PUCD 2035
CORE 1 INTERACTION

SECTION B
T 9–11h40

SESSION SIX
OCTOBER 3, 2023

FLEXBOX

ENDLESS APPRECIATION
TO ERIC LI +
MICHAEL FEHRENBACH
FOR THE MATERIAL

DAVIS SCHERER
DSCHERER@NEWSCHOOL.EDU

AGENDA

1. HARMONIC COLLECTION HUBS
2. FLEXBOX
3. HOMEWORK
4. LECTURE @ 10H30
   ANNIKA HASTEEN-IZORA
   TISHMAN AUDITORIUM

HC HUBS

Break into groups of 4. Discuss your HC hubs as they relate to your concepts. I’ll be coming around. (15 MINS)

FLEXBOX

We’ve done basic box model positioning, but in the 2010s, **FLEXBOX** was rolled out in CSS as a display property to facilitate layouts that the box model couldn’t support.

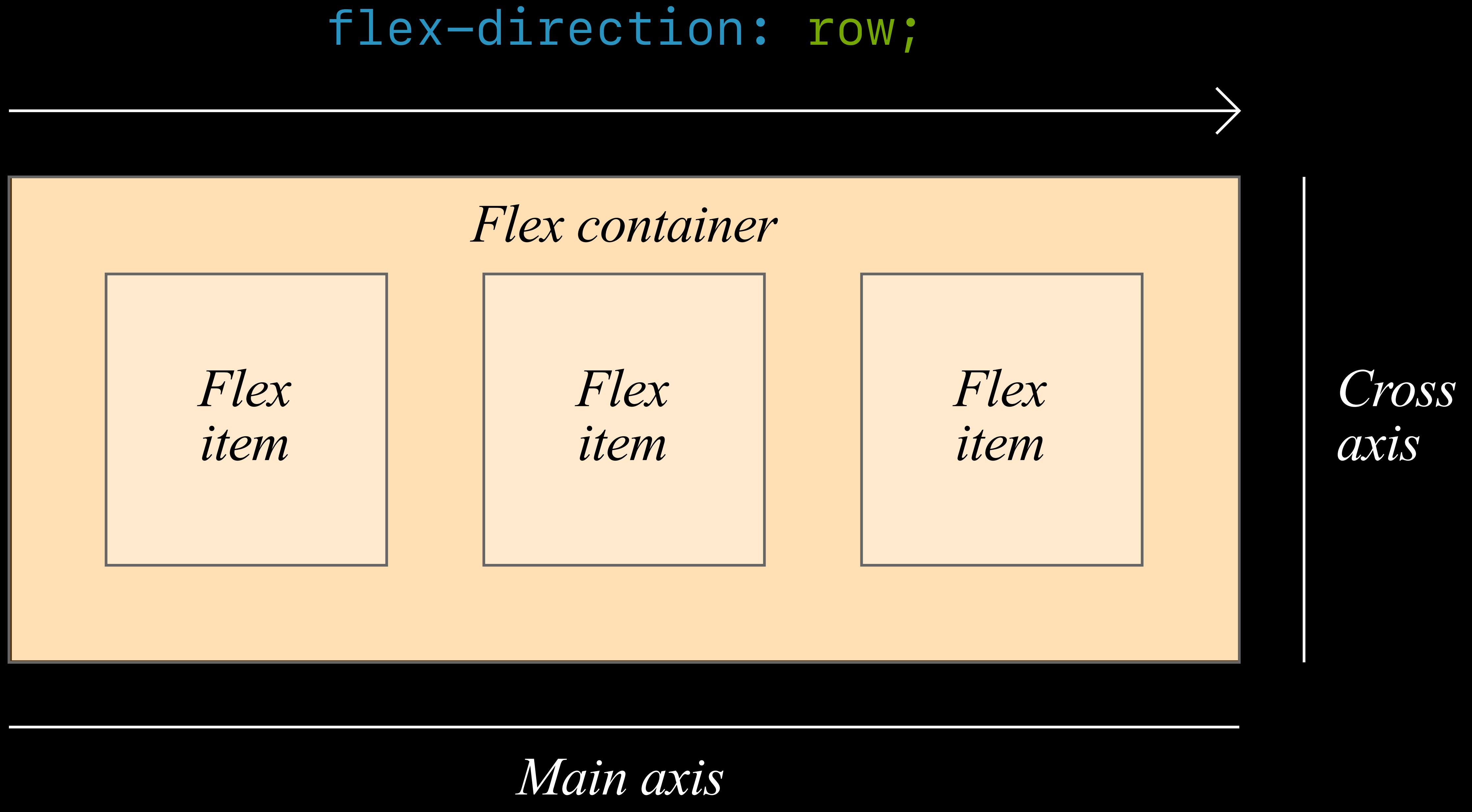
FLEXBOX

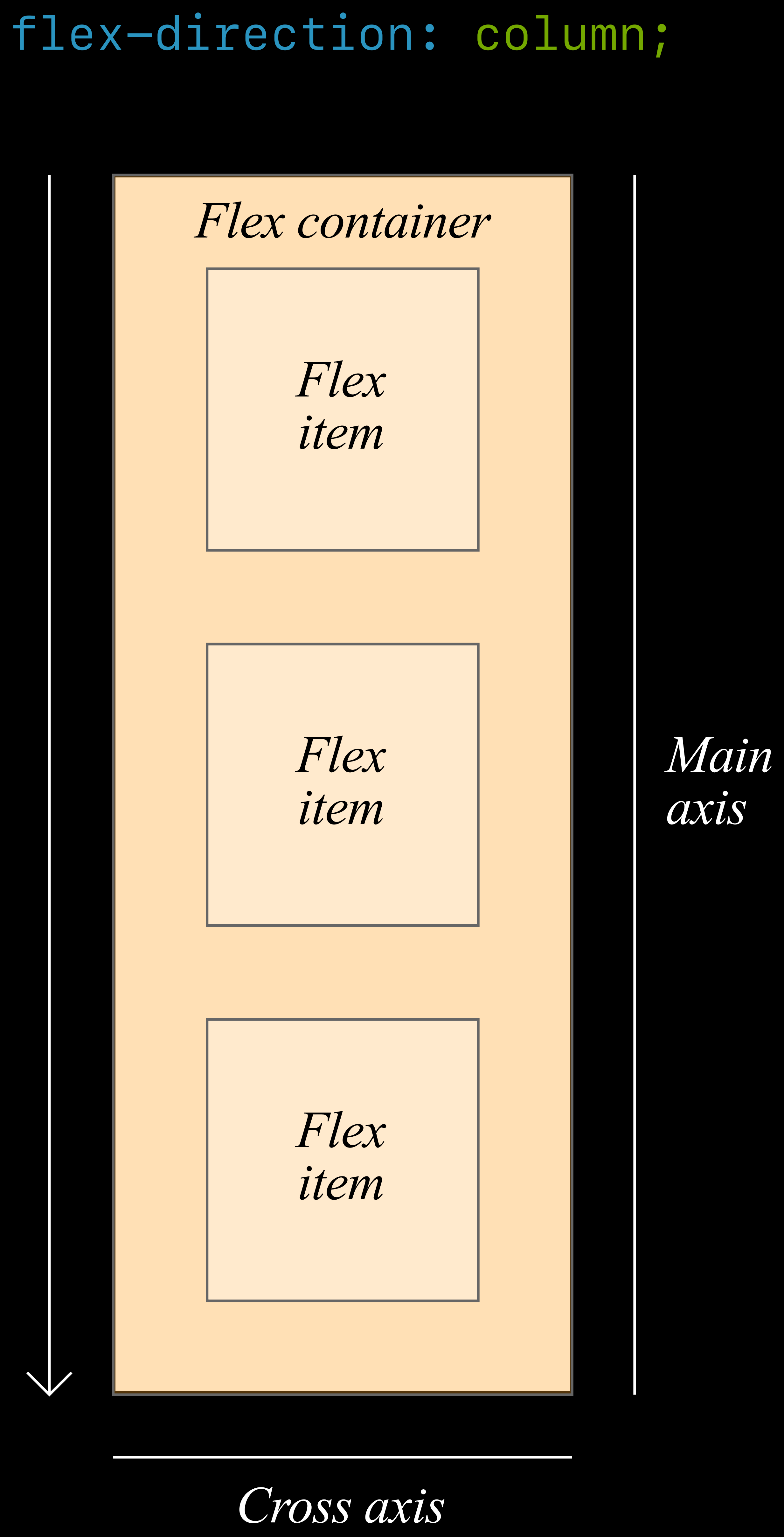
Flexbox is **one-dimensional**—meaning it typically is used to arrange things into rows or columns. These are referred to as **axes**.

FLEXBOX

The axis running in the direction of your flex items is your **main axis**, and the other axis is your **cross axis**.

