Responsive Layouts

Flex, Grid & Multi-Column

Abdelfattah Ragab

Responsive Layouts

Flex, Grid and Multi-Column

Abdelfattah Ragab

Introduction

Welcome to the book “Responsive Layouts: Flex, Grid and Multi-Column”
In this book I explain the three best-known responsive layouts: the Flexbox, the Grid and the Multi-Column layout.
Flexbox is a one-dimensional layout that only works in one dimension at a time, either horizontally or vertically.
The grid layout is a two-dimensional layout that distributes the elements horizontally and vertically at the same time.
The multi-column layout is a special layout for magazines and newspapers, where the text should flow in columns with spacing, rules, etc.
I'll explain all the properties and their values and how they affect the distribution of elements on the screen.
So let's get started.

Flexbox

Flexbox is a one-dimensional layout that works with one dimension at a time.
To use the flexbox layout, simply set the display property to flex or inline-flex as follows:
display: flex; or display: inline-flex;
If you set the display to flex, it is a block element, if you set it to inline-flex, it is an inline element. This is exactly the same as setting display to block or inline. And what about flex?
The flex influences how the subordinate elements should be distributed on the page.
So we have the flex container, the element with display: flex;
And we have the flex elements, the children of the flex container;
All properties that you apply to the container also affect the subordinate elements.

The container has its own properties and the children have their own properties and here's a quick overview of them all.

The container properties

- `display:` flex or display: inline-flex: Specifies that the container is a flex container and its children are flex items.
- `flex-direction:` Defines the direction of the main axis, which determines how flex items are positioned. Values can be row, row-reverse, column, or column-reverse.
- `justify-content:` Aligns flex items along the main axis. It controls the spacing between and around the flex items. Values include flex-start, flex-end, center, space-between, space-around, and space-evenly.
- `align-items:` Aligns flex items along the cross axis (perpendicular to the main axis). It controls how flex items are distributed vertically. Values can be flex-start, flex-end, center, baseline, or stretch.
- `flex-wrap:` Specifies whether flex items should wrap to multiple lines when they exceed the width of the flex container. Values include nowrap, wrap, and wrap-reverse.

- `align-content:` Defines the alignment of flex lines when there is extra space on the cross axis. It applies to multi-line flex containers. Values can be flex-start, flex-end, center, space-between, space-around, or stretch.
- `gap`: Defines the space between the flex elements

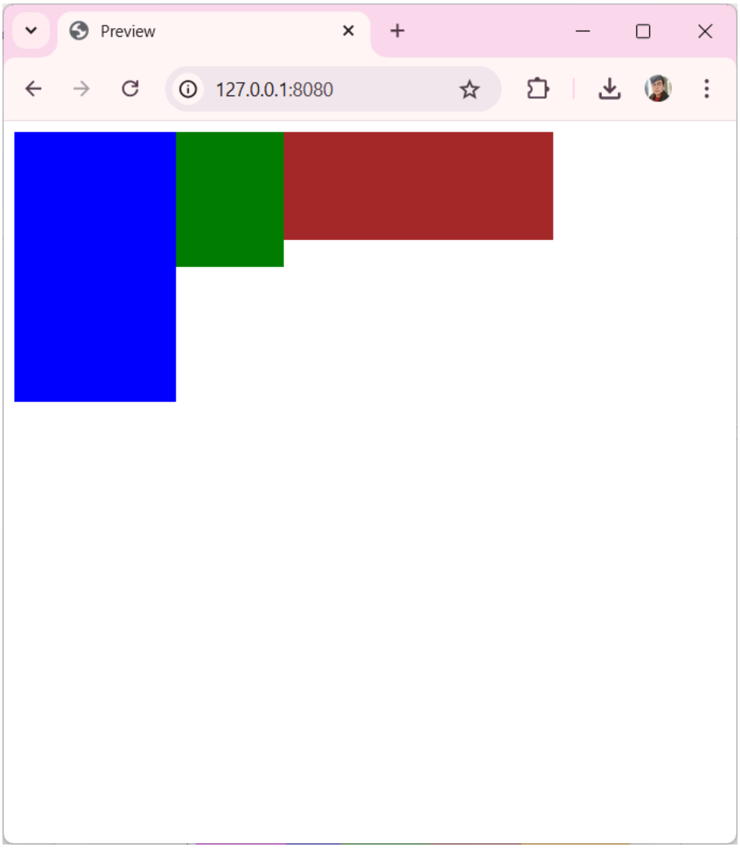
The Item properties

- `flex-grow:` Specifies the ability of a flex item to grow to fill available space. It determines how the remaining space is distributed among flex items. Default value is 0.
- `flex-shrink:` Specifies the ability of a flex item to shrink if necessary when the flex container is too small. Default value is 1.
- `flex-basis:` Specifies the initial size of a flex item before it's distributed in the flex container. Values can be a length, a percentage, or auto.
- `flex:` Shorthand for flex-grow, flex-shrink, and flex-basis.
- `order:` Defines the order in which flex items are displayed. Lower values appear first. Default value is 0.

