths, see next example.

Center a flex item when adjacent items vary in size

Scenario from the question:

> - in a row of three flex items, affix the middle item to the center of the container (`justify-content: center`) and align the adjacent items to the container edges (`justify-self: flex-start` and `justify-self: flex-end`).
>
>   *Note that values `space-around` and `space-between` on `justify-content` property will not keep the middle item centered in relation to the container if the adjacent items have different widths (see demo).*

As noted, unless all flex items are of equal width or height (depending on `flex-direction`), the middle item cannot be truly centered. This problem makes a strong case for a `justify-self` property (designed to handle the task, of course).

Show code snippet

Here are two methods for solving this problem:

Solution #1: Absolute Positioning

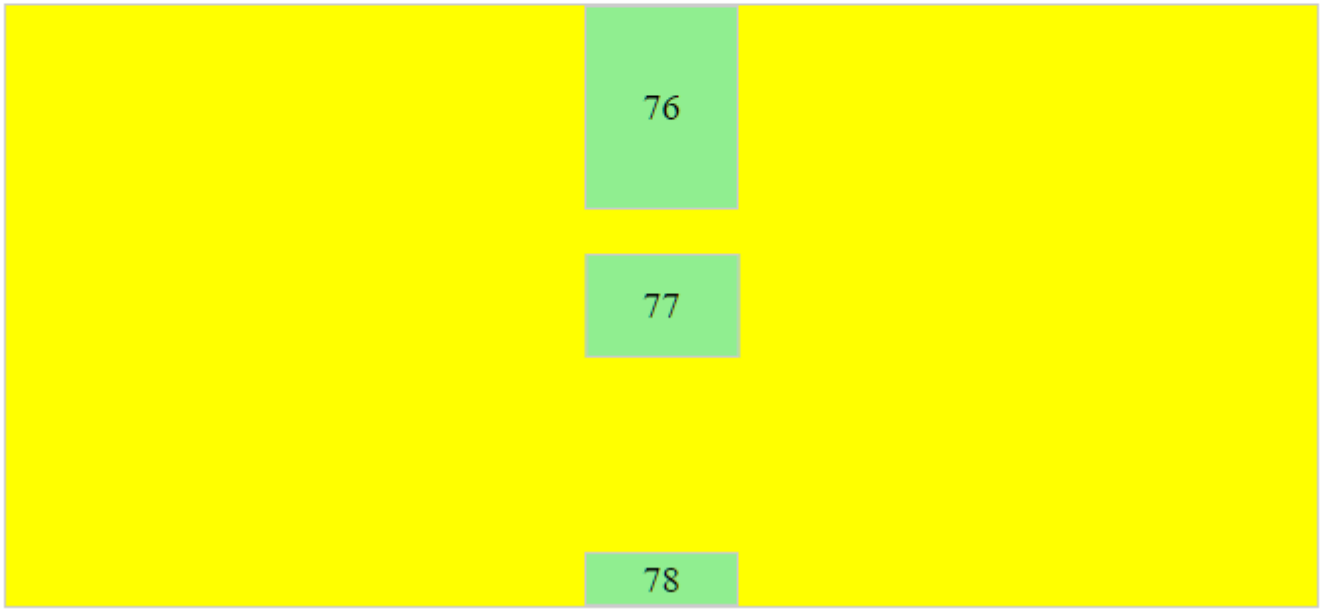
The flexbox spec allows for absolute positioning of flex items. This allows for the middle item to be perfectly centered regardless of the size of its siblings.

Just keep in mind that, like all absolutely positioned elements, the items are removed from the document flow. This means they don't take up space in the container and can overlap their siblings.

In the examples below, the middle item is centered with absolute positioning and the outer items remain in-flow. But the same layout can be achieved in reverse fashion: Center the middle item with `justify-content: center` and absolutely position the outer items.

```
.container { position: relative; }
.box71     { width: 150px; }
.box72     { position: absolute; left: 50%; transform: translate(-50%,0); }
.box73     { width: 25px; margin-left: auto; }
```


```
.container { flex-direction: column; align-items: center; position: relative; }
.box76     { height: 100px; }
.box77     { position: absolute; left: 50%; top: 50%; transform: translate(-50%,-50%); }
.box78     { height: 25px; margin-top: auto; }
```



Solution #2: Nested Flex Containers (no absolute positioning)

```
.container {
  display: flex;
}
.box {
  flex: 1;
  display: flex;
  justify-content: center;
}
.box71 > span { margin-right: auto; }
.box73 > span { margin-left: auto;  }

/* non-essential */
.box {
  align-items: center;
  border: 1px solid #ccc;
  background-color: lightgreen;
  height: 40px;
}
```

```
<div class="container">
  <div class="box box71"><span>71 short</span></div>
  <div class="box box72"><span>72 centered</span></div>
  <div class="box box73"><span>73 looooooooooooooong</span></div>
</div>
```

Run code snippet Expand snippet

Here's how it works:

- The top-level div (`.container`) is a flex container.
- Each child div (`.box`) is now a flex item.
- Each `.box` item is given `flex: 1` in order to distribute container space equally.
- Now the items are consuming all space in the row and are equal width.
- Make each item a (nested) flex container and add `justify-content: center` .
- Now each `span` element is a centered flex item.
- Use flex `auto` margins to shift the outer `span` s left and right.

You could also forgo `justify-content` and use `auto` margins exclusively.

But `justify-content` can work here because `auto` margins always have priority. From the spec:

> **8.1. Aligning with `auto` margins**
>
> Prior to alignment via `justify-content` and `align-self` , any positive free space is distributed to auto margins in that dimension.

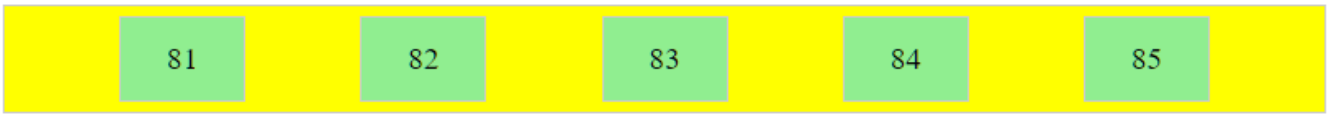
justify-content: space-same (concept)

Going back to `justify-content` for a minute, here's an idea for one more option.

- **`space-same`** ~ A hybrid of `space-between` and `space-around`. Flex items are evenly spaced (like `space-between`), except instead of half-size spaces on both ends (like `space-around`), there are full-size spaces on both ends.

This layout can be achieved with `::before` and `::after` pseudo-elements on the flex container.

```
.container          { justify-content: space-between; }
.container::before  { content: ''; }
.container::after   { content: ''; }
```



(credit: @oriol for the code, and @crl for the label)

UPDATE: Browsers have begun implementing **`space-evenly`**, which accomplishes the above. See this post for details: Equal space between flex items

PLAYGROUND (includes code for all examples above)

-- Michael_BMichael_B 176k5252 gold badges288288 silver badges400400 bronze badges