MAIN VS CROSS AXIS



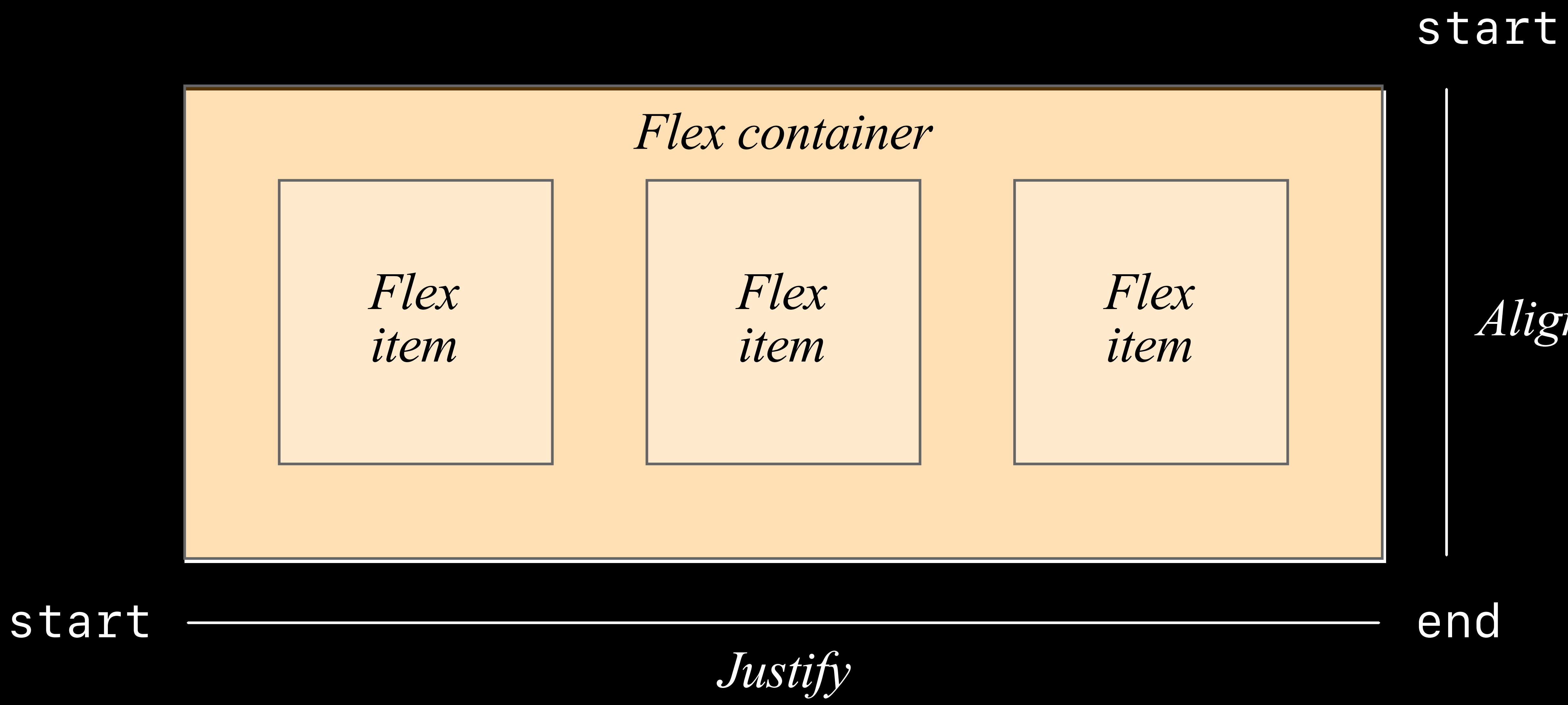


FLEXBOX

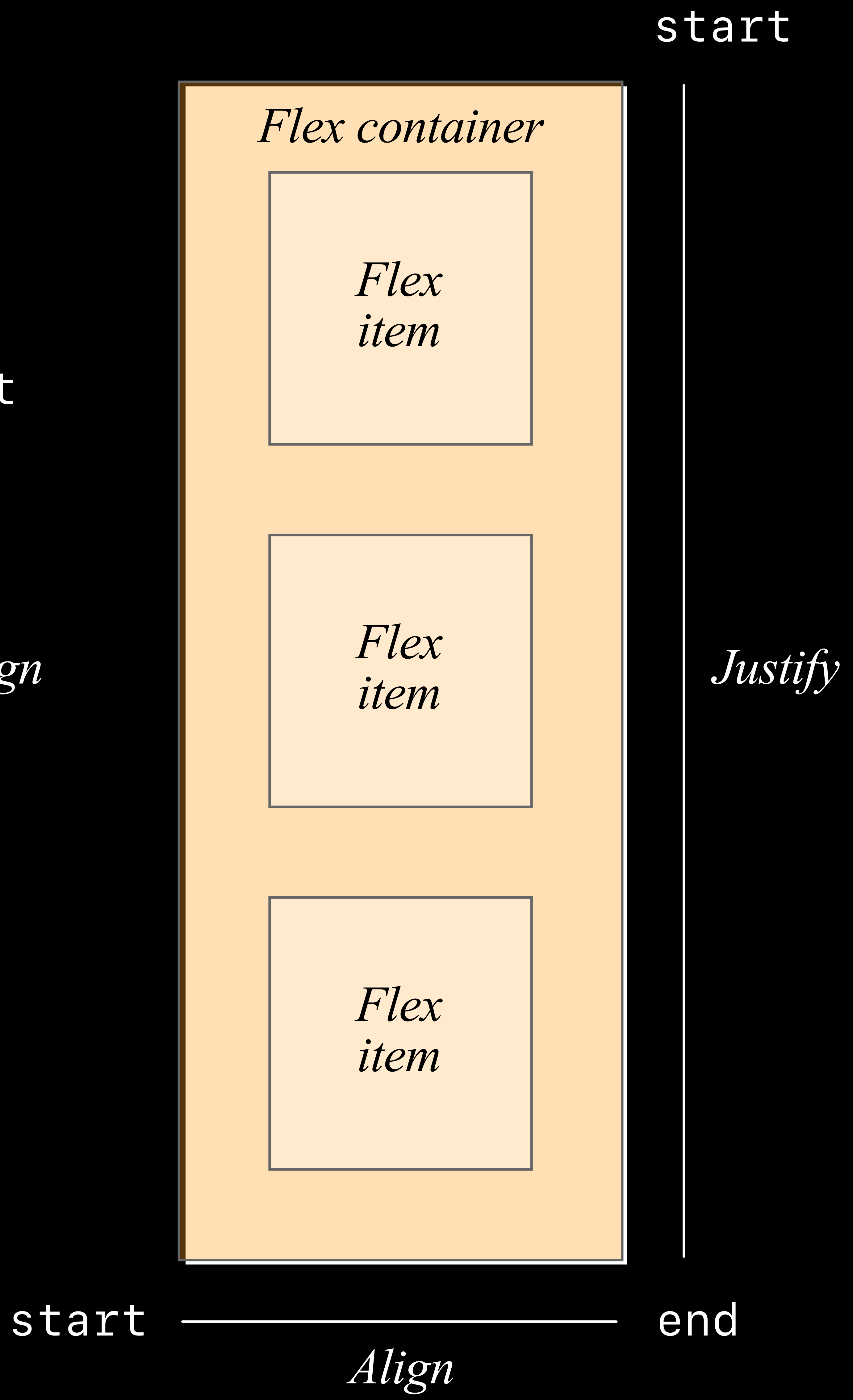
You can position elements relative to this axis (and its **start** and **end**)— flex elements are **justified** along the main axis and **aligned** along the cross axis.

START/END JUSTIFY/ALIGN

`flex-direction: row;`



`flex-direction: column;`



CONTAINER PROPERTIES

Unlike the box model, flex properties are applied to the parent, but affect the position of the children.

In other words, a `div` with `display: flex` is actually dictating the layout of the elements inside.

CONTAINER PROPERTIES

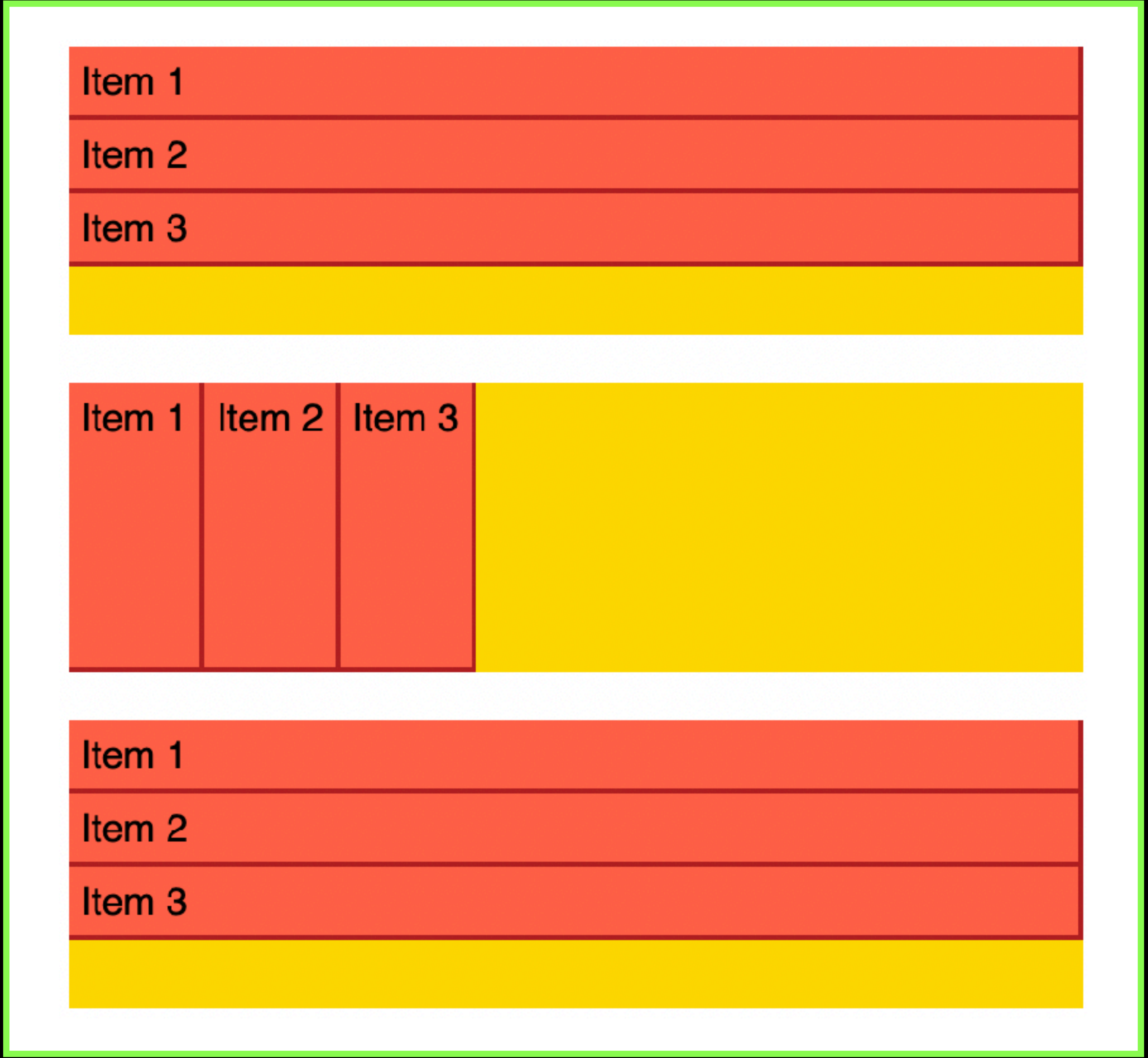
Once you set an element to flex, you set the direction of its main axis with `flex-direction`. The default (if you don’t set it) is `flex-direction: row`.

CONTAINER PROPERTIES

```html
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link href="style.css" rel="stylesheet">
</head>
<body>
    <section>
        <div>
            <p>Item 1</p>
        </div>
        <div>
            <p>Item 2</p>
        </div>
        <div>
            <p>Item 3</p>
        </div>
    </section>
    <section>
        <div>
            <p>Item 1</p>
        </div>
        <div>
            <p>Item 2</p>
        </div>
        <div>
            <p>Item 3</p>
        </div>
    </section>
    <section>
        <div>
            <p>Item 1</p>
        </div>
        <div>
            <p>Item 2</p>
        </div>
        <div>
            <p>Item 3</p>
        </div>
    </section>

</body>
</html>
```

