16A

CSS LAYOUT WITH FLEXBOX

OVERVIEW

- Flexbox terminology
- Flexbox containers
- Flow: Flow direction and text wrap
- Alignment on main and cross axes
- Specifying how items in a flexbox "flex"
- Changing the order of flex items

About Flexbox

- **Flexbox** is a display mode that lays out elements along one axis (horizontal or vertical).
- Useful for menu options, galleries, product listings, etc.
- Items in a flexbox can expand, shrink, and/or wrap onto multiple lines, making it a great tool for responsive layouts.
- Items can be reordered, so they aren't tied to the source order.
- Flexbox can be used for individual components on a page or the whole page layout.

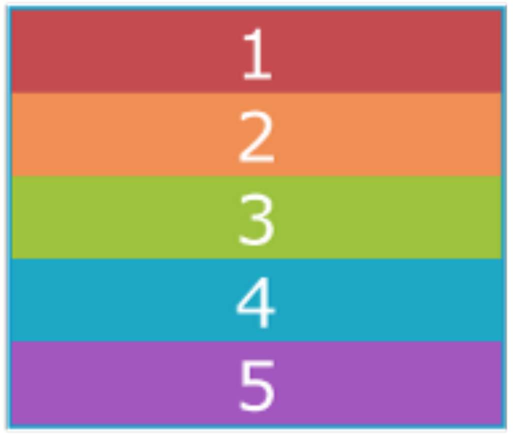
Flexbox Container

`display: flex`

- To turn on Flexbox mode, set the element's **`display`** to **`flex`**.
- This makes the element a **flexbox container**.
- All of its direct children become **flex items** in that container.
- By default, items line up in the writing direction of the document (left to right rows in left-to-right reading languages).

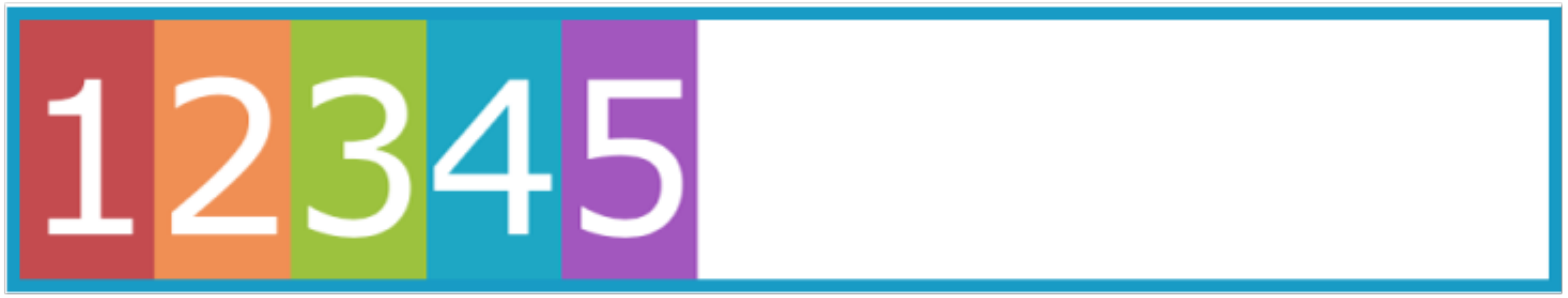
Flexbox Container (cont'd)

By default, the **divs** display as block elements, stacking up vertically. Turning on flexbox mode makes them line up in a row.

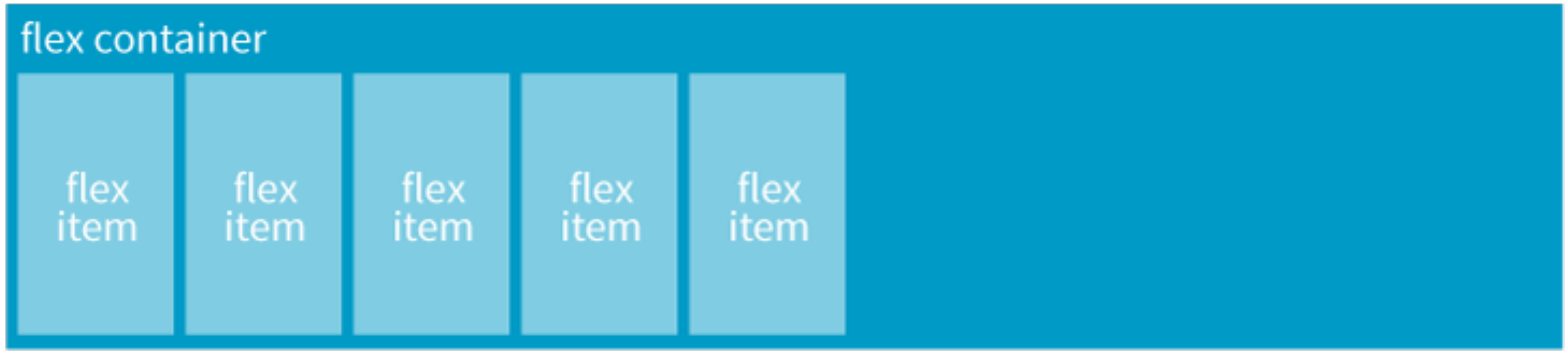


block layout mode

`display: flex;`



flexbox layout mode

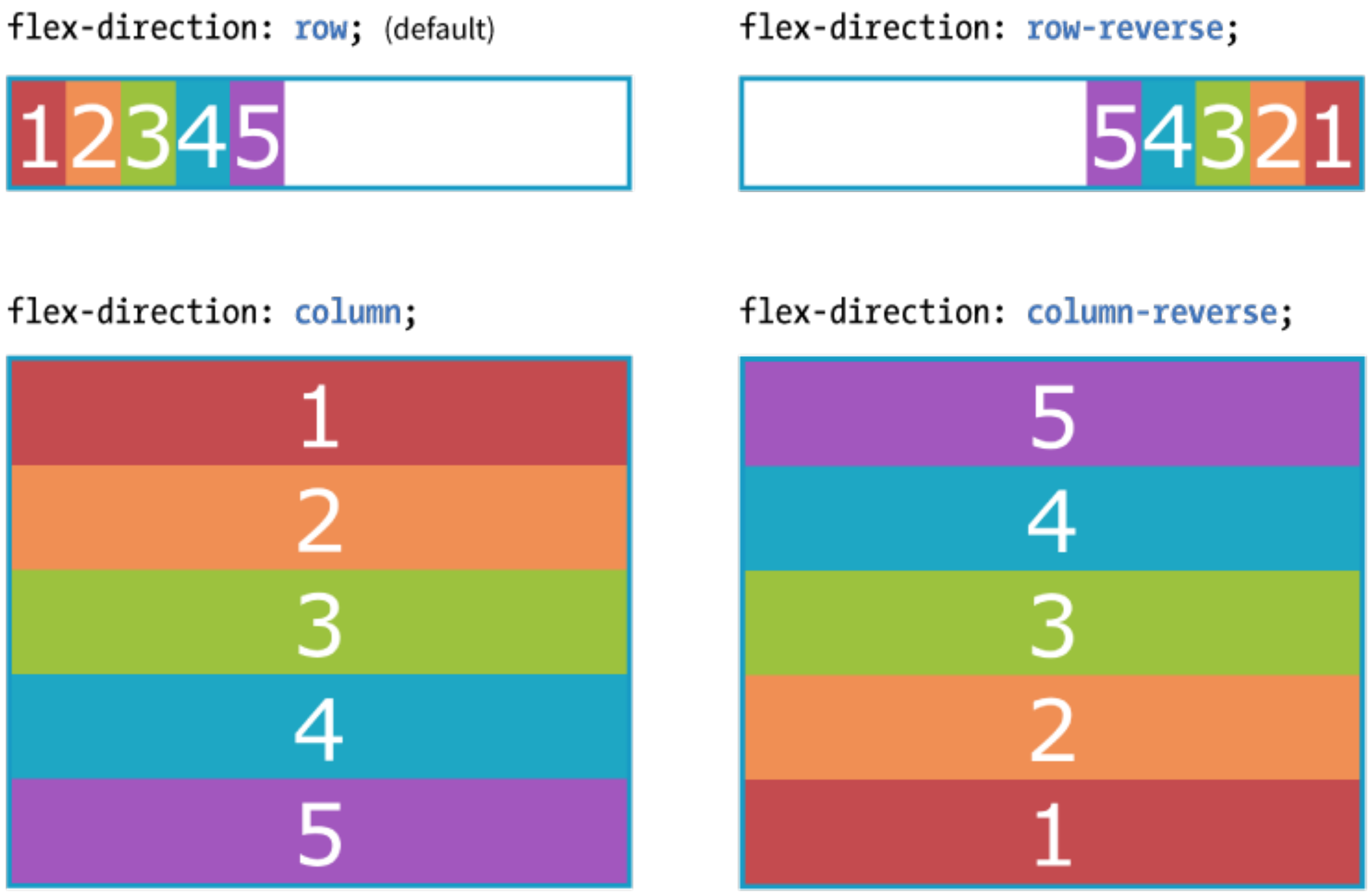


Rows and Columns (Direction)