- `align-self:` Overrides the align-items property for a specific flex item. It allows you to align an item along the cross axis independently. Values can be auto, flex-start, flex-end, center, baseline, or stretch.

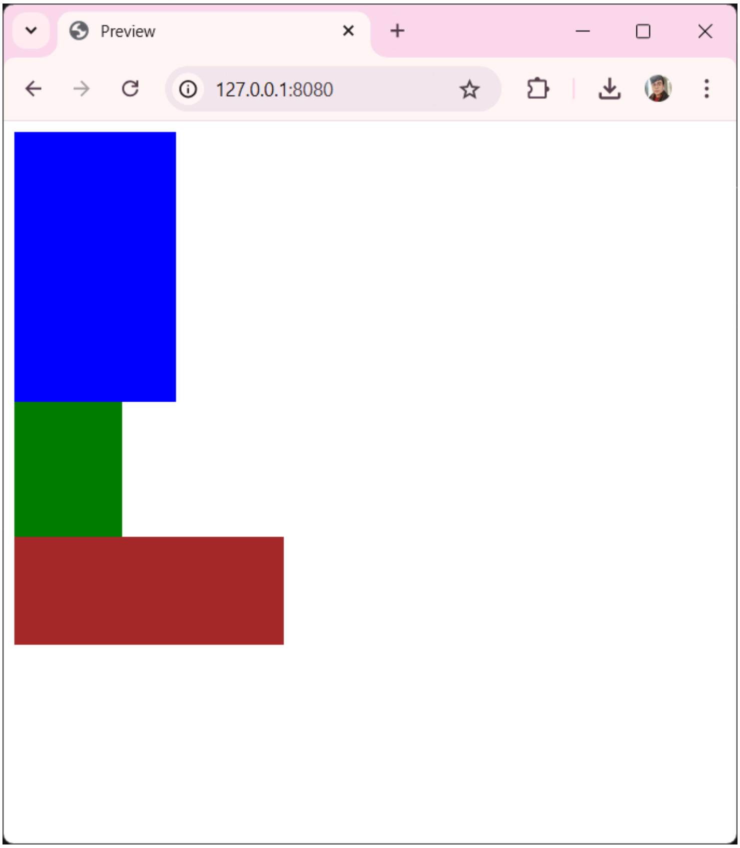
Example

We have 3 div elements with different sizes and colors as follows. By setting display flex in their wrapper, they are displayed as follows:



```
<style>
  .wrapper {
    display: flex;
  }
  .div-1 {
    width: 120px;
    height: 200px;
    background-color: blue;
  }
  .div-2 {
    width: 80px;
    height: 100px;
    background-color: green;
  }
  .div-3 {
    width: 200px;
    height: 80px;
    background-color: brown;
  }
</style>
<div class="wrapper">
  <div class="div-1"></div>
  <div class="div-2"></div>
  <div class="div-3"></div>
</div>
```

Before applying display flex to the wrapper, they looked like this



Now let’s understand how flex layout works.

Flex layout has two main axes, the main axis and the cross axis

The main axis is defined by the flex-direction property, and the cross axis is perpendicular to it.
flex-direction can have four values: row, column, row-reverse and column-reverse.
if flex-direction is not specified, it is set to row by default