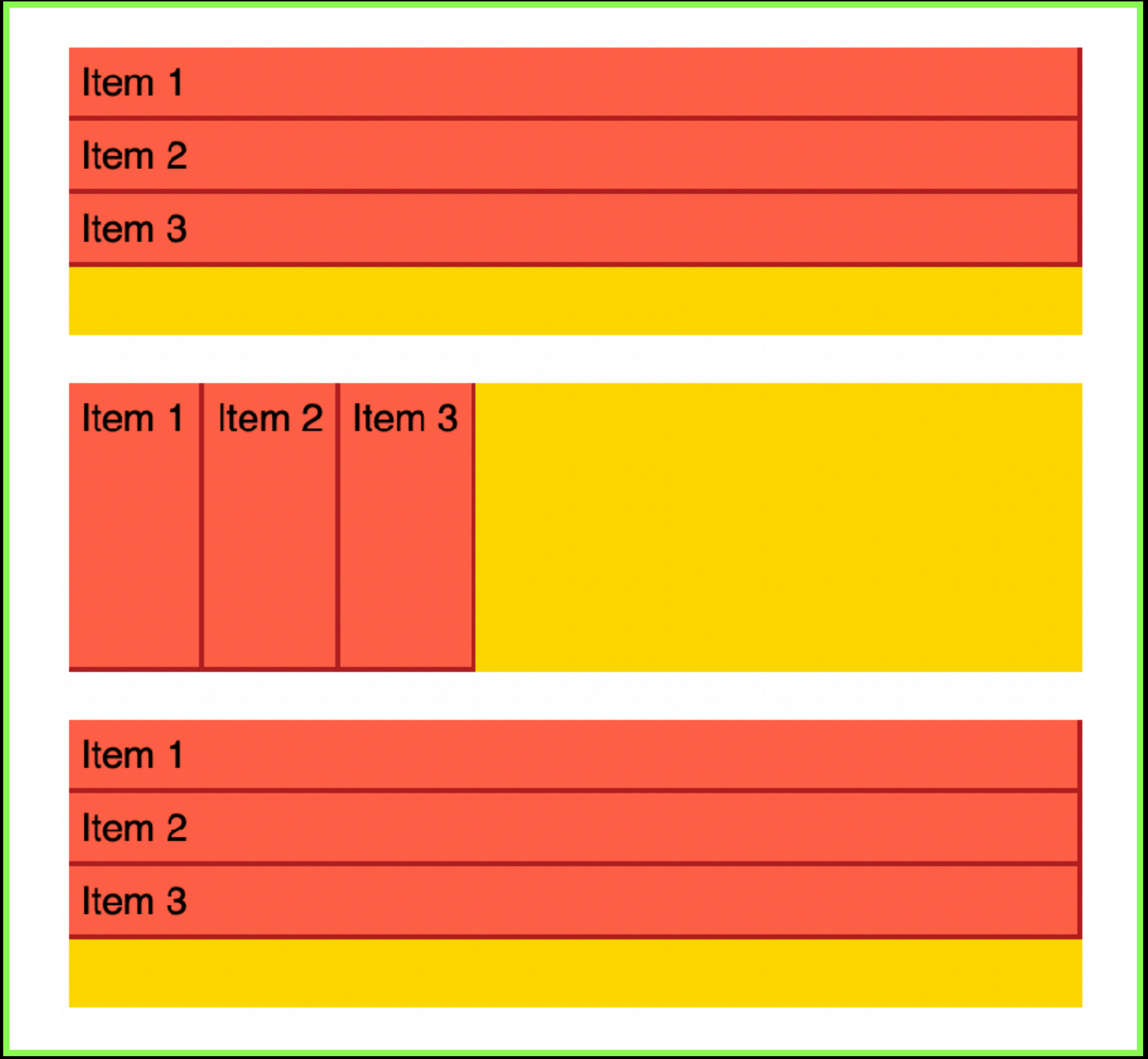
```css
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: tomato;
  border-bottom: 2px solid firebrick;
  border-right: 2px solid firebrick;
  padding: 5px;
}

section {
  background-color: gold;
  min-height: 120px;
}

section:nth-child(2),
section:nth-child(3) {
  display: flex;
  margin-top: 20px;
}

section:nth-child(3) { flex-direction: column; }
```

CONTAINER PROPERTIES



```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: tomato;
  border-bottom: 2px solid firebrick;
  border-right: 2px solid firebrick;
  padding: 5px;
}

section {
  background-color: gold;
  min-height: 120px;
}

section:nth-child(2),
section:nth-child(3) {
  display: flex;
  margin-top: 20px;
}

section:nth-child(3) { flex-direction: column; }
```

CONTAINER PROPERTIES



```
body {
  font-family: sans-serif;
  padding: 20px;
}

div {
  background-color: tomato;
  border-bottom: 2px solid firebrick;
  border-right: 2px solid firebrick;
  padding: 5px;
}

section {
  background-color: gold;
  min-height: 120px;
}

section:nth-child(2),
section:nth-child(3) {
  display: flex;
  margin-top: 20px;
}

section:nth-child(3) { flex-direction: column; }
```

block

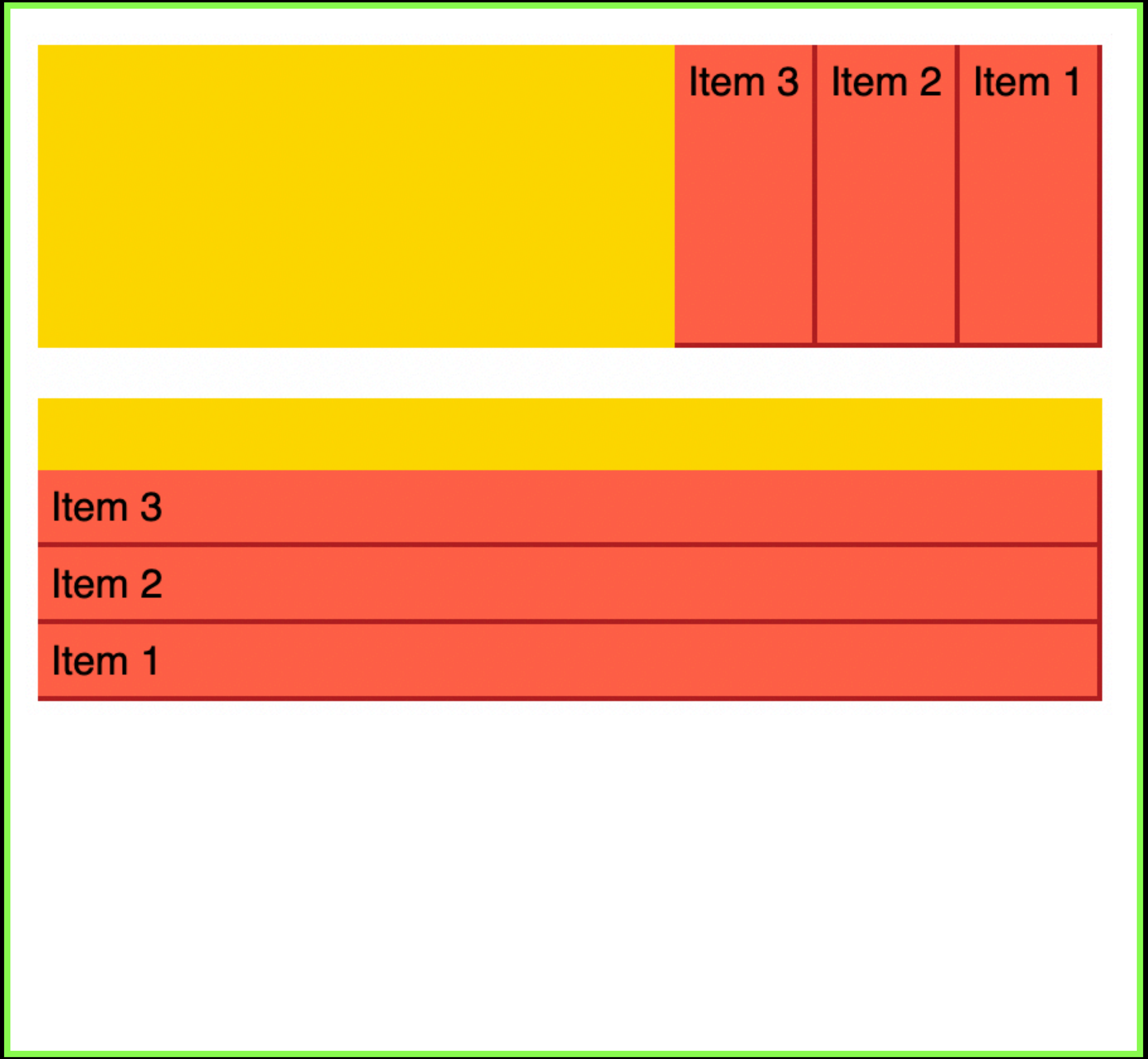
flex-direction: row (default)

flex-direction: column

CONTAINER PROPERTIES
/*REVERSE*/

You can combine a flex-direction with *reverse* to flip the start and end of the main axis.

CONTAINER PROPERTIES /*REVERSE*/



```
…

section {
  display: flex;
}

section:first-child {
  flex-direction: row-reverse;
}

section:last-child {
  flex-direction: column-reverse;
}
```

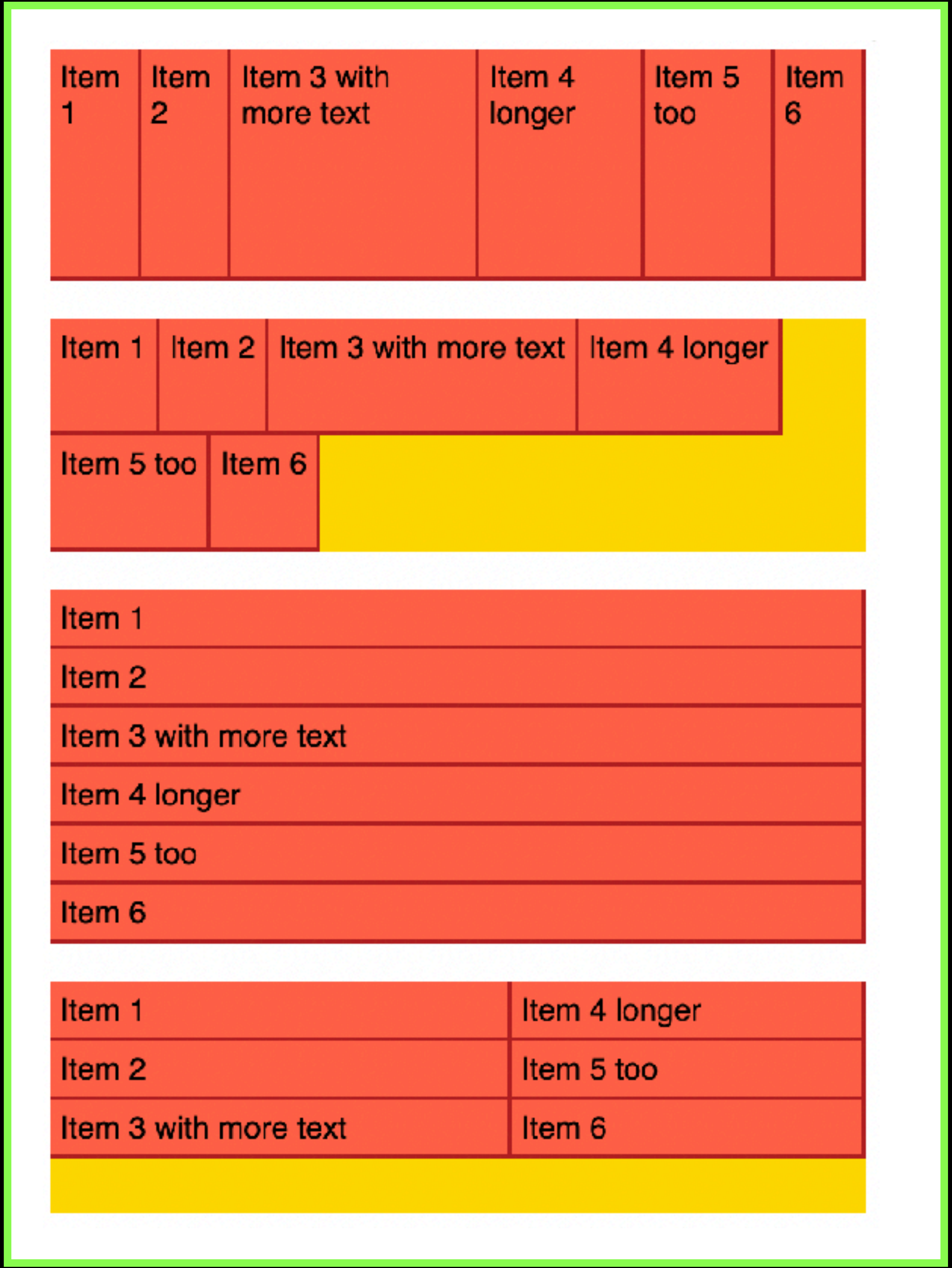
CONTAINER PROPERTIES
/*REVERSE*/

*Reordering with *reverse* does **not** change the order in the DOM—screen readers, for instance, will not reverse your sequence of elements—keep this in mind for accessibility reasons*

CONTAINER PROPERTIES
/*FLEX-WRAP*/

Remember, flex box is one dimensional. You can tell your elements to **wrap** onto a second/third line, rather than just compressing.

