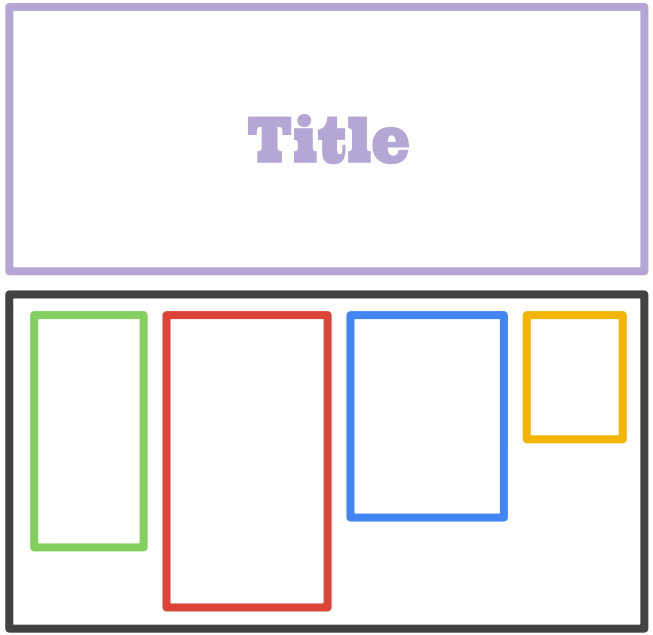
Flexbox

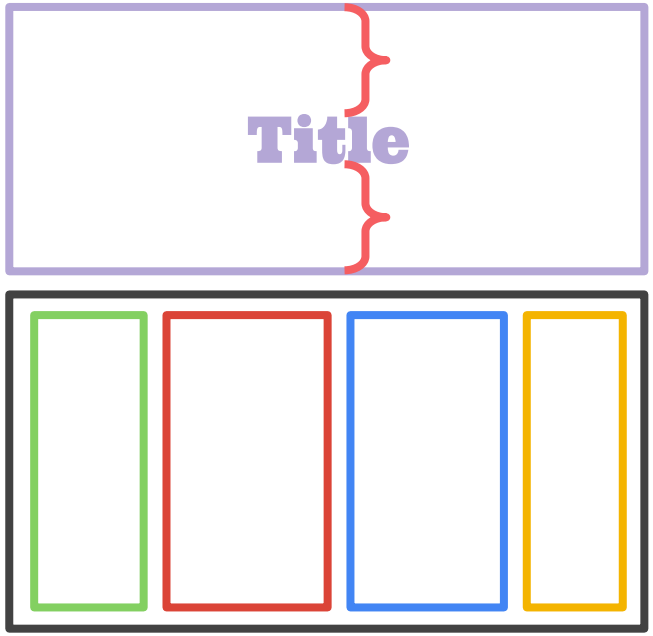
Issue 1: Equal height columns

- The items inside the containers should have the same height

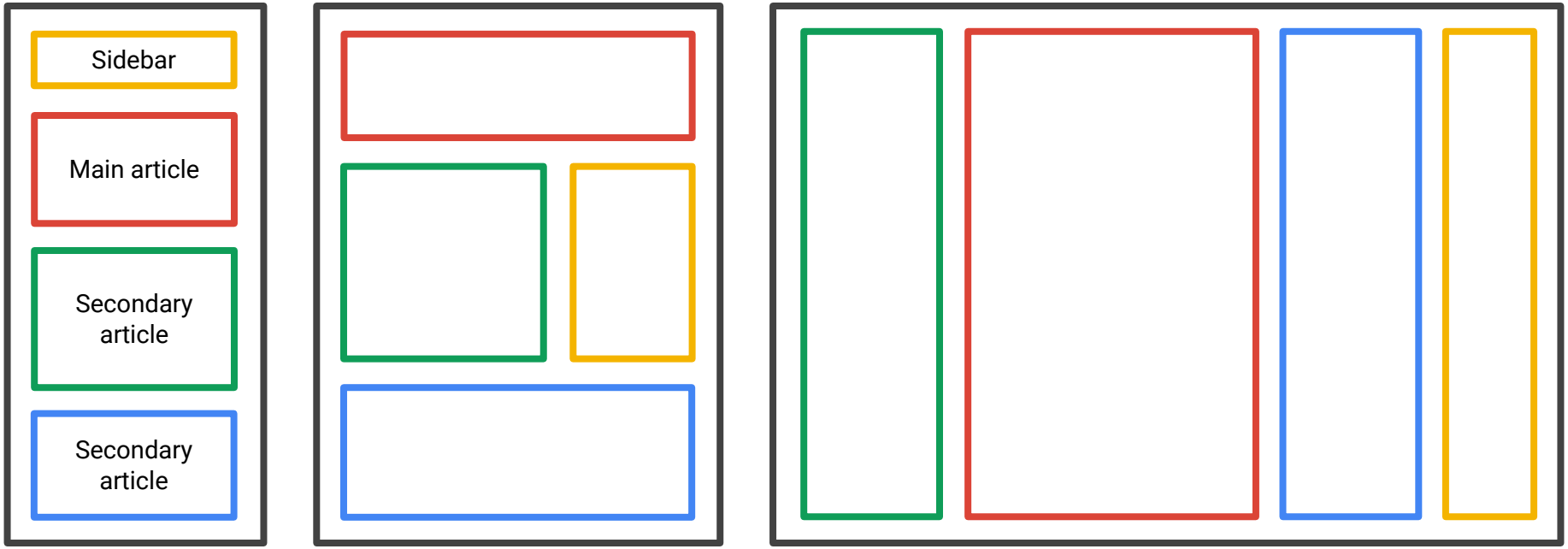


Issue 2: Alignment

- Create a header in the page
  - sibling of the container
  - give it some height
- Position a headline inside: it should be **vertically and horizontally centered**



Issue 3: Reordering



Issues:

1. Equal height columns
2. Alignment
3. Reordering content

Flexbox

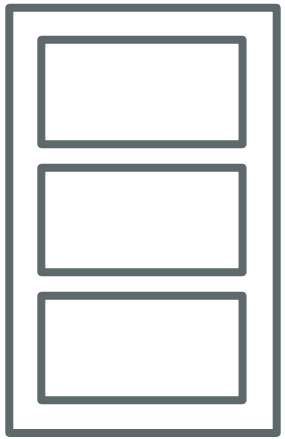
- Flexible boxes, or flexbox
- New layout mode in CSS3
- Elements behave predictably when the page layout changes size
- It does not use floats

Flex container & flex items

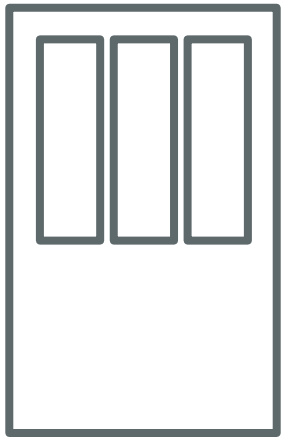
- Flexbox consists of flex containers and flex items.
- A flex container is declared by setting the display property of an element to either
  - **`flex`** (rendered as a block)
  - **`inline-flex`** (rendered as inline).

Flex container & flex items

- Inside a flex container there is one or more flex items.
- Flex items are positioned inside a flex container along a **flex line**.
- By default the flex line is the horizontal one.



`block`



`flex`

Let's solve some issues!

- Make the container a flexbox

display:

- **flex**
- **inline-flex**

Issues:

1. ~~Equal height columns~~
2. Alignment
3. Reordering content

Clean the code

- The content is automatically positioned on the horizontal line: we don't need all the `float` rules!

Flex container

1. flex-direction
2. justify-content
3. align-items
4. flex-wrap
5. align-content

1. Flex direction

Direction of the flexible items inside the flex container

- `row` (default)
- `row-reverse`
- `column`
- `column-reverse`

Responsiveness without media queries! (1)