`flex-direction`

Values: `row`, `column`, `row-reverse`, `column-reverse`

The default value is **`row`** (for L-to-R languages), but you can change the direction so items flow in columns or in reverse order:


flex-direction: row; (default)
1 2 3 4 5
flex-direction: row-reverse;
5 4 3 2 1
flex-direction: column;
1 2 3 4 5
flex-direction: column-reverse;
5 4 3 2 1

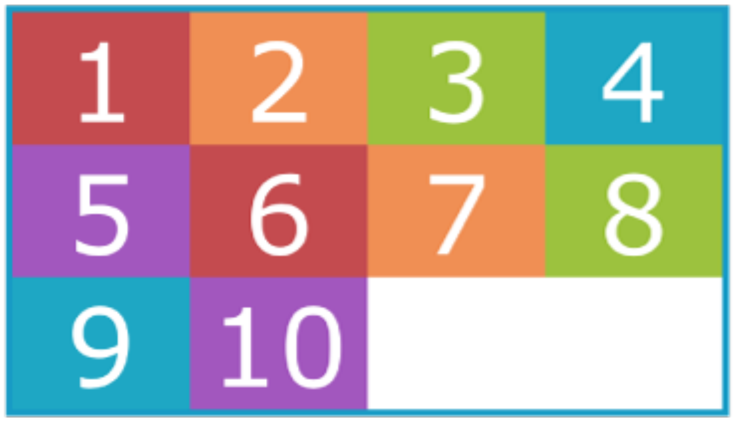
Wrapping Flex Lines

`flex-wrap`

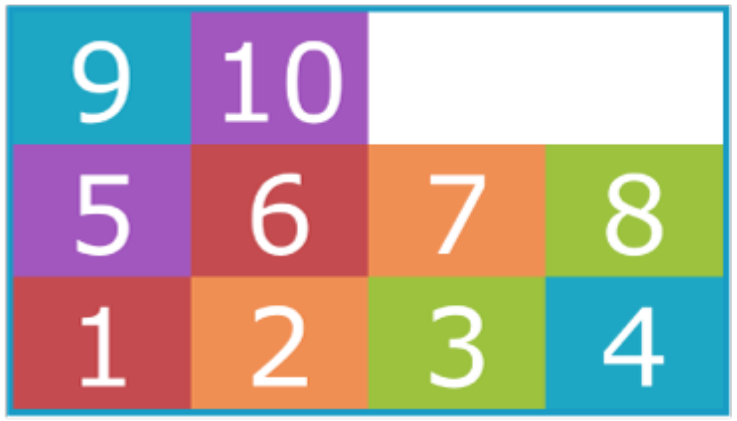
Values: `wrap`, `nowrap`, `wrap-reverse`

Flex items line up on one axis, but you can allow that axis to wrap onto multiple lines with the **`flex-wrap`** property:

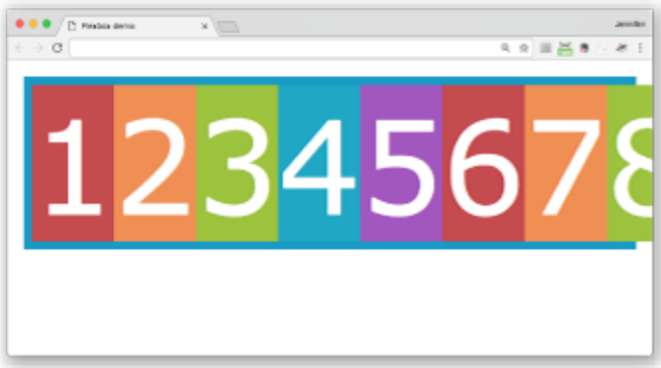
`flex-wrap: wrap;`



`flex-wrap: wrap-reverse;`



`flex-wrap: nowrap;` (default)



When wrapping is disabled, flex items squish if there is not enough room, and if they can't squish any further, may get cut off if there is not enough room in the viewport.

Flex Flow (Direction + Wrap)

`flex-flow`

Values: *Flex-direction flex-flow*

The shorthand **`flex-flow`** property specifies both direction and wrap in one declaration.

Example

```
#container {
  display: flex;
  height: 350px;
  flex-flow: column wrap;
}
```

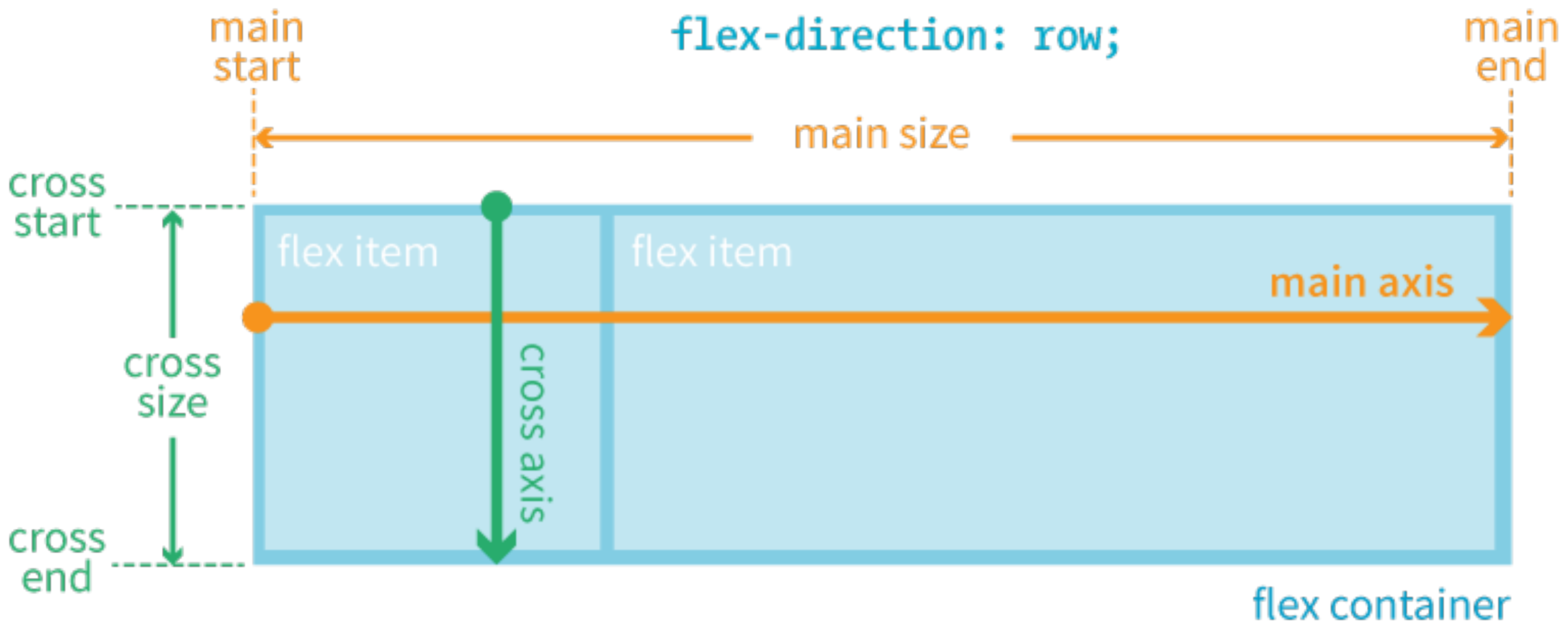
Flexbox Alignment Terminology

- Flexbox is “direction-agnostic,” so we talk in terms of *main axis* and *cross axis* instead of rows and columns.
- The **main axis** runs in whatever direction the flow has been set.
- The **cross axis** runs perpendicular to the main axis.
- Both axes have a **start**, **end**, and **size**.