CONTAINER PROPERTIES /*WRAP*/



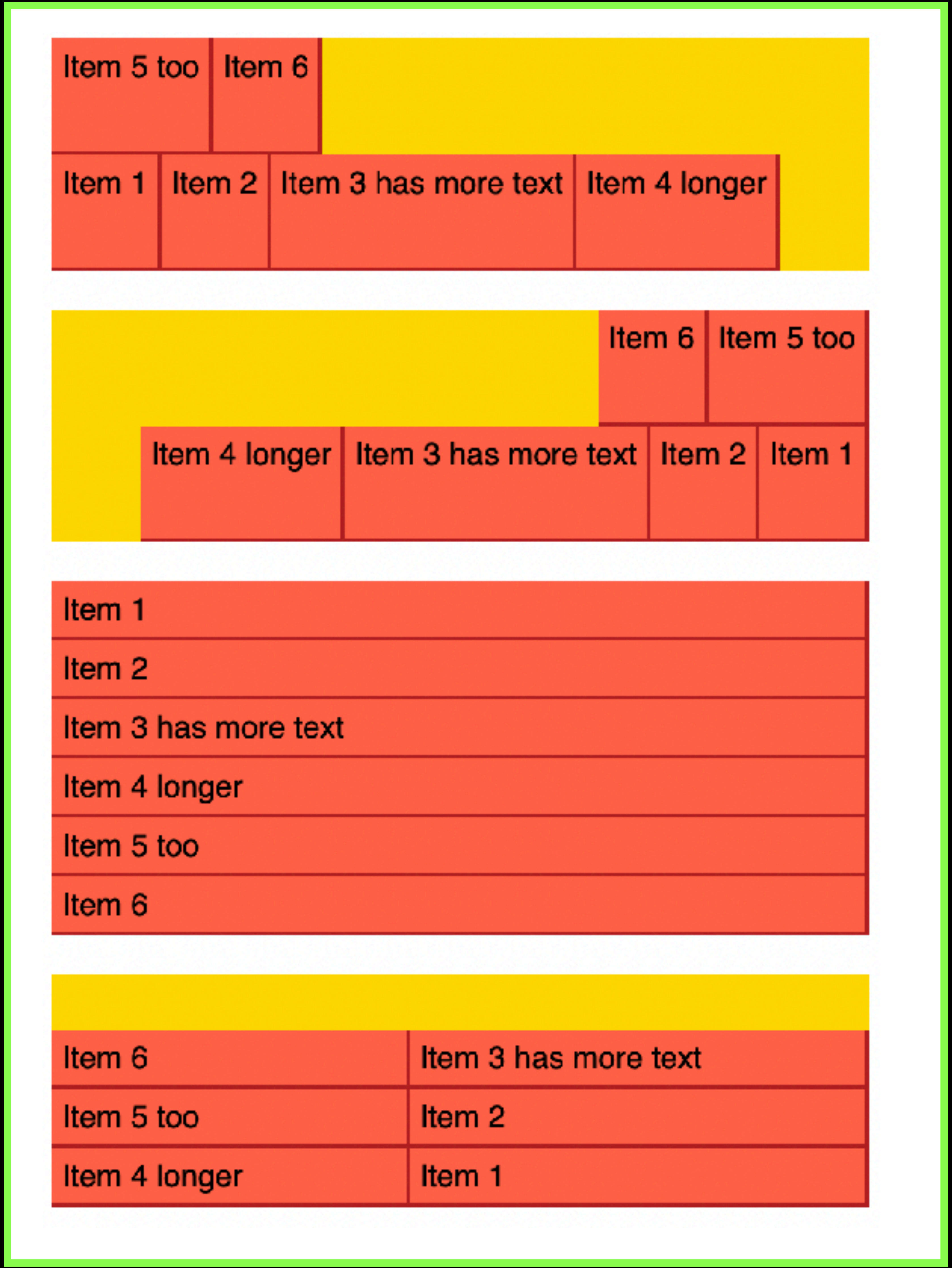
```
…

section {
  display: flex;
}

section:nth-child(1),
section:nth-child(3) {
  flex-wrap: nowrap; /* Default. */
}

section:nth-child(2),
section:nth-child(4) {
  flex-wrap: wrap;
}

section:nth-child(3),
  section:nth-child(4) {
  flex-direction: column;
}

section:nth-child(4) {
  max-height: 120px; /* Force the wrap. */
}
```

CONTAINER PROPERTIES

/*WRAP REVERSE*/

You can combine the two to wrap things from end to start.

CONTAINER PROPERTIES /*WRAP*/

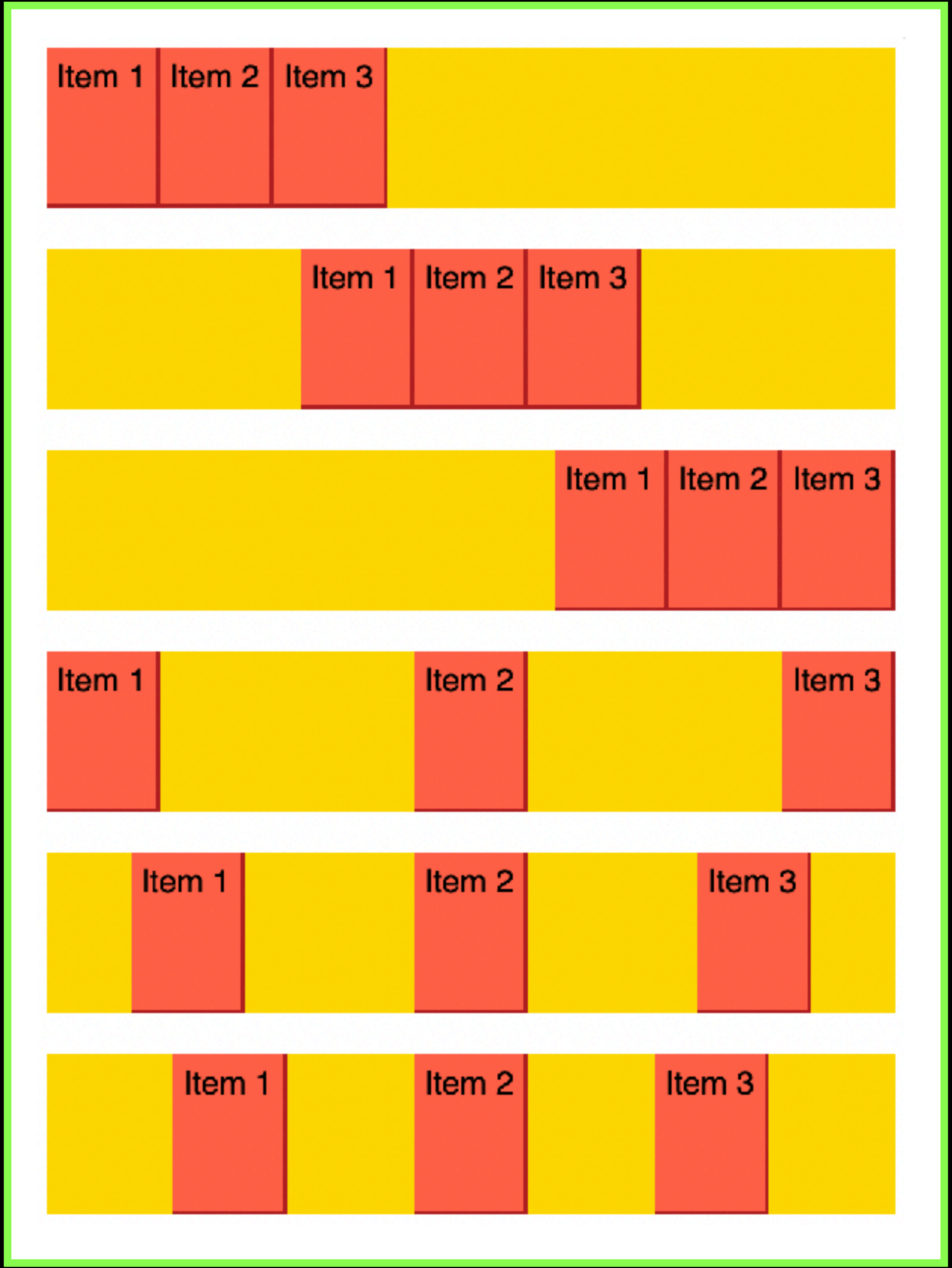


```
…

section {
  display: flex;
  flex-wrap: wrap-reverse;
}

section:nth-child(2){
  flex-direction: row-reverse;
}

section:nth-child(3){
  flex-direction: column;
}


section:nth-child(4){
  flex-direction: column-reverse;
  max-height:120px;
}
```