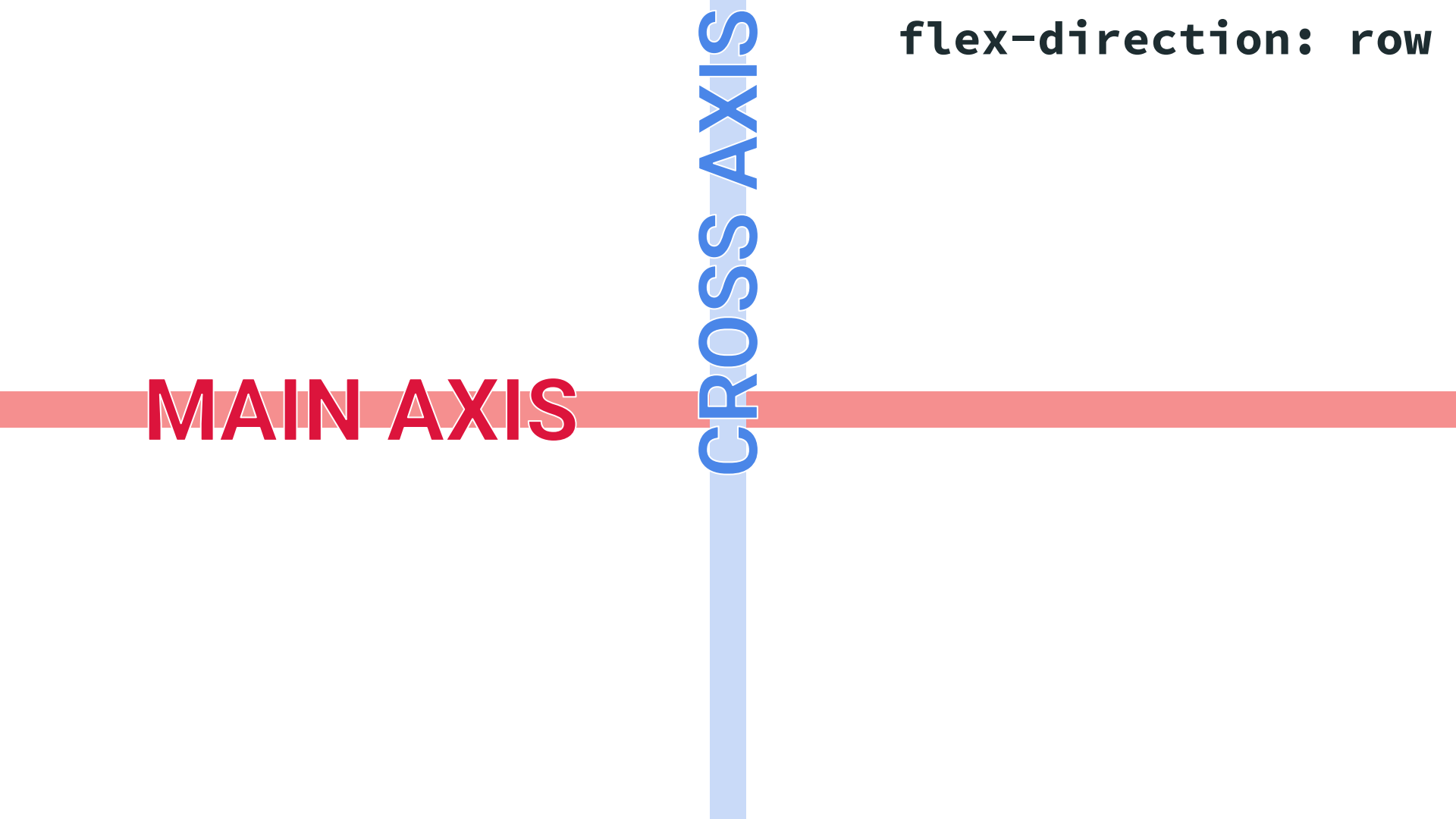
- Define the flex direction to be **column** for the smallest viewport and **row** for the other viewports