ROW: Main and Cross Axes

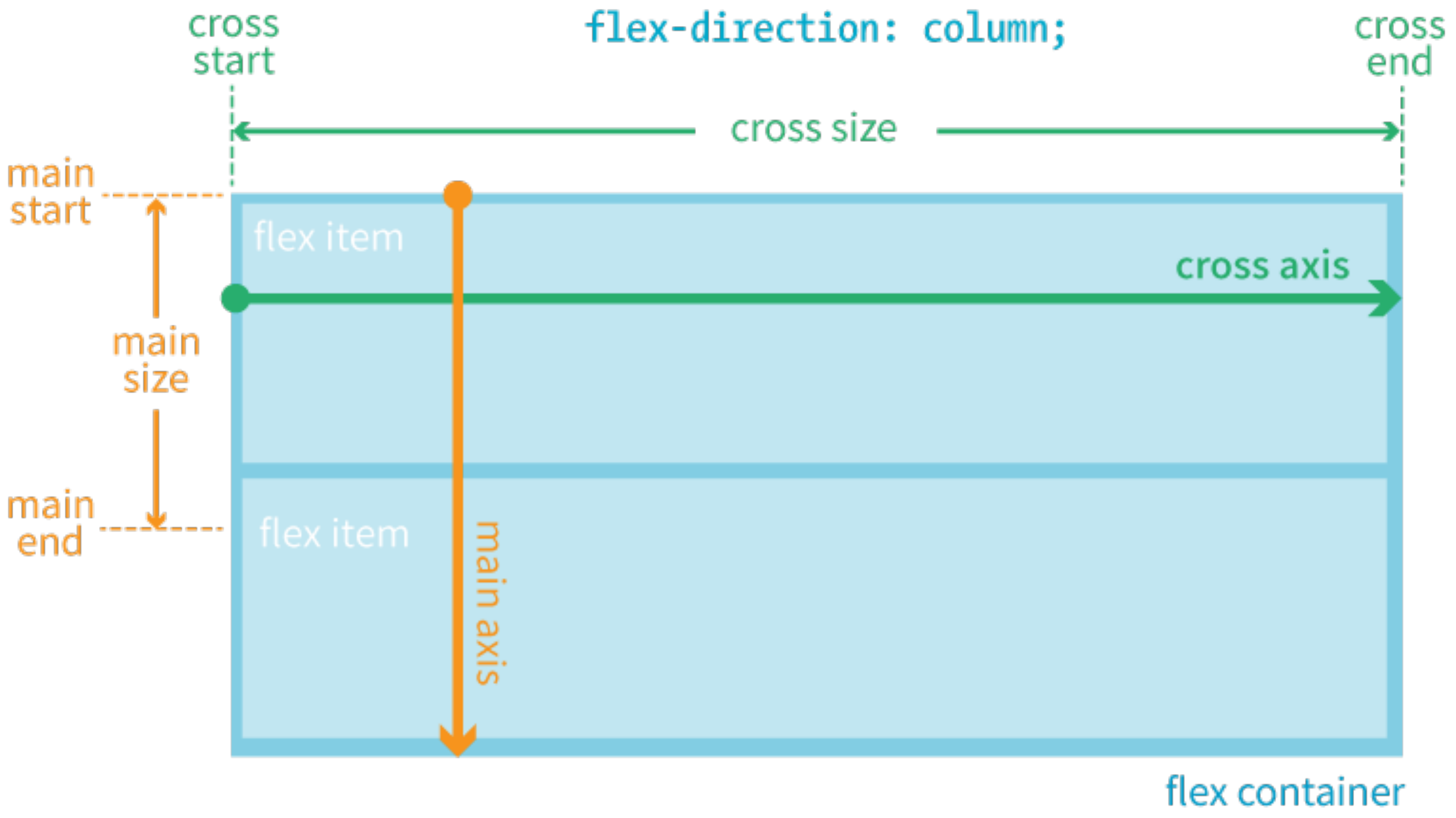
FOR LANGUAGES THAT READ HORIZONTALLY FROM LEFT TO RIGHT:

When `flex-direction` is set to `row`, the main axis is horizontal and the cross axis is vertical.



COLUMN: Main and Cross Axes

When **`flex-direction`** is set to **`column`**, the main axis is vertical and the cross axis is horizontal.



Aligning on the Main Axis

`justify-content`

Values: `flex-start`, `flex-end`, `center`, `space-between`, `space-around`

When there is space left over on the **main axis**, you can specify how the items align with the **`justify-content`** property (notice we say *start* and *end* instead of left/right or top/bottom).

The **`justify-content`** property applies to the **flex container**.

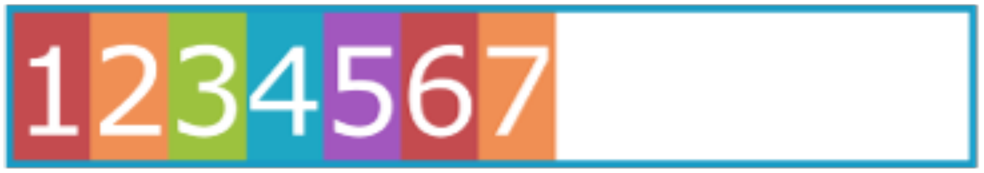
Example:

```
#container {
  display: flex;
  justify-content: flex-start;
}
```

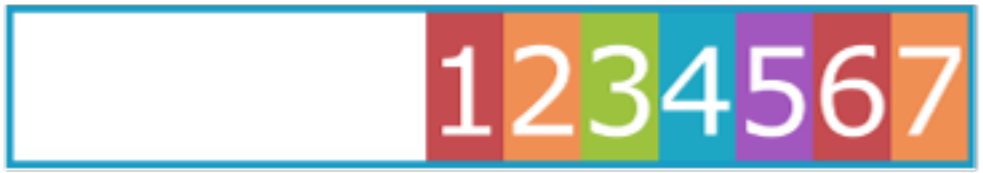
Aligning on the Main Axis (cont'd)

When the direction is **row**, and the **main axis is horizontal**

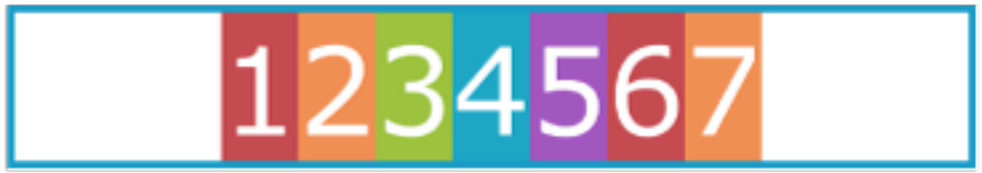
`justify-content: flex-start;` (default)