CONTAINER PROPERTIES
/*JUSTIFY*/

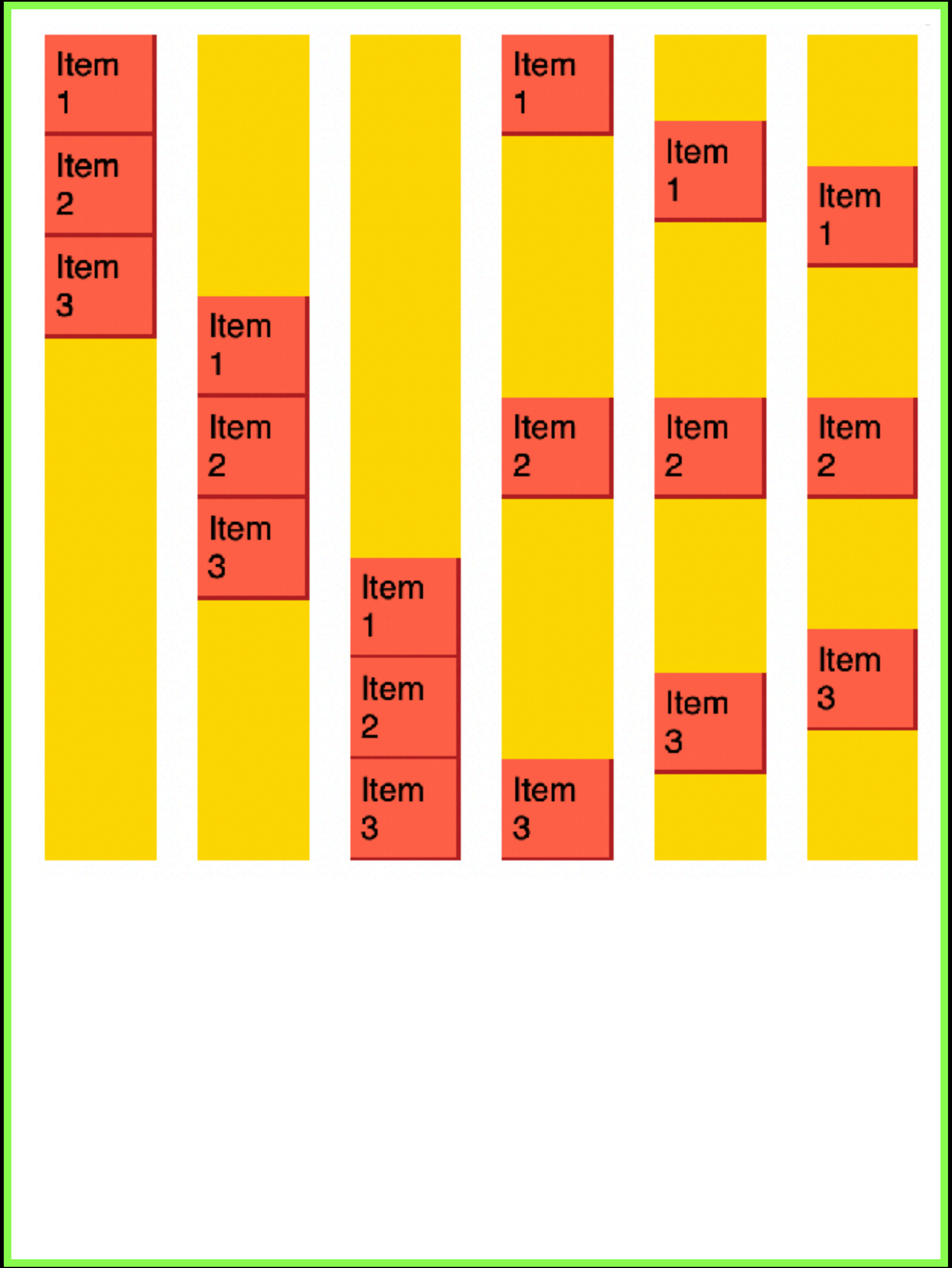
`justify-content` is where flex box really starts to shine. It automatically starts to portion off free space along the main axis.

CONTAINER PROPERTIES /*JUSTIFY*/



```
section {
   display: flex;
   height: 80px;
}

section:nth-child(1) {
   justify-content: start; /* Default. */
}

section:nth-child(2) {
   justify-content: center;
}
section:nth-child(3) {
   justify-content: end;
}
section:nth-child(4) {
   justify-content: space-between;
}

section:nth-child(5) {
   justify-content: space-between;
}

section:nth-child(6) {
   justify-content: space-evenly;
}
```

CONTAINER PROPERTIES /*JUSTIFY*/



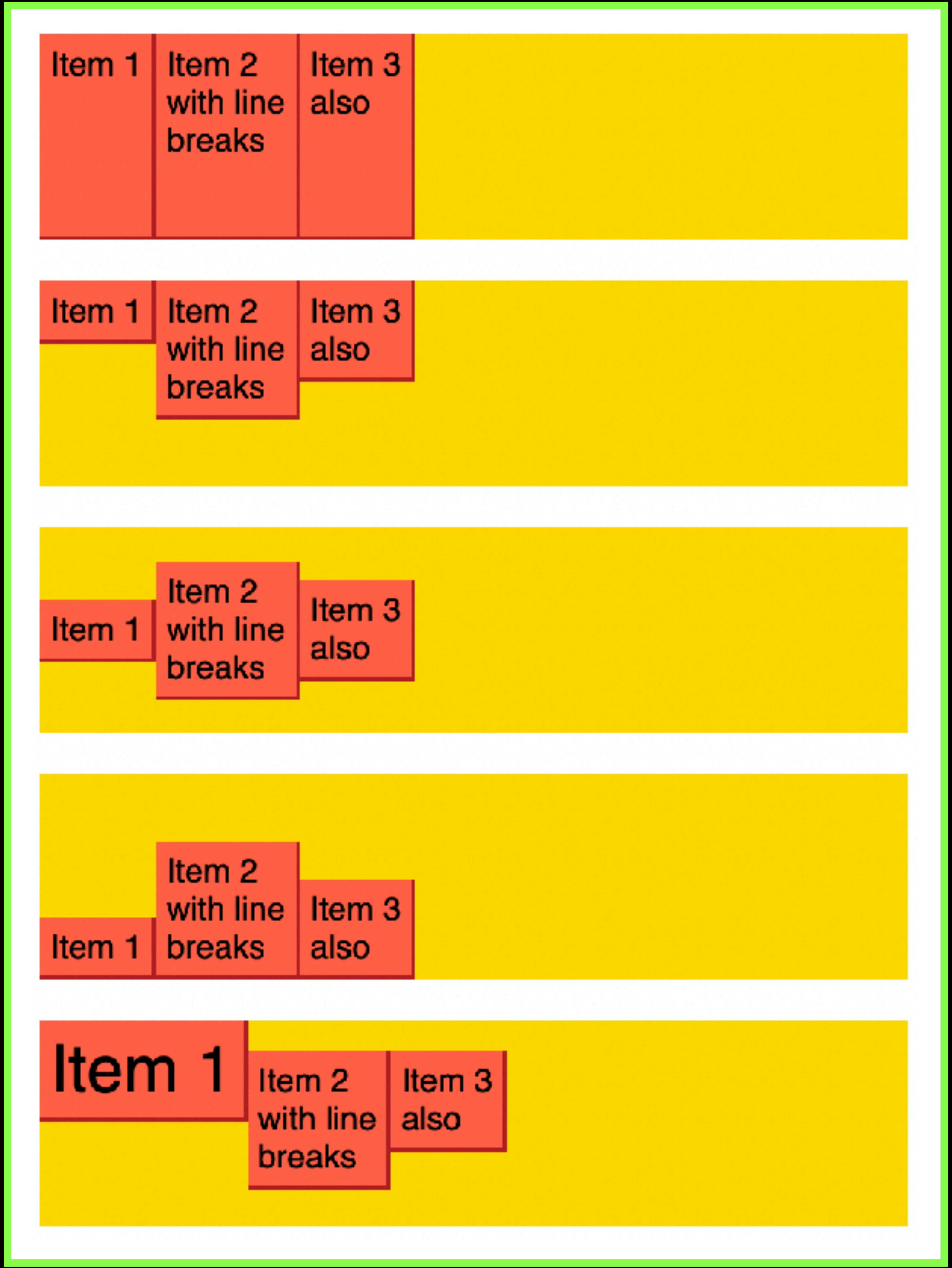
```
body {
   display: flex; /* `section` in a row! */
   column-gap: 20px; /* Will talk about ↓. */
}

section {
   display: flex;
   flex-direction: column; /* Vertical! */
   height: 400px; /* Force a height. */
}

section:nth-child(1) {
   justify-content: start; /* Default. */
}

section:nth-child(2) {
   justify-content: center;
}
section:nth-child(3) {
   justify-content: end;
}
section:nth-child(4) {
   justify-content: space-between;
}

section:nth-child(5) {
   justify-content: space-between;
}

section:nth-child(6) {
   justify-content: space-evenly;
}
```

CONTAINER PROPERTIES

/*ALIGN ITEMS*/

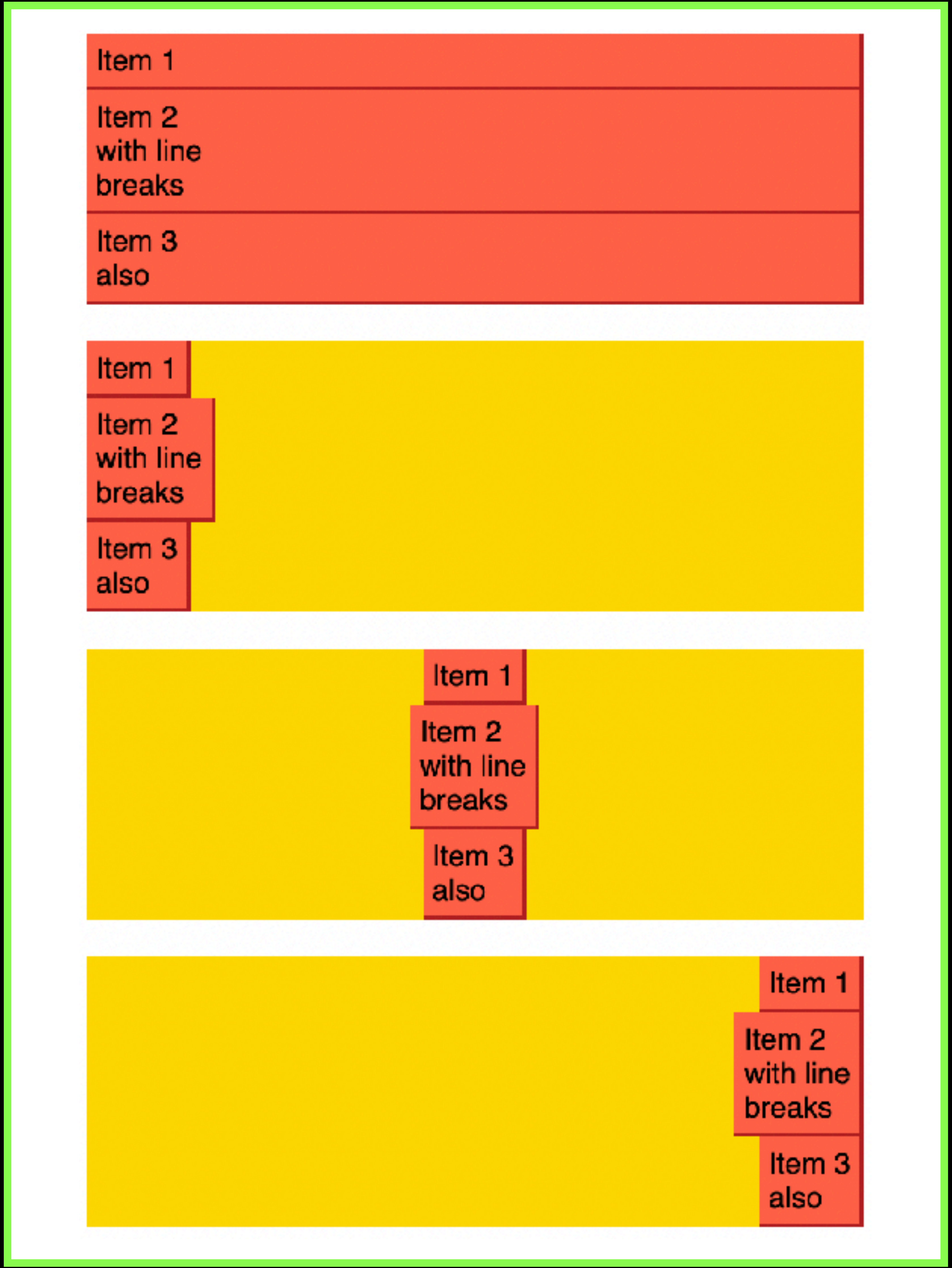
`align-items` similarly allows you to align along the cross axis.