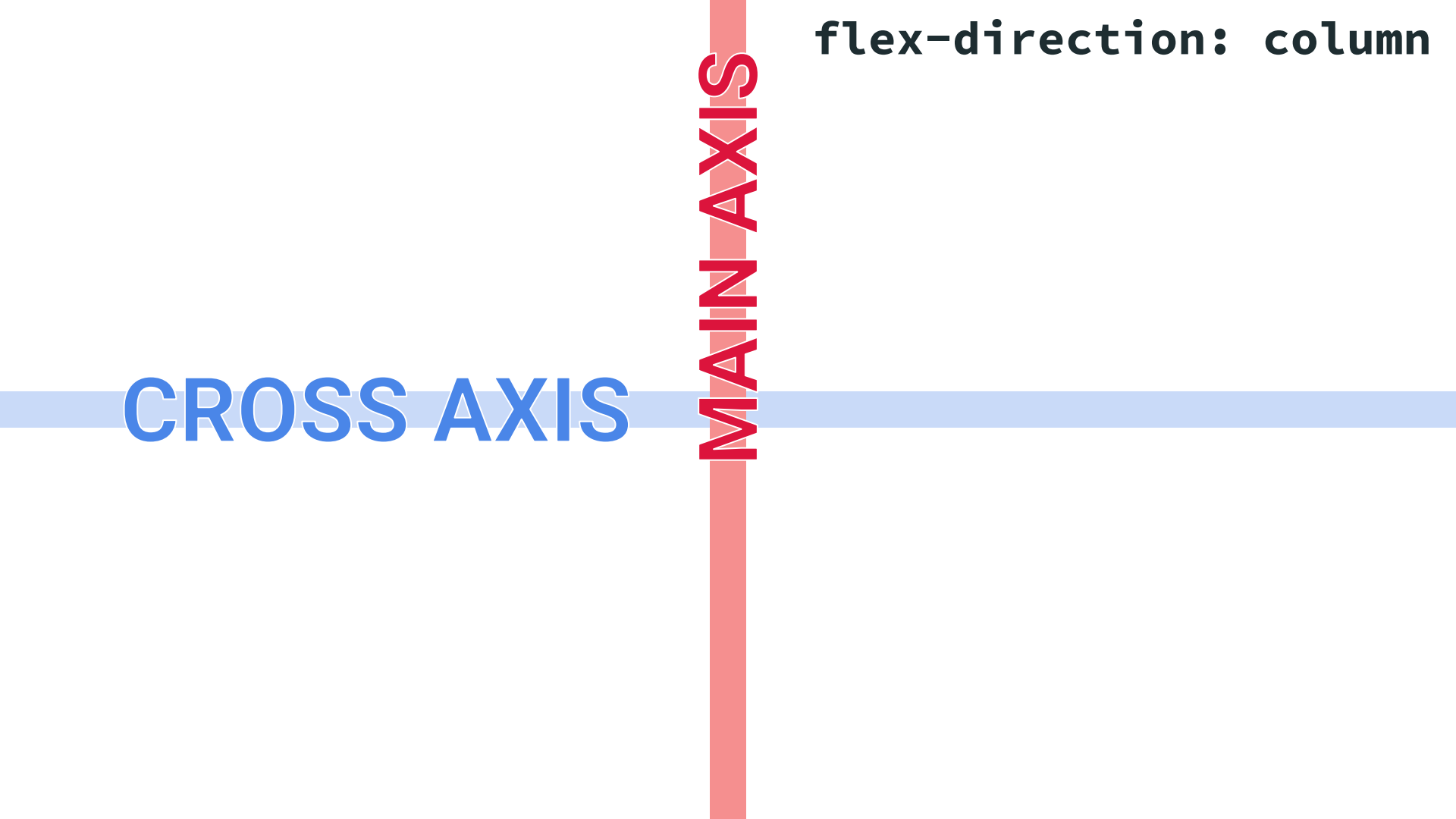
`flex-direction:`

- **`row`** (default)
- **`row-reverse`**
- **`column`**
- **`column-reverse`**

2. Justify content

Horizontally aligns the flex items when the items do not use all available space on the **main-axis**


flex-direction: row
CROSS AXIS
MAIN AXIS


flex-direction: column
MAIN AXIS
CROSS AXIS

2. Justify content

Horizontally aligns the flex items when the items do not use all available space on the **main-axis**

- **`flex-start`** – Default value. Items are positioned at the beginning of the container
- **`flex-end`** – Items are positioned at the end of the container
- **`center`** – Items are positioned at the center of the container
- **`space-between`** – Items are positioned with space between the lines
- **`space-around`** – Items are positioned with space before, between, and after the lines

Justify content

- Create a navigation bar under the header
- Add a list with some navigation items:
  - Politics
  - Business
  - Culture
  - Opinions
  - Sports
  - TV
  - Environment
  - Tech
  - Travel
- Make the list a flexbox and align the items

`justify-content:`

- **`flex-start`** (default)
- **`flex-end`**
- **`center`**
- **`space-between`**
- **`space-around`**