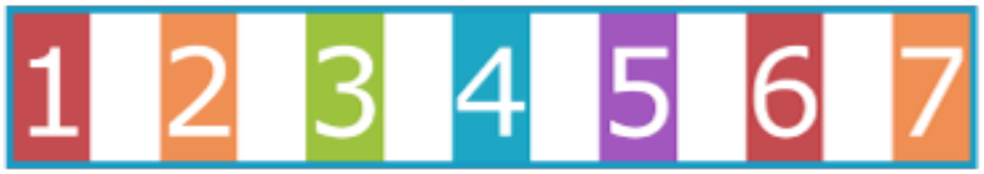
`justify-content: flex-end;`



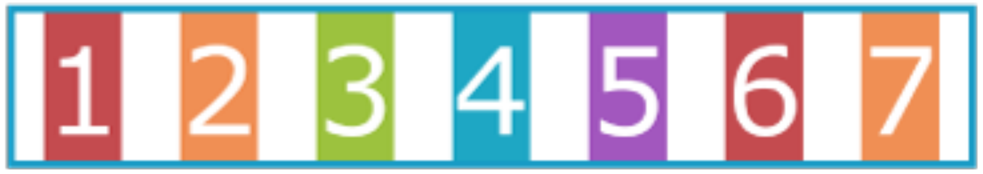
`justify-content: center;`



`justify-content: space-between;`



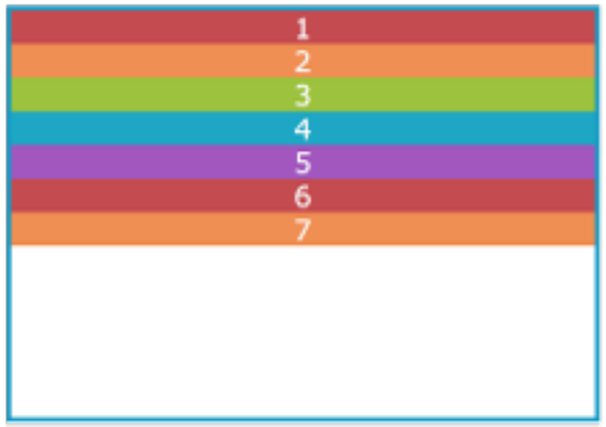
`justify-content: space-around;`



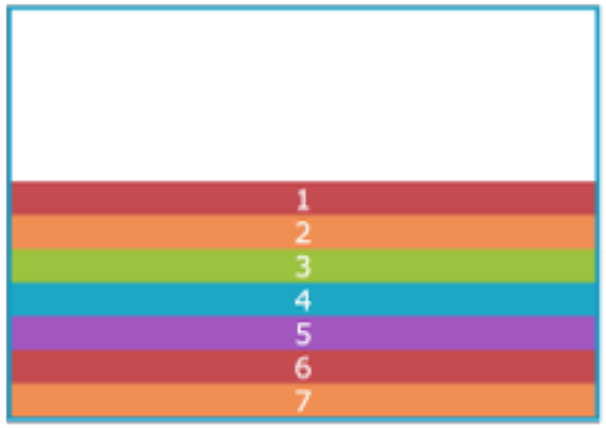
Aligning on the Main Axis (cont'd.)

When the direction is **column**, and the main axis is **vertical**

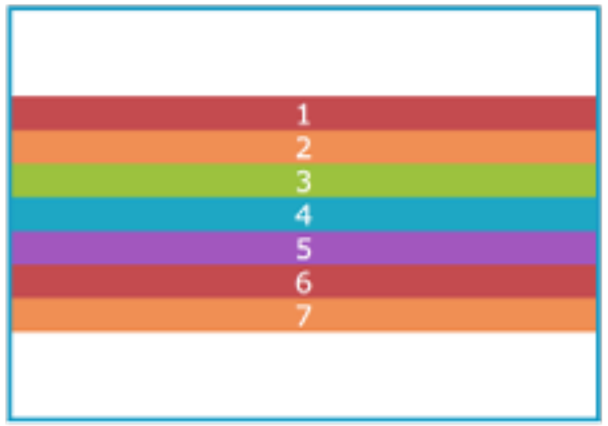
`justify-content: flex-start;` (default)



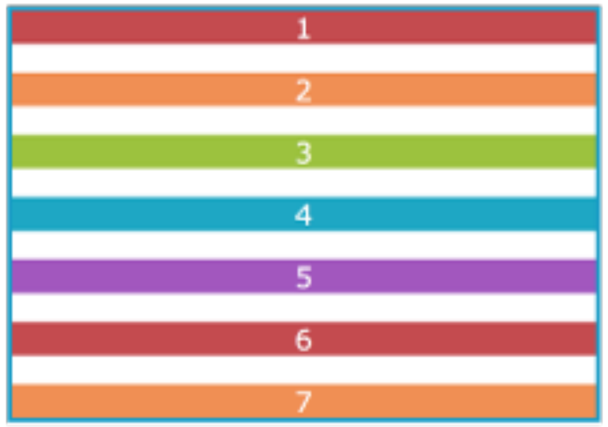
`justify-content: flex-end;`



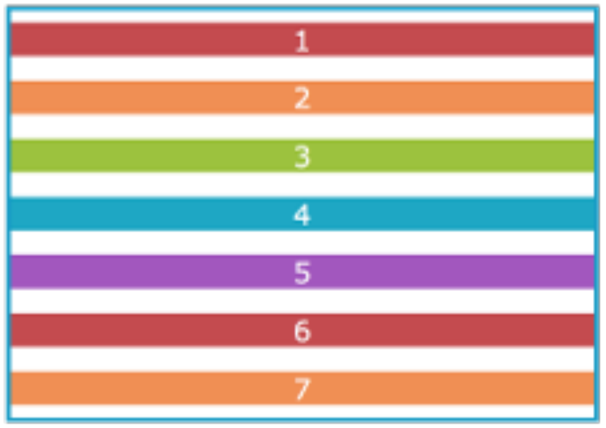
`justify-content: center;`



`justify-content: space-between;`



`justify-content: space-around;`



NOTE: I needed to specify a height on the container to create extra space on the main axis. By default, it's just high enough to contain the content.

A WORD FROM THE AUTHOR

“Keeping the main and cross axes straight in your mind when changing between rows and columns is one of the trickiest parts of using Flexbox.

Once you master that, you’ve got it!”

—Jennifer Robbins

Aligning on the Cross Axis

`align-items`

Values: `flex-start`, `flex-end`, `center`, `baseline`, `stretch`

When there is space left over on the **cross axis**, you can specify how the items align with the **`align-items`** property.

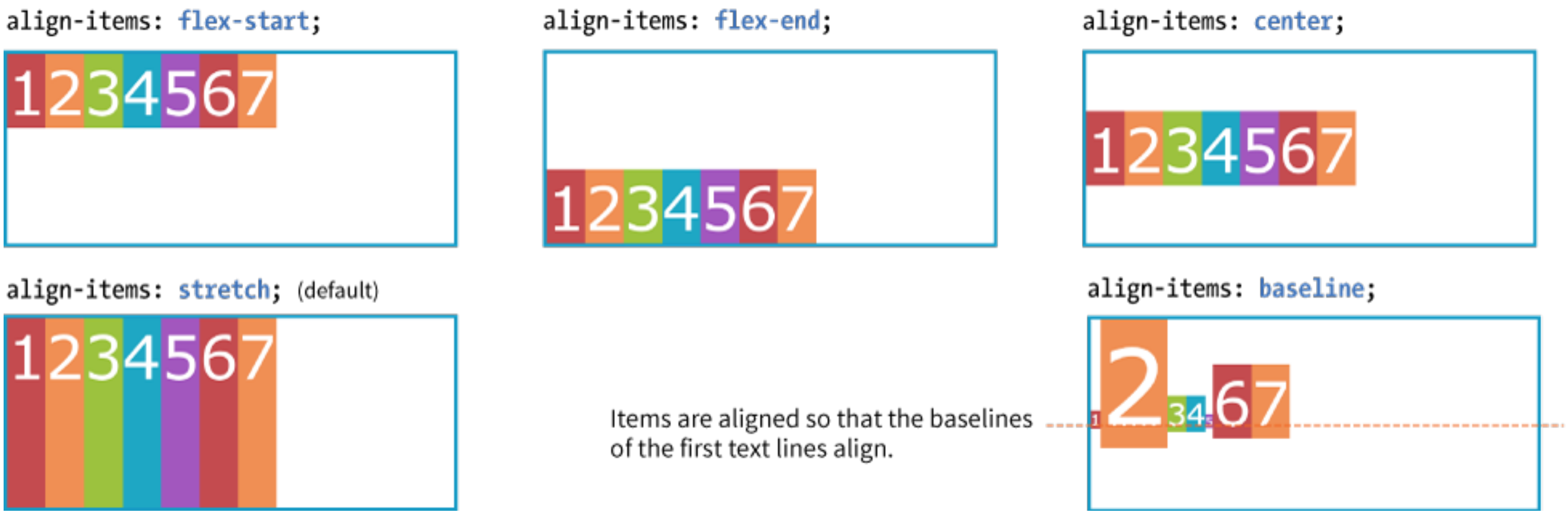
The **`align-items`** property applies to the **flex container**.