CONTAINER PROPERTIES /*ALIGN*/



```
section {
  display: flex;
  height: 100px;
}

section:nth-child(1) {
  align-items: stretch;
}

section:nth-child(2) {
  align-items: start; /* Default. */
}
section:nth-child(3) {
  align-items: center;
}
section:nth-child(4) {
  align-items: end;
}

section:nth-child(5) {
  align-items: baseline;
}

section:nth-child(5) div:nth-child(1) {
  font-size:200%;
}
```

CONTAINER PROPERTIES /*ALIGN*/

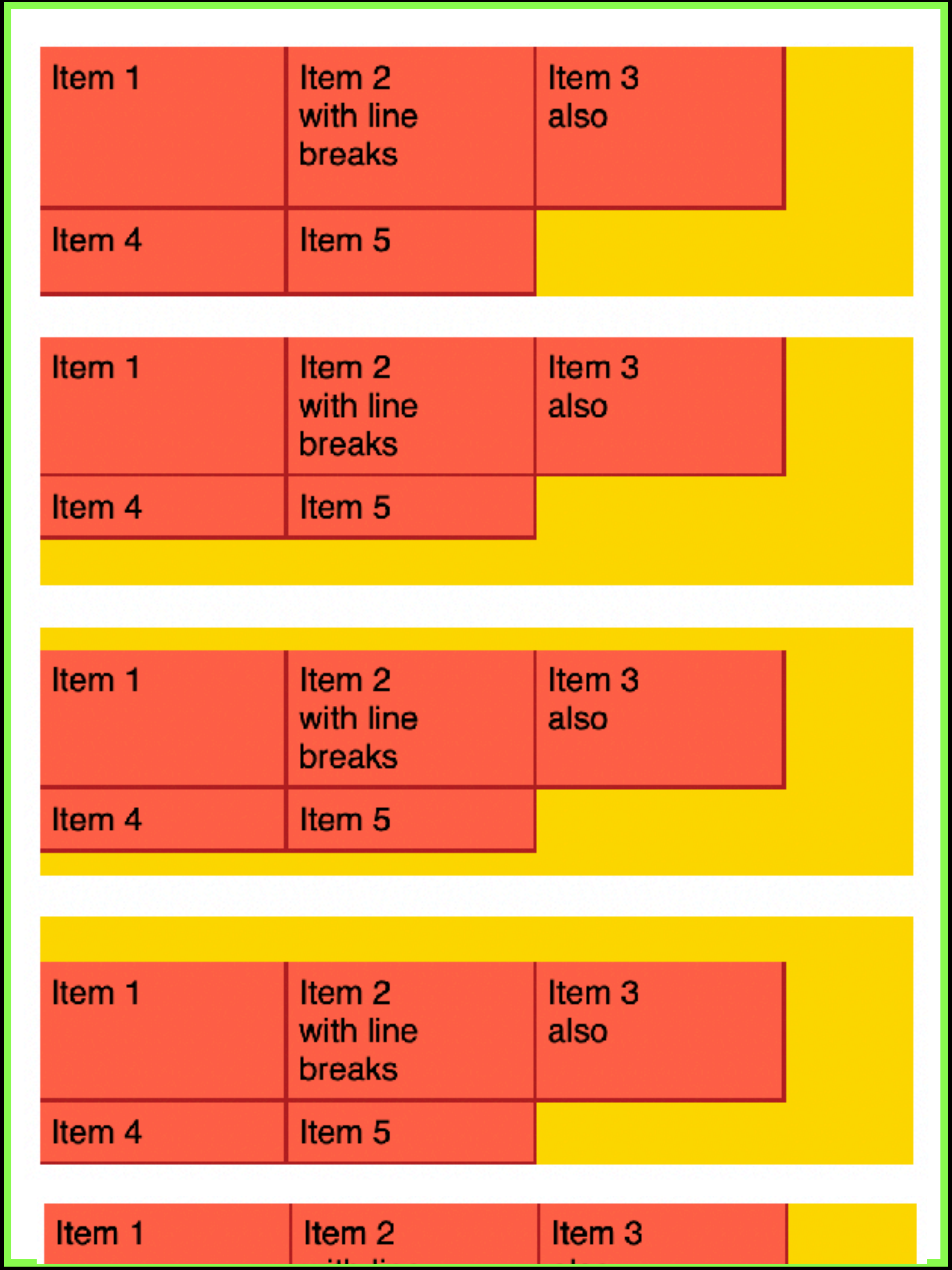


```
section {
  display: flex;
  flex-direction: column;
}

section:nth-child(1) {
  align-items: stretch;
}

section:nth-child(2) {
  align-items: start; /* Default. */
}
section:nth-child(3) {
  align-items: center;
}
section:nth-child(4) {
  align-items: end;
}
```

CONTAINER PROPERTIES
/*ALIGN CONTENT*/

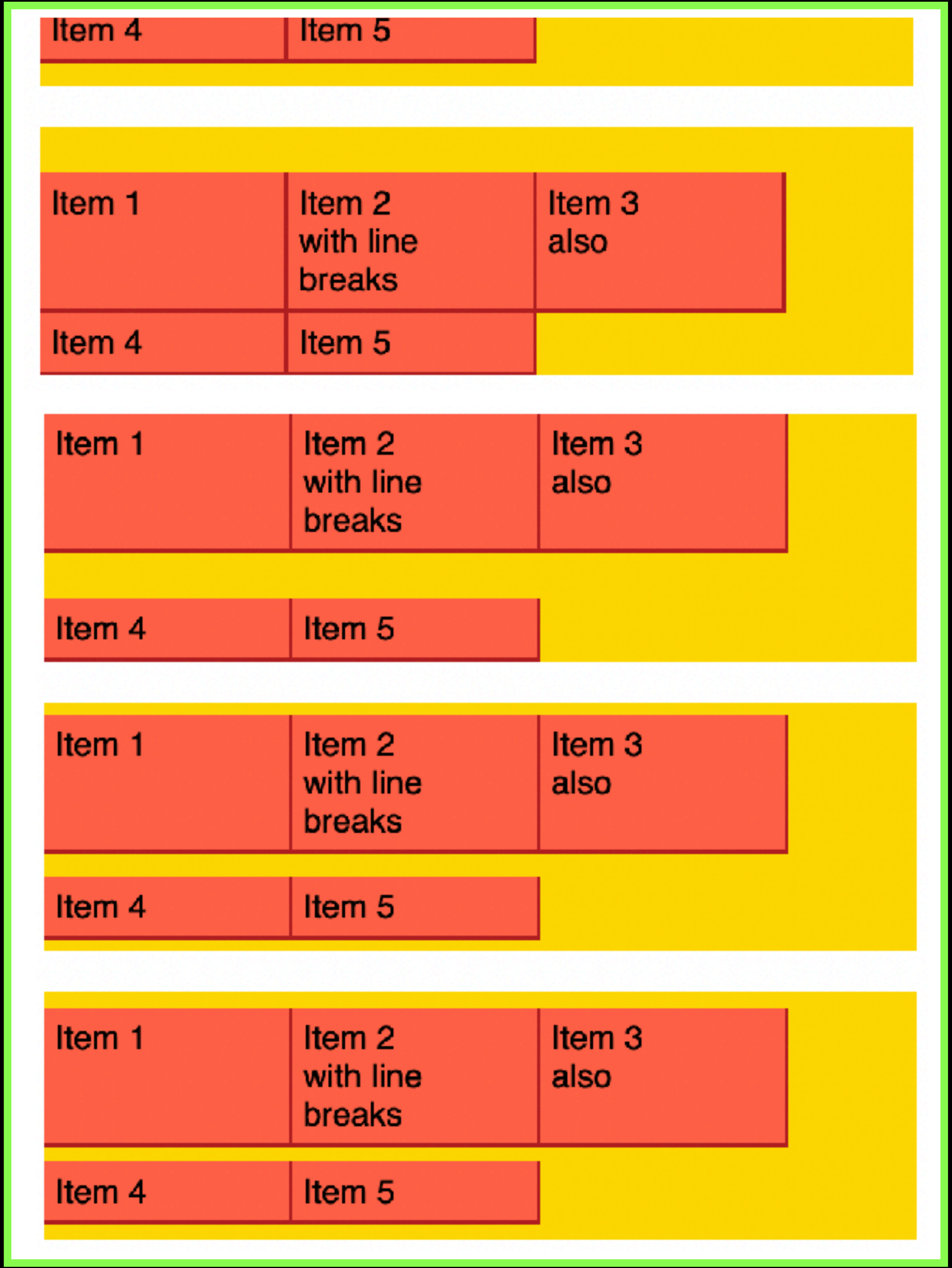
`align-content`, on the other hand, allows you to position the lines (rows/columns) within the container

CONTAINER PROPERTIES /*ALIGN*/



```
section {
   display: flex;
   flex-wrap: wrap;
   height: 120px;
}

div { width: 120px; }

section:nth-child(1) {
   align-content: stretch; /* Default. */
}

section:nth-child(2) {
   align-content: start;
}
section:nth-child(3) {
   align-content: center;
}
section:nth-child(4) {
   align-content: end;
}

section:nth-child(5) {
   align-content: space-between;
}

section:nth-child(6) {
   align-content: space-around;
}

section:nth-child(7) {
   align-content: space-evenly;
}
```