flex-direction

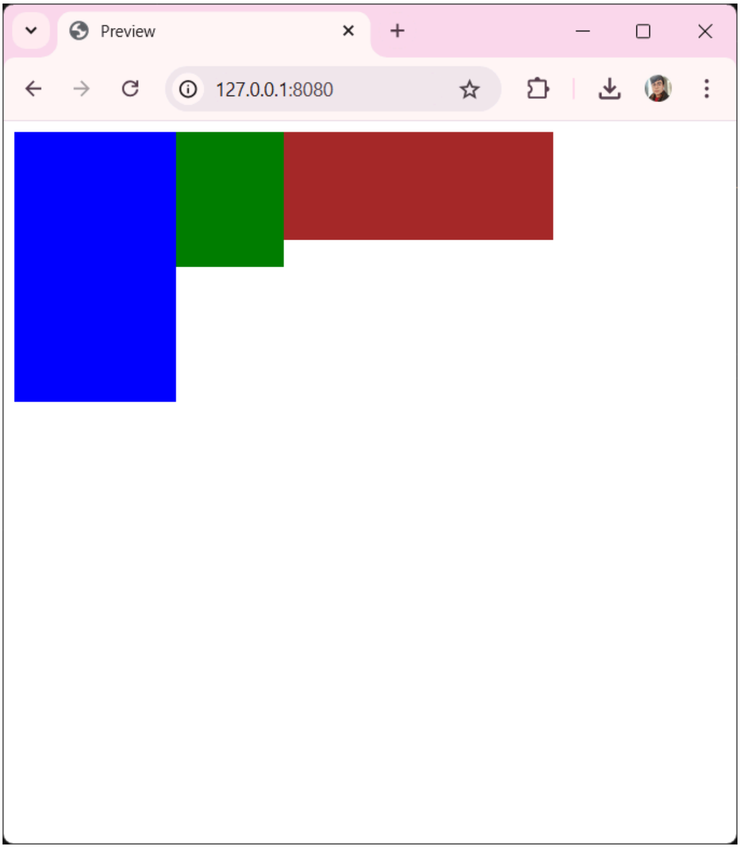
The flex-direction property specifies the direction of the flexible elements.
If the element is not a flexible element, the flex-direction property has no effect.

Values

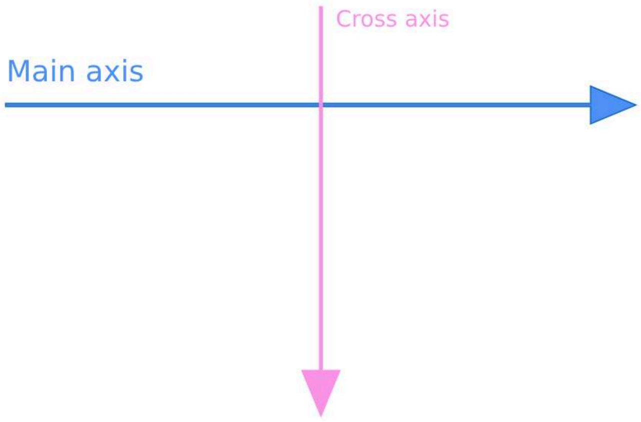
- row
- row-reverse
- column
- column-reverse

row

Default value. The flexible items are displayed horizontally, as a row



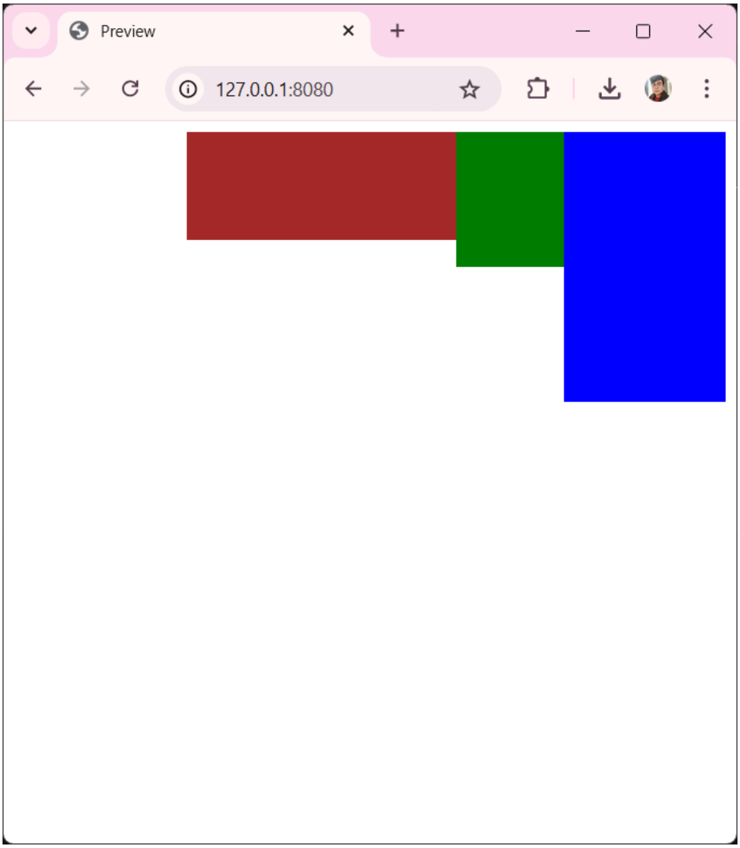
```
.wrapper {
  display: flex;
  flex-direction: row;
}
```