Example:

```
#container {
  display: flex;
  flex-direction: row;
  height: 200px;
  align-items: flex-start;
}
```

Aligning on the Cross Axis (cont'd)

When the direction is **row**, the main axis is horizontal, and the **cross axis is vertical**.


align-items: flex-start;
align-items: flex-end;
align-items: center;
align-items: stretch; (default)
align-items: baseline;
1234567
Items are aligned so that the baselines of the first text lines align.

NOTE: I needed to specify a height on the container to create extra space on the cross axis. By default, it's just high enough to contain the content.

Aligning on the CROSS Axis (cont'd)

`align-self`

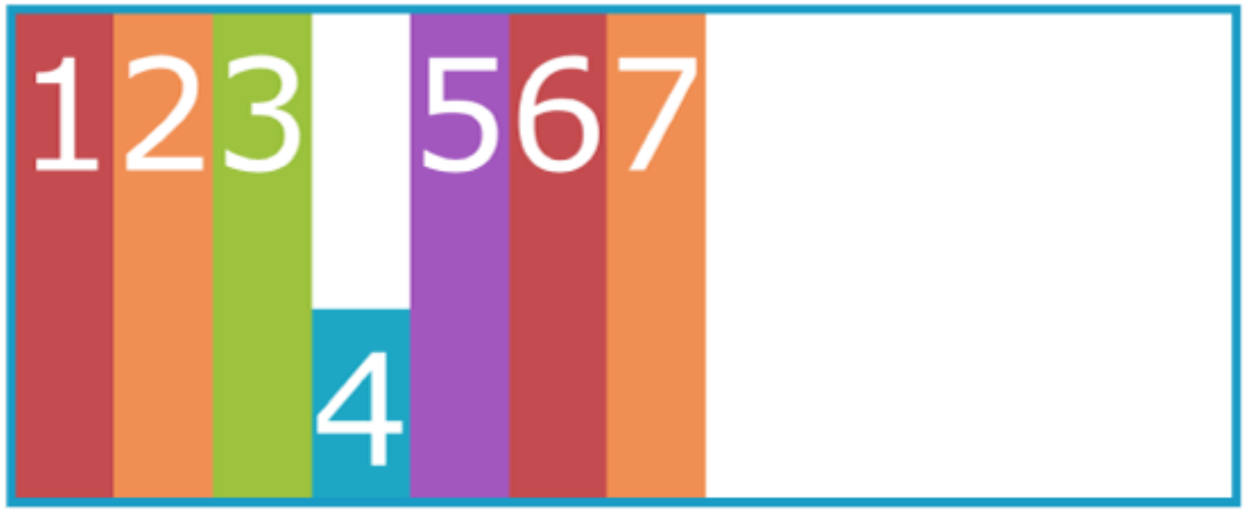
Values: `flex-start`, `flex-end`, `center`, `baseline`, `stretch`

Aligns an **individual item** on the cross axis. This is useful if one or more items should override the **`align-items`** setting for the container.

The **`align-self`** property applies to the **flex item**.

Example:

```
.box4 {
  align-self: flex-end;
}
```



Aligning on the CROSS Axis (cont’d)

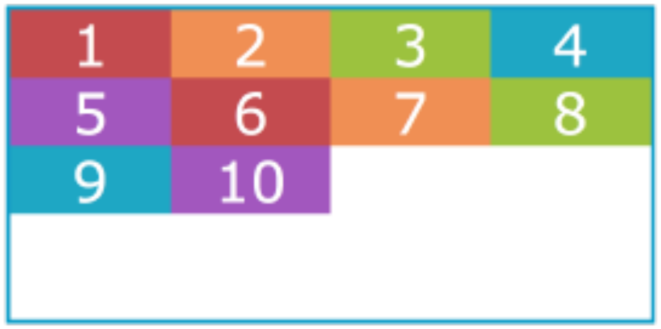
`align-content`

Values: `flex-start`, `flex-end`, `center`, `space-around`, `space-between`, `stretch`

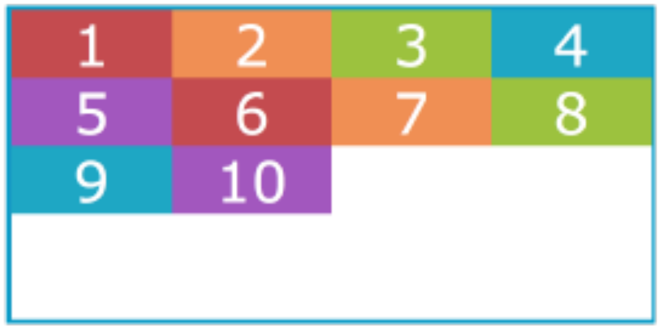
When lines are set to **wrap** and there is extra space on the cross axis, use `align-content` to **align the lines of content**.

The **`align-content`** property applies to the **flex container**.