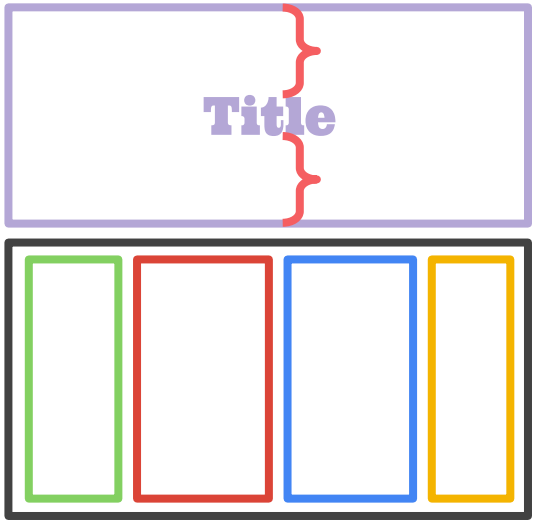
Align items

Vertically aligns the flexible container's items when the items do not use all available space on the cross-axis

- **`stretch`** - Default value. Items are stretched to fit the container
- **`flex-start`** - Items are positioned at the top of the container
- **`flex-end`** - Items are positioned at the bottom of the container
- **`center`** - Items are positioned in the middle of the container
- **`baseline`** - Items are positioned at the baseline of the container

Vertical alignment!

You probably know now how to solve that vertical alignment issue in the title, right?



`align-items:`

- `stretch` (default)
- `flex-start`
- `flex-end`
- `center`
- `baseline`

Issues:

1. ~~Equal height columns~~
2. ~~Alignment~~
3. Reordering content

Flex wrap

specifies whether the flex items should wrap or not, if there is not enough room for them on one **flex line**.

- **`nowrap`** - Default value. The flexible items will not wrap
- **`wrap`** - The flexible items will wrap **if necessary**
- **`wrap-reverse`** - The flexible items will wrap, **if necessary**, in reverse order

Responsiveness without media queries! (2)

- Make the navigation bar responsive without using media queries (hint: the element should wrap on the next line as soon as the viewport is too small)

`flex-wrap:`

- **`nowrap`** (default)
- **`wrap`**
- **`wrap-reverse`**

SHORTHAND!

```
flex-flow: row wrap;
```

wrap

direction

Align content

Modifies the behavior of the **flex-wrap** property. It is similar to **align-items**, but instead of aligning flex items, it aligns **flex lines**.

- **`stretch`** - Default value. Lines stretch to take up the remaining space
- **`flex-start`** - Lines are packed toward the start of the flex container
- **`flex-end`** - Lines are packed toward the end of the flex container
- **`center`** - Lines are packed toward the center of the flex container
- **`space-between`** - Lines are evenly distributed in the flex container
- **`space-around`** - Lines are evenly distributed in the flex container, with half-size spaces on either end

Align content

- Set the **minimum height** of the navigation bar to 60px
- Align the content so that it looks nice when the items wrap

`align-content:`

- `stretch` (default)
- `flex-start`
- `flex-end`
- `center`
- `space-between`
- `space-around`

Flex items

Flex items are defined by three properties:

- **flex-grow**
- **flex-shrink**
- **flex-basis**

Flex-grow

How much the item will grow relative to the rest of the flexible items inside the same container

- 1
- 2
- 3
- 4
- ...

Flex grow

In the jobs section, use flexbox to size the elements of the form

- The input fields and the input button should expand and take all the space
- The input fields should grow **twice as much** as the input button

flex-grow:

- **1**
- **2**
- **3**
- **4**
- **...**

Flex shrink

How the item will shrink relative to the rest of the flexible items inside the same container

- 1
- 2
- 3
- 4
- ...