CONTAINER PROPERTIES /*ALIGN*/



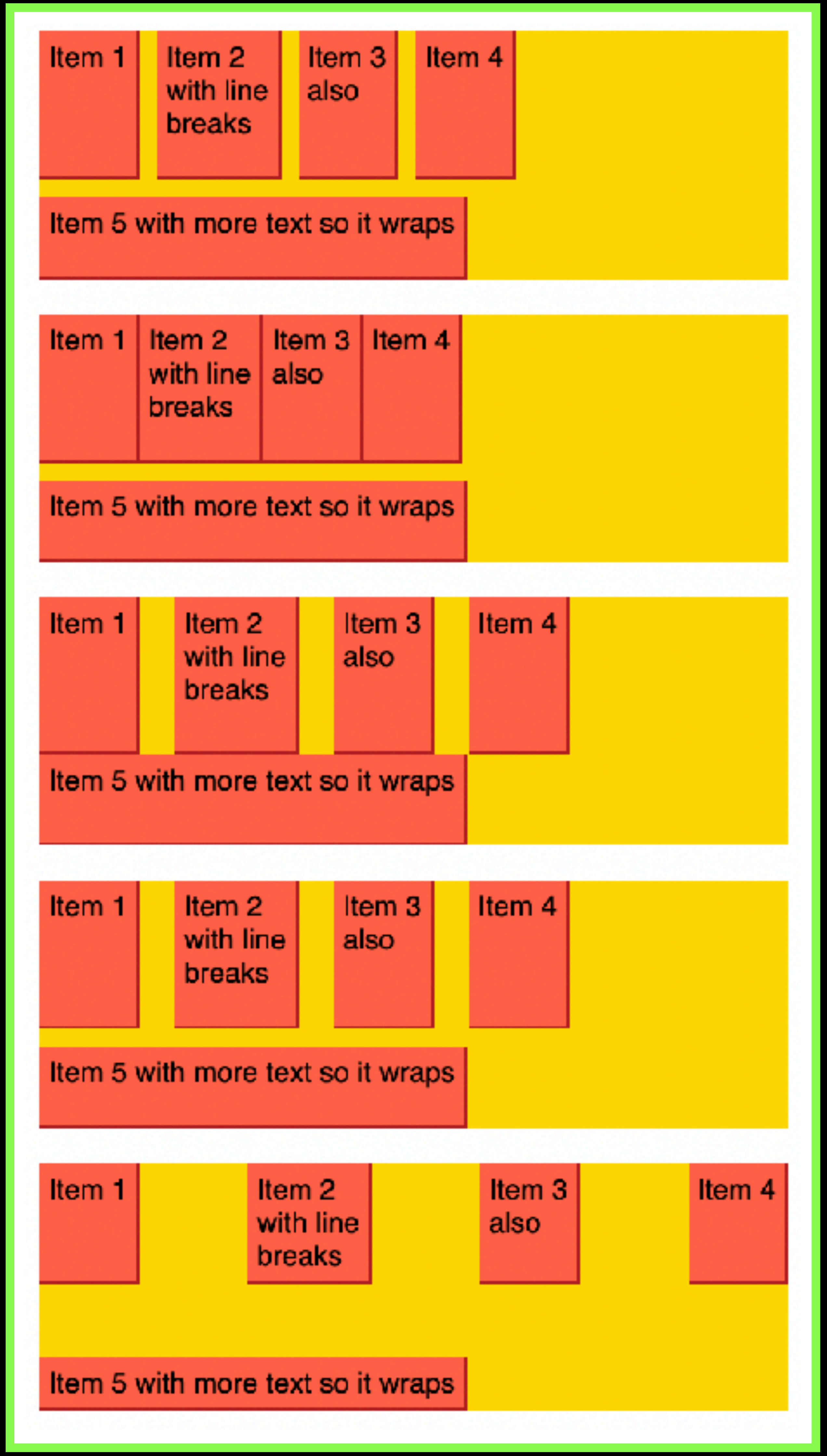
```css
section {
   display: flex;
   flex-wrap: wrap;
   height: 120px;
}

div { width: 120px; }

section:nth-child(1) {
   align-content: stretch; /* Default. */
}

section:nth-child(2) {
   align-content: start;
}
section:nth-child(3) {
   align-content: center;
}
section:nth-child(4) {
   align-content: end;
}

section:nth-child(5) {
   align-content: space-between;
}

section:nth-child(6) {
   align-content: space-around;
}

section:nth-child(7) {
   align-content: space-evenly;
}
```

CONTAINER PROPERTIES

/*GAP PROPERTIES*/

Gap (as opposed to margin) properties are great in flex situations because they won’t affect the outer margins of your items—although you could use this too.

CONTAINER PROPERTIES /*GAP*/

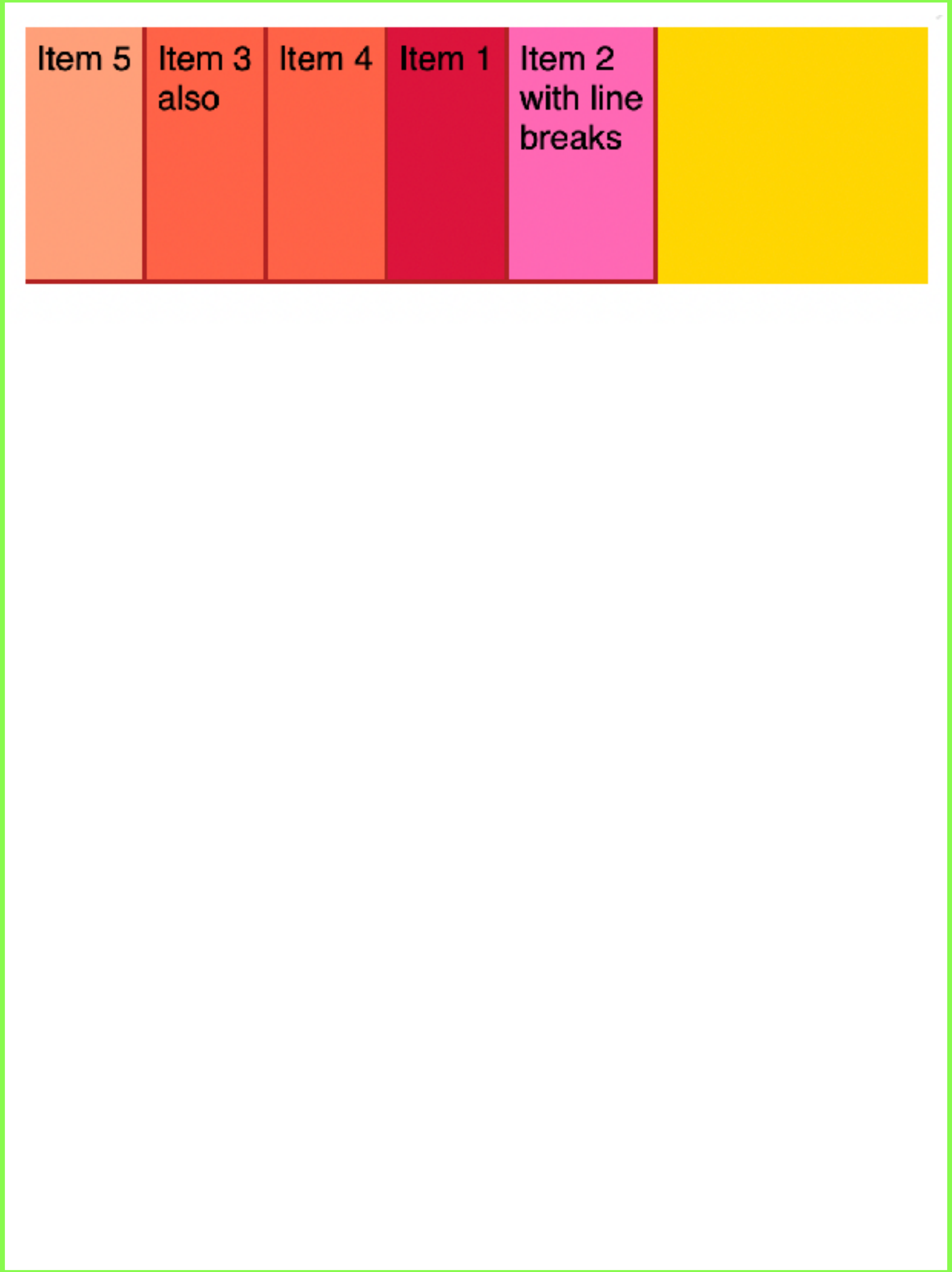


```
section {
  display: flex;
  flex-wrap: wrap;
  height: 140px;
}

section:nth-child(1) { gap: 10px; }

section:nth-child(2) { row-gap: 10px; }

section:nth-child(3) { column-gap: 20px; }

section:nth-child(4) { gap: 10px 20px; /* Shorthand. */ }

section:nth-child(5) {
  justify-content: space-between;
  align-content: space-between;
  gap: 10px; /* No effect here. */
  }
```

ITEM (CHILD) PROPERTIES
/*ORDER*/

Some flex properties can be applied directly to the children. You can manually set the order that divs appear in along the axis (similar to reverse)

ITEM PROPERTIES /*ORDER*/



```
section {
  display: flex;
  height: 120px;
}

div { order: 0; } /* Default. */

div:nth-child(1) {
  background-color: crimson;
  order: 1;
}

div:nth-child(2) {
  background-color: hotpink;
  order: 2;
}

div:nth-child(5) {
  background-color: lightsalmon;
  order: -1;
}
```

ITEM (CHILD) PROPERTIES

/*ORDER*/

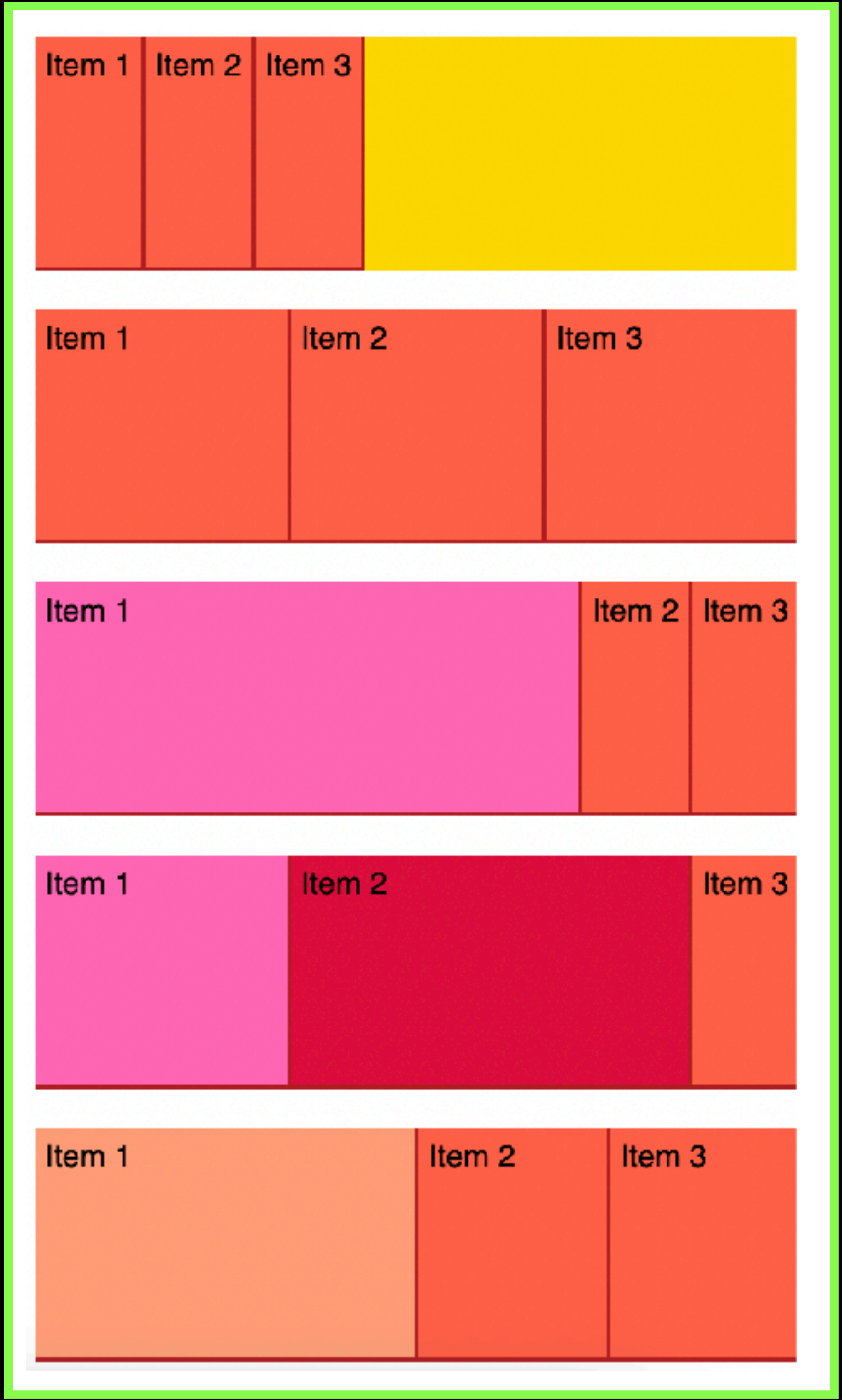
The change is only visual—so `first-child`, for instance, will still apply to the first element in the DOM / html **not** the order.