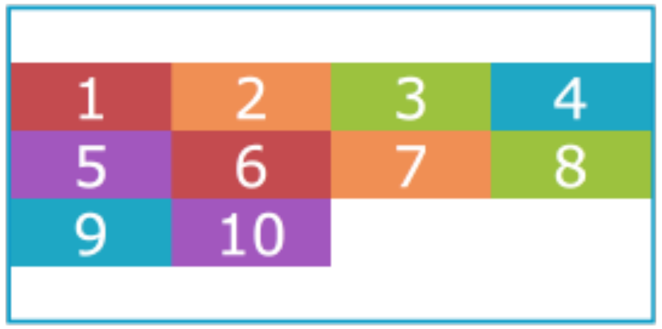
`align-content: flex-start;`



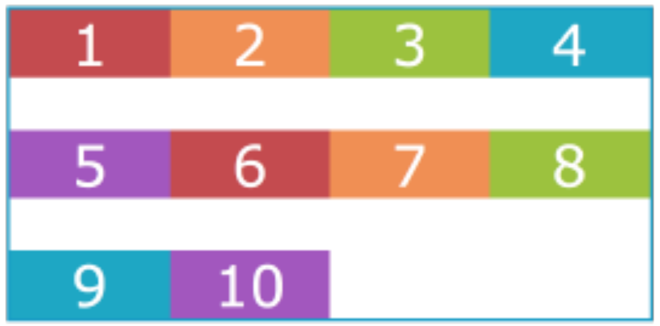
`align-content: flex-end;`



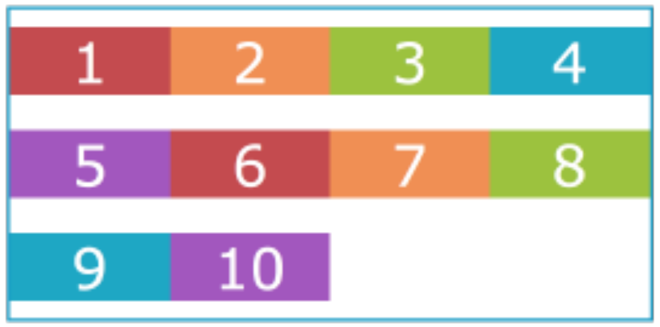
`align-content: center;`



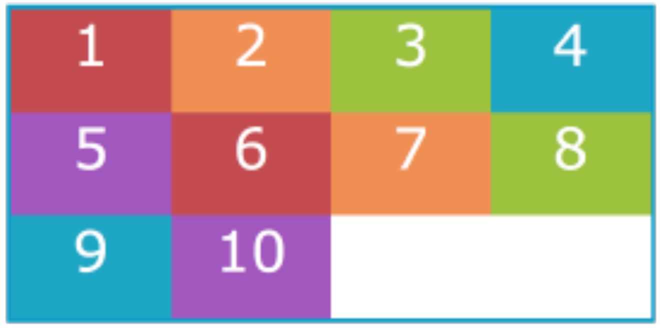
`align-content: space-between;`



`align-content: space-around;`



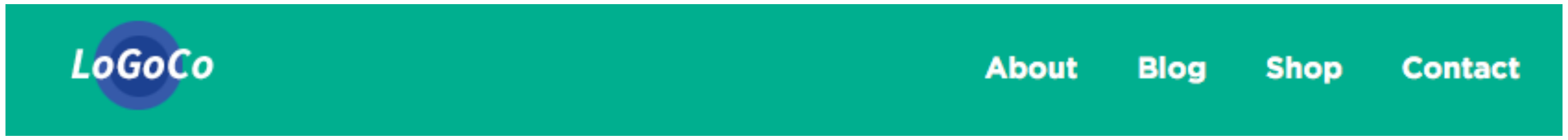
`align-content: stretch;` (default)



Aligning with Margins

Use a margin (set to auto) to put extra space on the side of particular flex items.

Example: Adding an auto margin to the right of the first flex item (the **`li`** with the logo) pushes the remaining **`li`** to the right:



```
ul {
  display: flex;
  align-items: center;
  ...
}
li.logo {
  margin-right: auto;
}
```

Specifying How Items “Flex”

`flex`

Values: `none`, '*flex-grow flex-shrink flex-basis*'

- Items can resize (flex) to fill the available space on the main axis in the container.
- The **`flex`** property identifies how much an item can grow and shrink and identifies a starting size
- It distributes extra space in the container *within* items (compared to `justify-content` that distributes space *between and around* items).

flex Property Example

`flex` is a shorthand for separate **`flex-grow`**, **`flex-shrink`**, and **`flex-basis`** properties.

The values 1 and 0 work like on/off switches.

```
li {
  flex: 1 0 200px;
}
```

In this example, list items in the flex container start at 200 pixels wide, are permitted to expand wider (`flex-grow: 1`), and are not permitted to shrink (`flex-shrink: 0`).

NOTE: The spec recommends always using the **`flex`** property and using individual properties only for overrides.

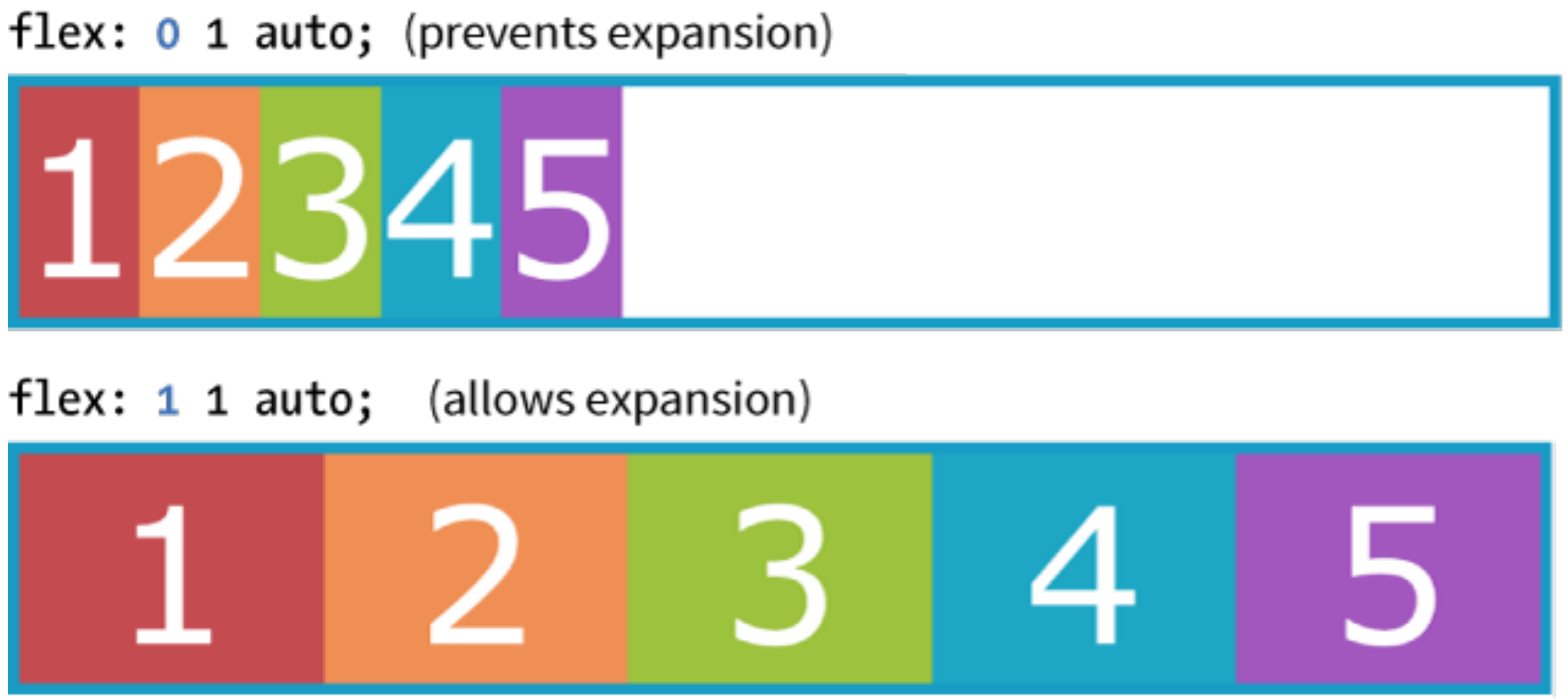
Expanding Items (flex-grow)

`flex-grow`

Values: *Number*

Specifies whether and in what proportion an item may stretch larger. 1 allows expansion; 0 prevents it.

`flex-grow` is applied to the **flex item element**.


flex: 0 1 auto; (prevents expansion)
12345
flex: 1 1 auto; (allows expansion)
1 2 3 4 5

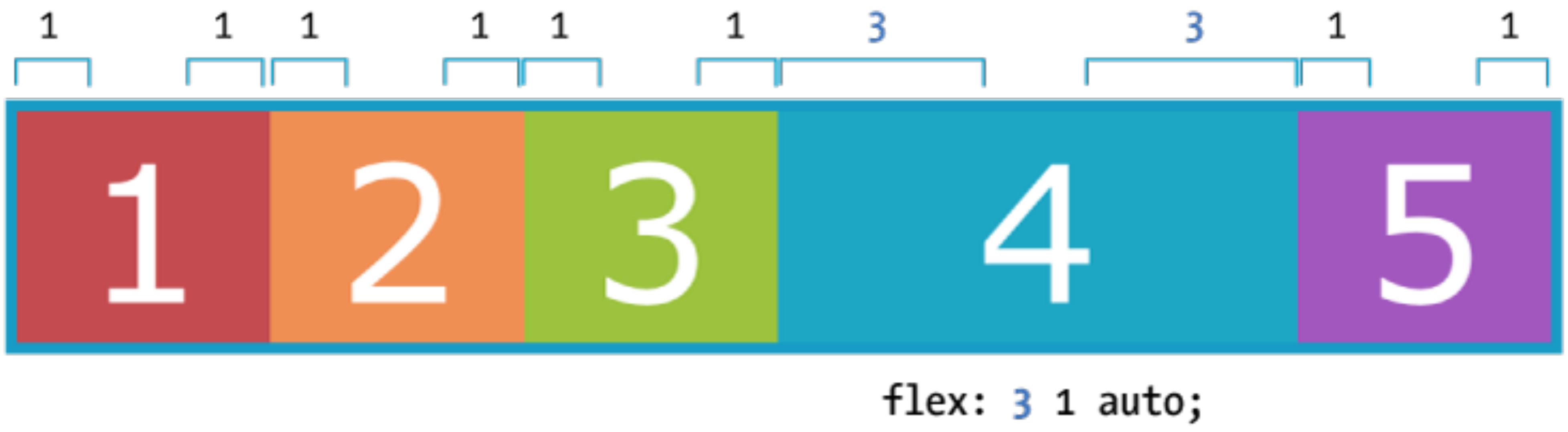
Expanding Items (cont'd)

Relative Flex

*When the **`flex-basis`** has a value **other than 0***, higher integer values act as a ratio that applies more space within that item.