Flex basis

The initial length of a flexible item

SHORTHAND!

```
flex: 1 0 25%;
```

grow

shrink

basis

Flex

Instead of defining width for each element of the container, use the flex property instead.

```
flex: [grow]
[shrink] [basis];
```

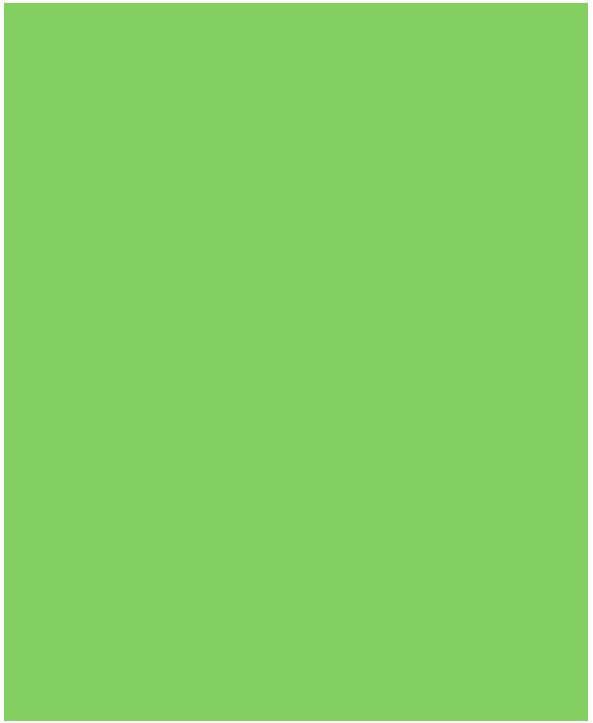
Align self

It makes possible to override the align-items value for a specific flex item

Align self

In the header, add two small white squares

- they should be aligned to the top, while the title is aligned to the middle

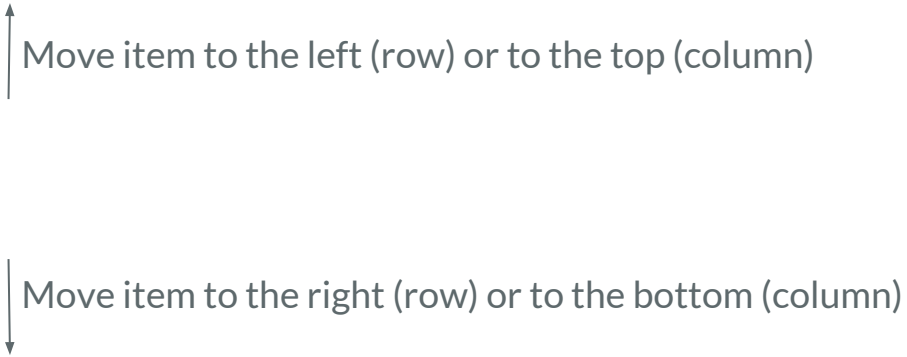


Order

By default, flex items are laid out in the source order.

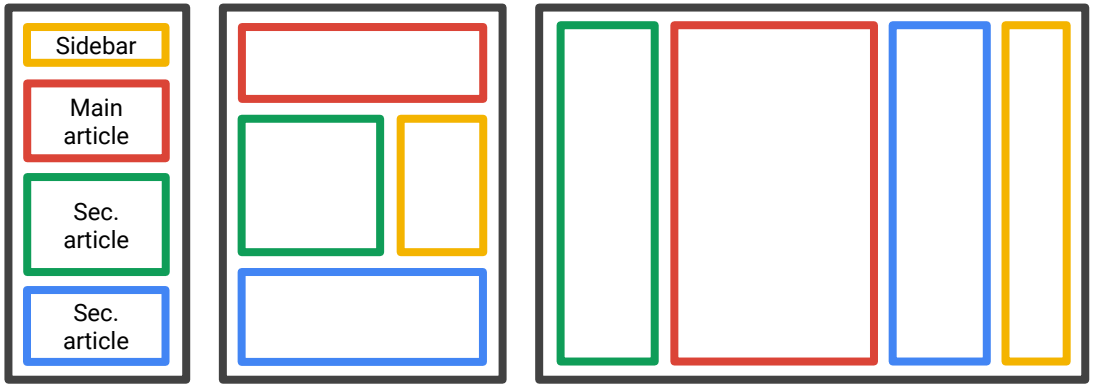
The order property controls the order in which they appear in the flex container.

- `...`
- `-1`
- `0`
- `1`
- `...`



Reordering

You should know how to solve this, now:




```
order: [integer]
```

Issues:

1. ~~Equal height columns~~
2. ~~Alignment~~
3. ~~Reordering content~~

Browser support?