ITEM (CHILD) PROPERTIES

/*GROW AND SHRINK*/

`flex-grow` and `flex-shrink` tells an element how much of the undefined leftover space it should take up. It does not use units, rather it is relative to other elements.

ITEM PROPERTIES /*GROW*/

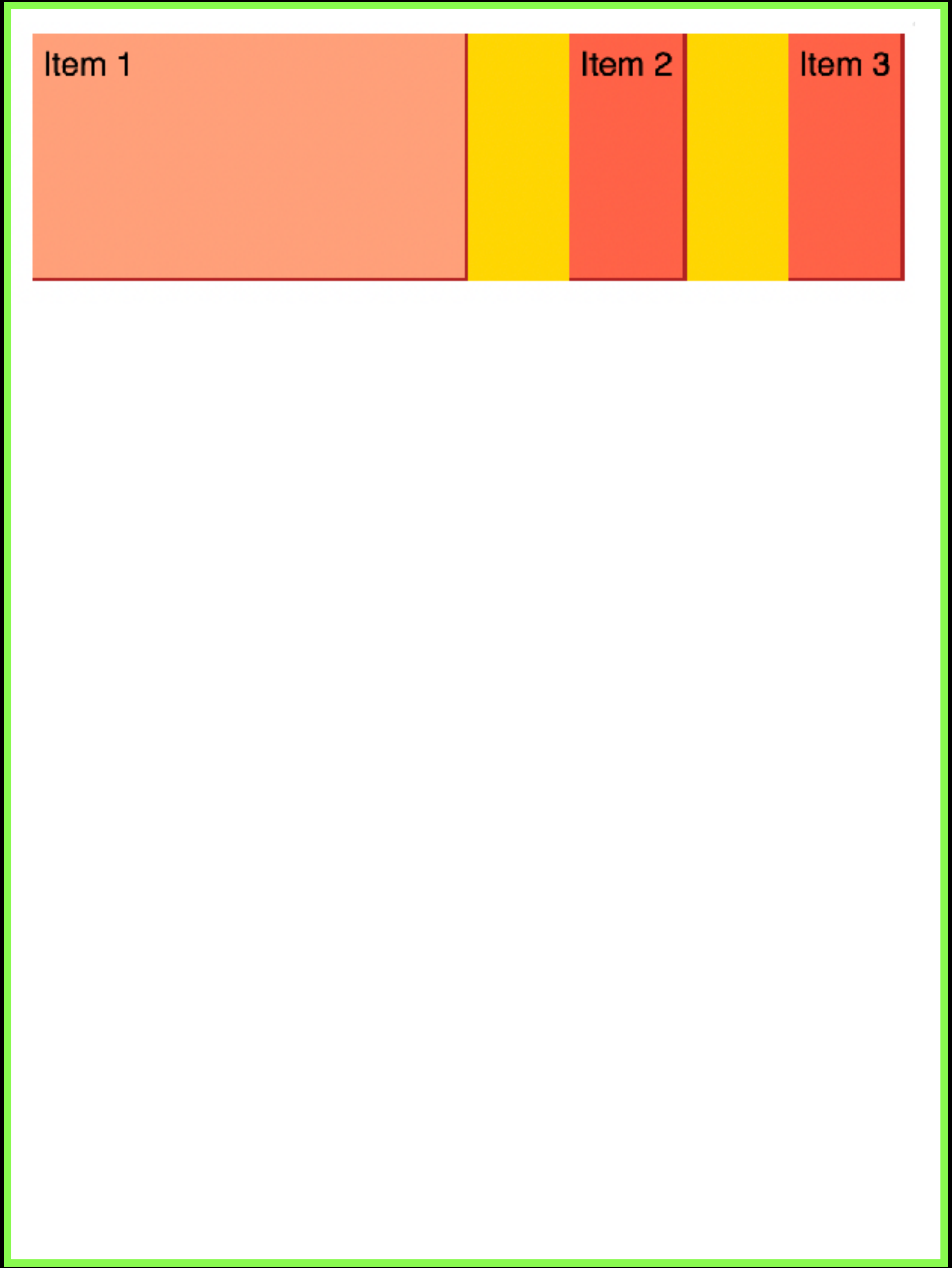


```css
section {
   display: flex;
   height: 120px;
}

section:nth-child(2) div {
   flex-grow: 1; /* All grow equally. */
}

section:nth-child(3) div:nth-child(1) {
   background-color: hotpink;
   flex-grow: 1; /* Just grow this one. */
}

section:nth-child(4) div:nth-child(1) {
   background-color: hotpink;
   flex-grow: 1;
}

section:nth-child(4) div:nth-child(2) {
   background-color: crimson;
   flex-grow: 2; /* Grow twice as much. */
}

section:nth-child(5) div {
   width: 50%; /* With 3 `div` would be 150%. */
}

section:nth-child(5) div:nth-child(1) {
   background-color: lightsalmon;
   flex-shrink: 0; /* Don’t shrink this one! */
}
```

ITEM (CHILD) PROPERTIES
/*FLEX-BASIS*/

`flex-basis` defines the width and height of an element before the space is distributed among all elements.

ITEM PROPERTIES /*BASIS*/

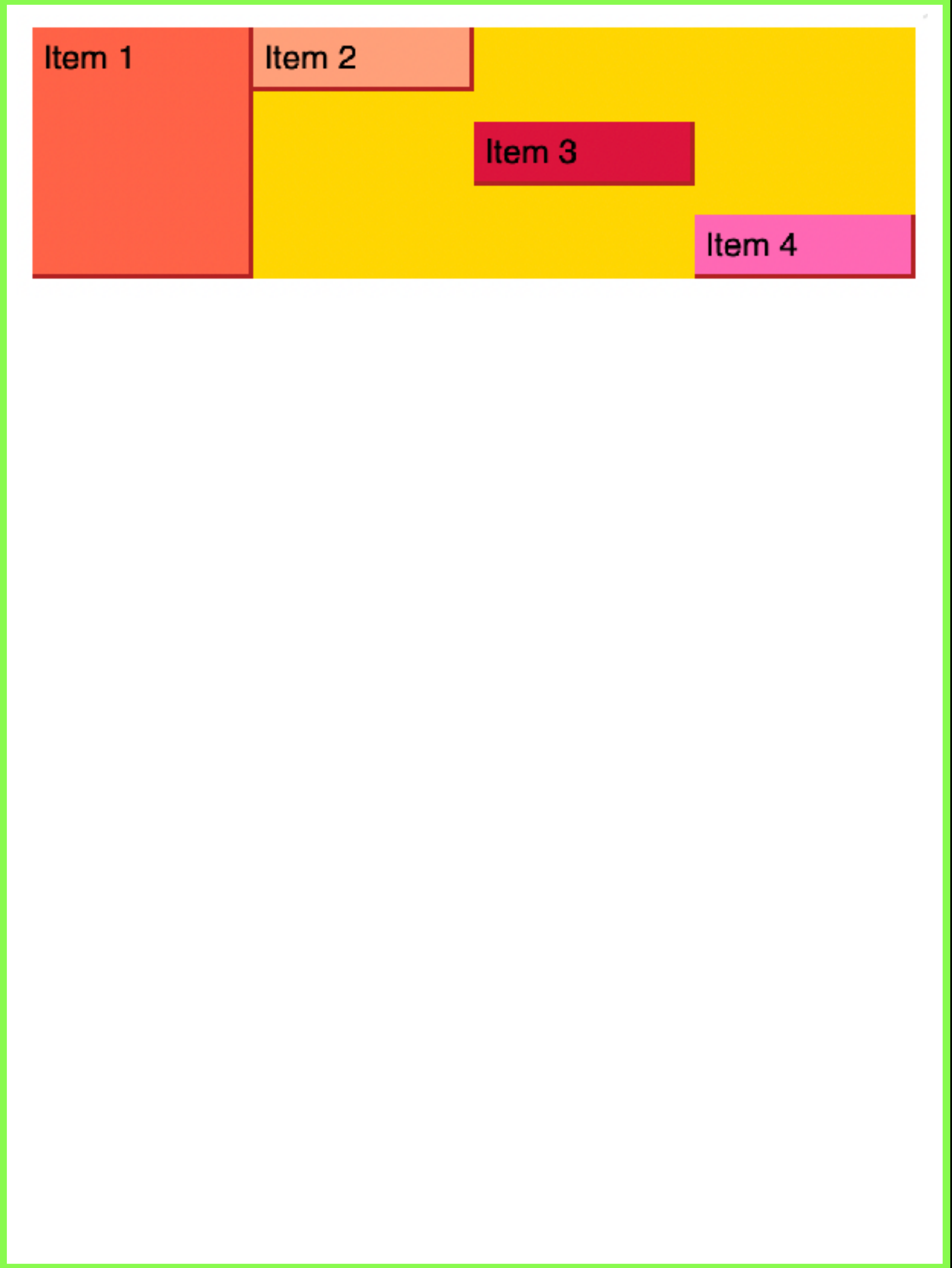


```
section {
  display: flex;
  height: 120px;
  justify-content: space-between;
}

div:nth-child(1) {
  background-color: lightsalmon;
  flex-basis: 50%;
}
```

ITEM (CHILD) PROPERTIES
/*ALIGN-SELF*/

Lastly… `align-self` overrides the alignment properties applied to a parent to realign a single child.

ITEM PROPERTIES /*ALIGN SELF*/



```
section {
  display: flex;
  height: 120px;
}

div { flex-grow: 1; }

div:nth-child(1) {
  align-self: stretch; /* Default. */
}

div:nth-child(2) {
  background-color: lightsalmon;
  align-self: start;
}

div:nth-child(3) {
  background-color: crimson;
  align-self: center;
}

div:nth-child(4) {
  background-color: hotpink;
  align-self: end;
}
```

NEXT WEEK

CSS GRID, MORE CSS, AND IMAGE TREATMENTS

HOMEWORK

- > Harmonic Collection Entry 5
- > Work on your Midterm!
  - > Refine your entries, and organize your files. Upload to GitHub and share with me if you have not already.
  - > Work on your reflection. Remember, it's 1-2 pages, double spaced, and the requirements are in the Week 5 slides.
  - > Sign up for a critique slot if you have not already.

SESSION SIX
OCTOBER 3, 2023

THANK YOU

DAVIS SCHERER
DSCHERER@NEWSCHOOL.EDU