Example: A value of **`3`** assigns **three times more space** to box4 than items with a `flex-grow` value of **`1`**. (Note that it isn't necessarily 3x as wide as the other items.)

```
.box4 { flex: 3 1 auto; }
```



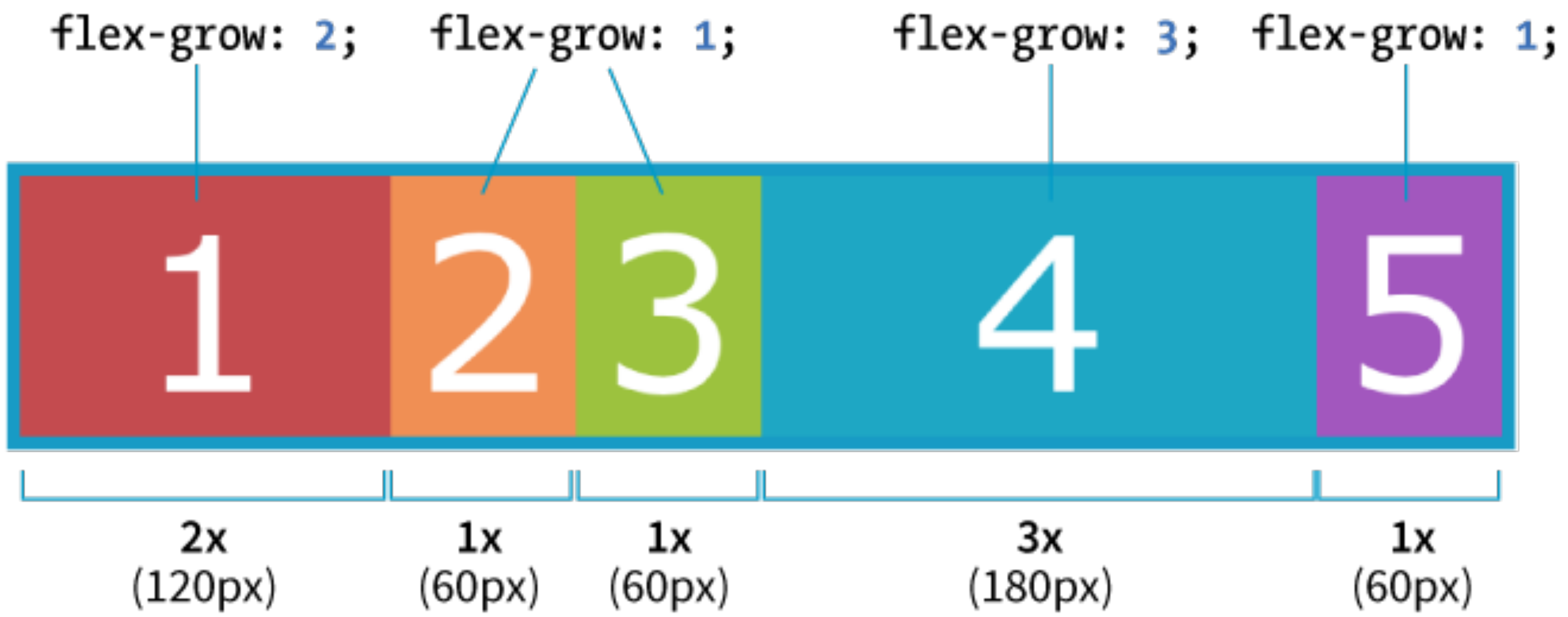
Expanding Items (cont'd.)

Absolute Flex

*When the **`flex-basis`** is **0***, items get sized proportionally according to the flex ratio.

Example: A value of **3** makes "box4" **3x as wide** as the others when `flex-basis: 0`.

```
.box4 { flex: 3 1 0%; }
```



Shortcut flex Values

- **`flex: initial`** (same as `flex: 0 1 auto;`)
  Prevents the item from growing, but allows it to shrink to fit the container
- **`flex: auto`** (same as `flex: 1 1 auto;`)
  Allows items to be fully flexible as needed. Size is based on the width/height properties.
- **`flex: none`** (same as `flex: 0 0 auto;`)
  Creates a completely inflexible item while sizing it to the width/height properties.
- **`flex:` *integer*** (same as `flex:` *integer* `1 0px;`)
  Creates a flexible item with **absolute flex** (so `flex-grow` integer values are applied proportionally)

Changing Item Order

`order`

Values: *Number*

Specifies the order in which a particular item should appear in the flow (independent of the HTML source order):

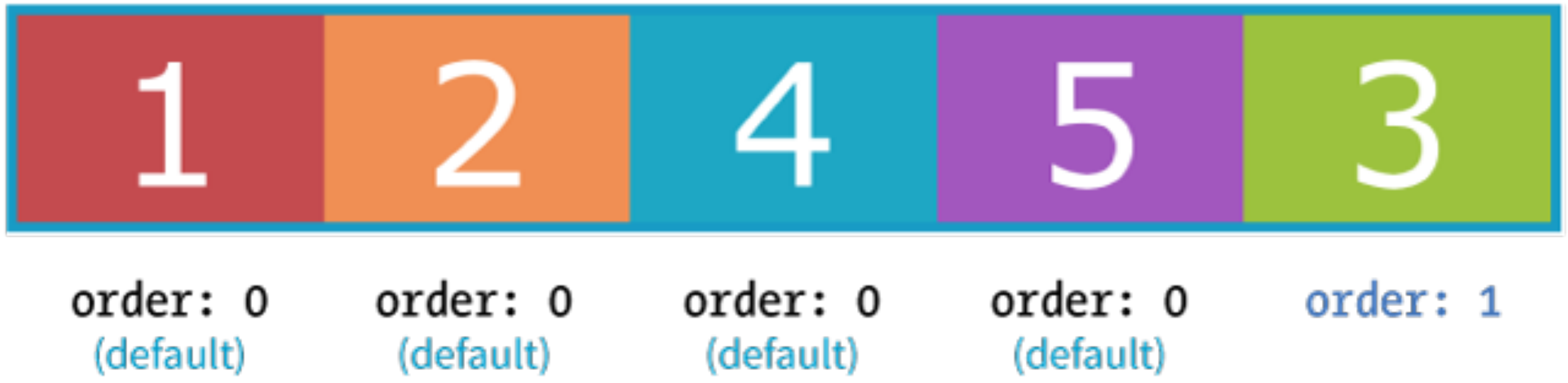
- **`order`** is applied to the **flex item element**.
- The default is 0. Items with the same order value are placed according to their order in the source.
- Items with different order values are arranged from lowest to highest.
- The specific number value doesn't matter; only how it relates to other values (like z-index) matters.