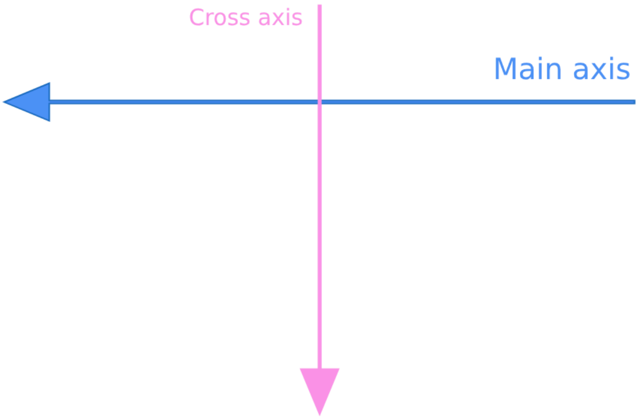
flex-direction: row;

row-reverse

Same as row, but in reverse order



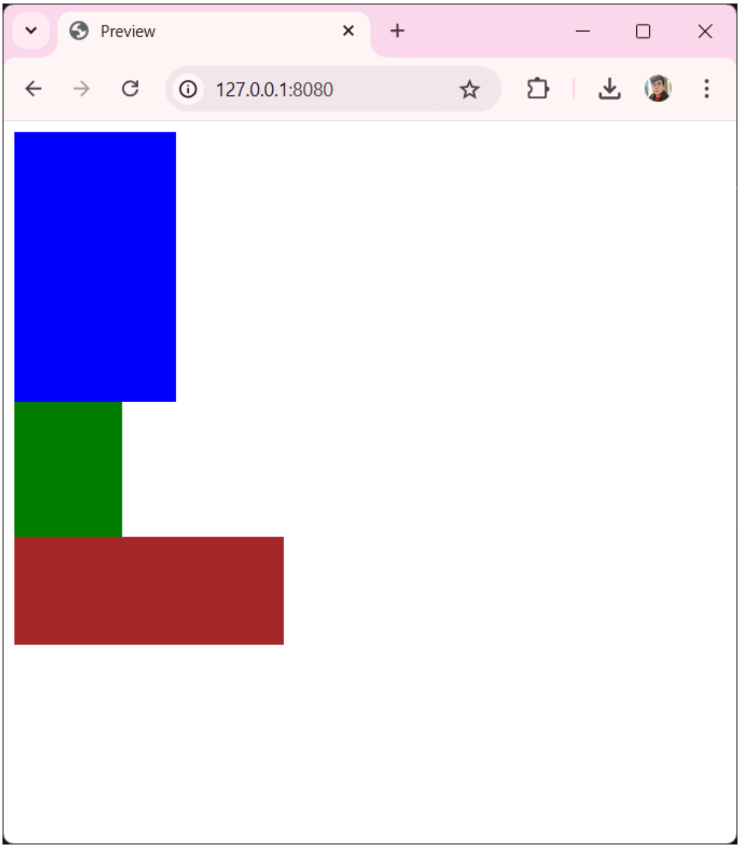
```
.wrapper {
  display: flex;
  flex-direction: row-reverse;
}
```



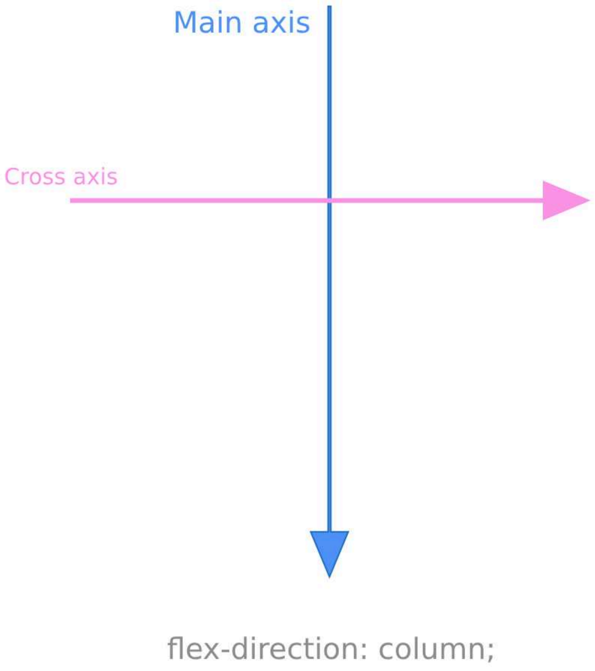
flex-direction: row-reverse;