Changing Item Order (cont'd)

Example:

"box3" has a higher order value (1) than the others with default order of 0. It appears last in the line even though it's third in the markup:

```
.box3 {
  order: 1;
}
```



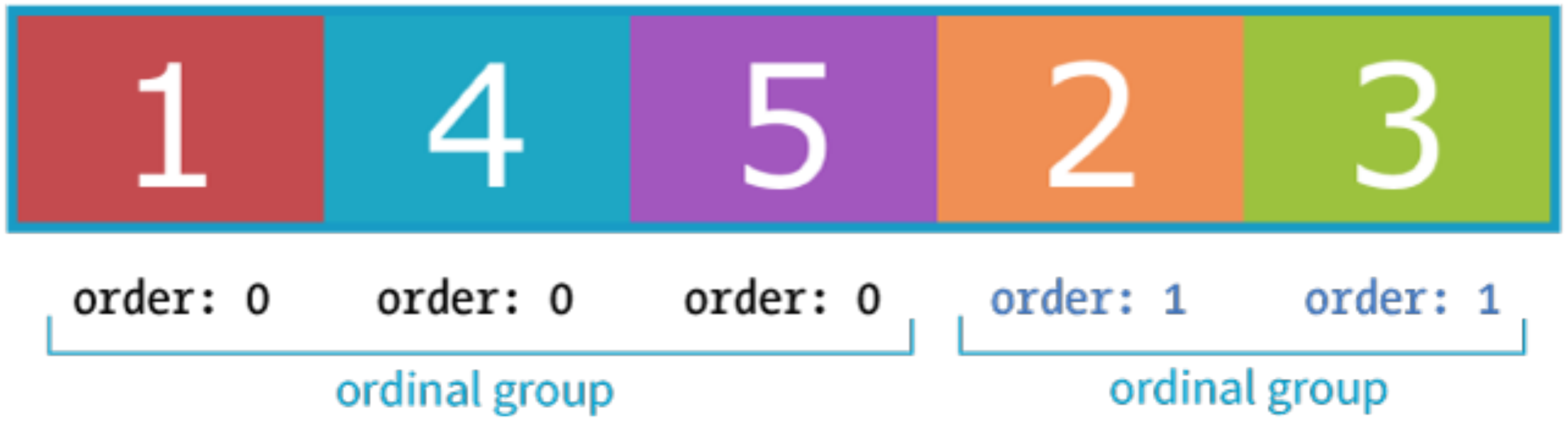
Changing Item Order (cont'd)

Ordinal groups

Items that share the same order value are called an ordinal group.

Ordinal groups stick together and are arranged from lowest value to highest:

```
.box2, .box3 {
  order: 1;
}
```



Browser Support for Flexbox

The Flexbox spec changed over the years and was implemented by browsers along the way:

- **Current version (2012):** `display: flex;`
  Supported by all current desktop and mobile browser versions

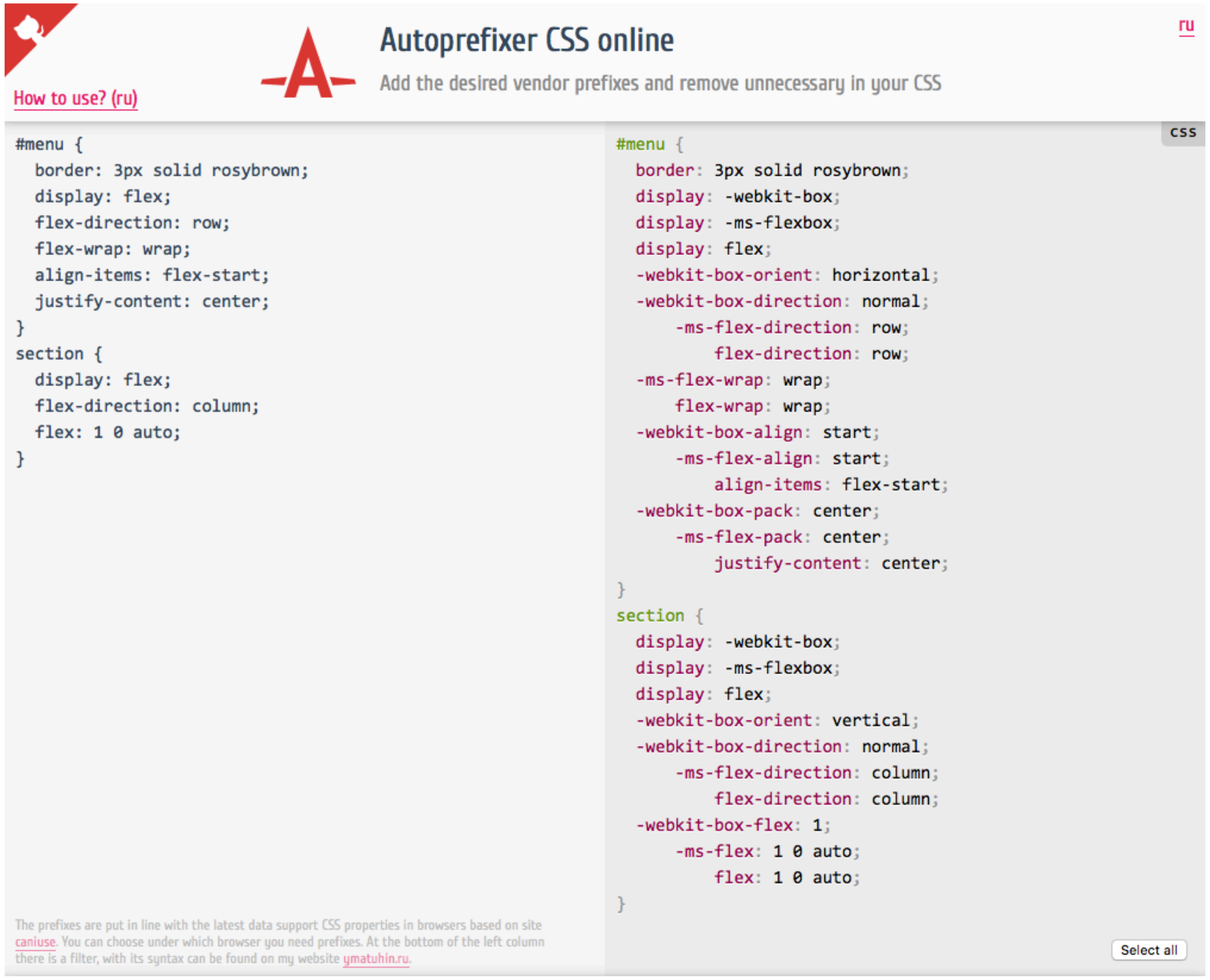
- **“Tweener” version (2011):** `display: flexbox;`
  Supported by IE10 only

- **Old version (2009):** `display: box;`
  Supported by Chrome <21, Safari 3.1–6, Firefox 2–21; iOS 3.2–6.1, Android 2.1–4.3

Browser Support (cont'd)

To ensure that Flexbox works across all supporting browsers, you need a lot of vendor prefixes and redundant declarations.

Use a tool like Autoprefixer to generate all that code for you (autoprefixer.github.io).


Autoprefixer CSS online

Add the desired vendor prefixes and remove unnecessary in your CSS

How to use? (ru)

ru

```
#menu {
  border: 3px solid rosybrown;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-items: flex-start;
  justify-content: center;
}
section {
  display: flex;
  flex-direction: column;
  flex: 1 0 auto;
}
```

The prefixes are put in line with the latest data support CSS properties in browsers based on site caniuse. You can choose under which browser you need prefixes. At the bottom of the left column there is a filter, with its syntax can be found on my website ymatuhin.ru.

```
#menu {
  border: 3px solid rosybrown;
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-orient: horizontal;
  -webkit-box-direction: normal;
      -ms-flex-direction: row;
          flex-direction: row;
  -ms-flex-wrap: wrap;
      flex-wrap: wrap;
  -webkit-box-align: start;
      -ms-flex-align: start;
          align-items: flex-start;
  -webkit-box-pack: center;
      -ms-flex-pack: center;
          justify-content: center;
}
section {
  display: -webkit-box;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-orient: vertical;
  -webkit-box-direction: normal;
      -ms-flex-direction: column;
          flex-direction: column;
  -webkit-box-flex: 1;
      -ms-flex: 1 0 auto;
          flex: 1 0 auto;
}
```

Flexbox Property Review

Flex container properties

```
display
flex-flow
   flex-direction
   flex-wrap
justify-content
align-items
align-content
```

Flex item properties

```
align-self
flex
   flex-grow
   flex-shrink
   flex-basis
order
```