column

The flexible items are displayed vertically, as a column

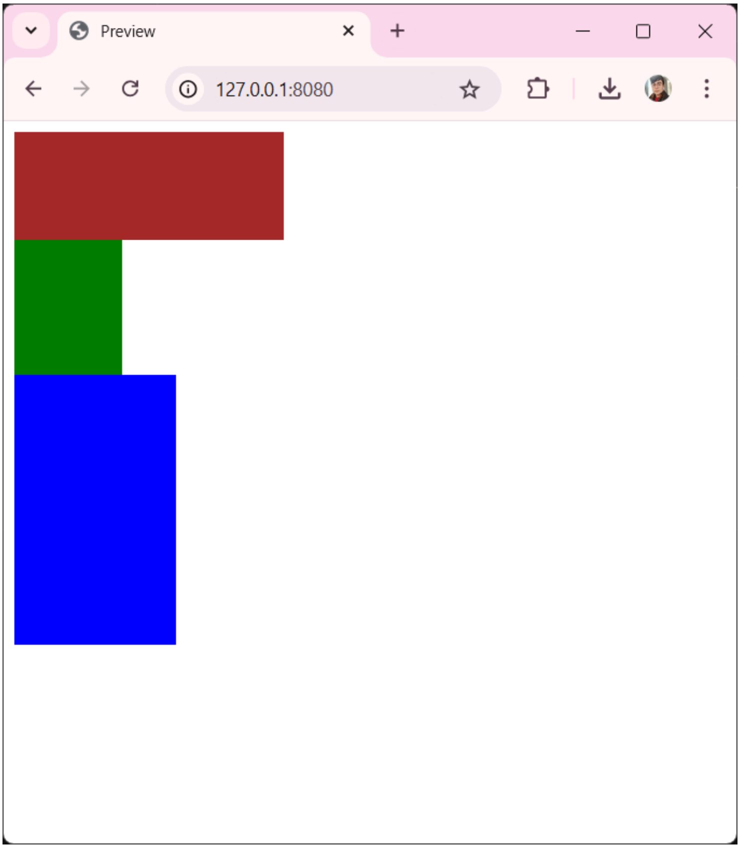


```
.wrapper {
  display: flex;
  flex-direction: column;
}
```

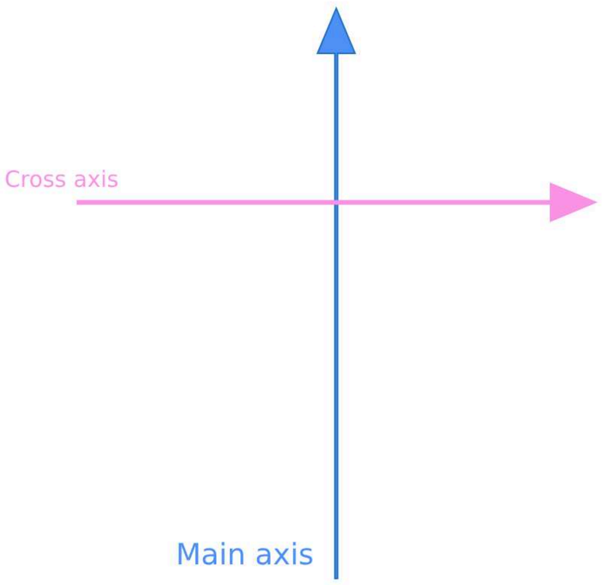


column-reverse

Same as column, but in reverse order



```
.wrapper {
  display: flex;
  flex-direction: column-reverse;
}
```



flex-direction: column-reverse;

Now let's look at how you can benefit from this, starting with two important flex layout properties justify-content and align-items.
I will explain the flex direction of the row in detail and leave it to you to try the other flex directions

justify-content

The CSS property justify-content determines how the browser distributes the space between and around content elements along the main axis of a flex container

Values

- flex-start
- flex-end
- center
- space-between
- space-around
- space-evenly

Let’s explain each value and see it in action
I'll start with the default value, flex-start

flex-start

Default value. Items are positioned at the beginning of the container