# Main and cross axes (OPTIONAL)

# Main and cross axes (OPTIONAL)

**Note**: This material is included for completeness. However, many are able to use flexbox satisfactorily without it. None of the material here will appear in any graded question.

## Concepts

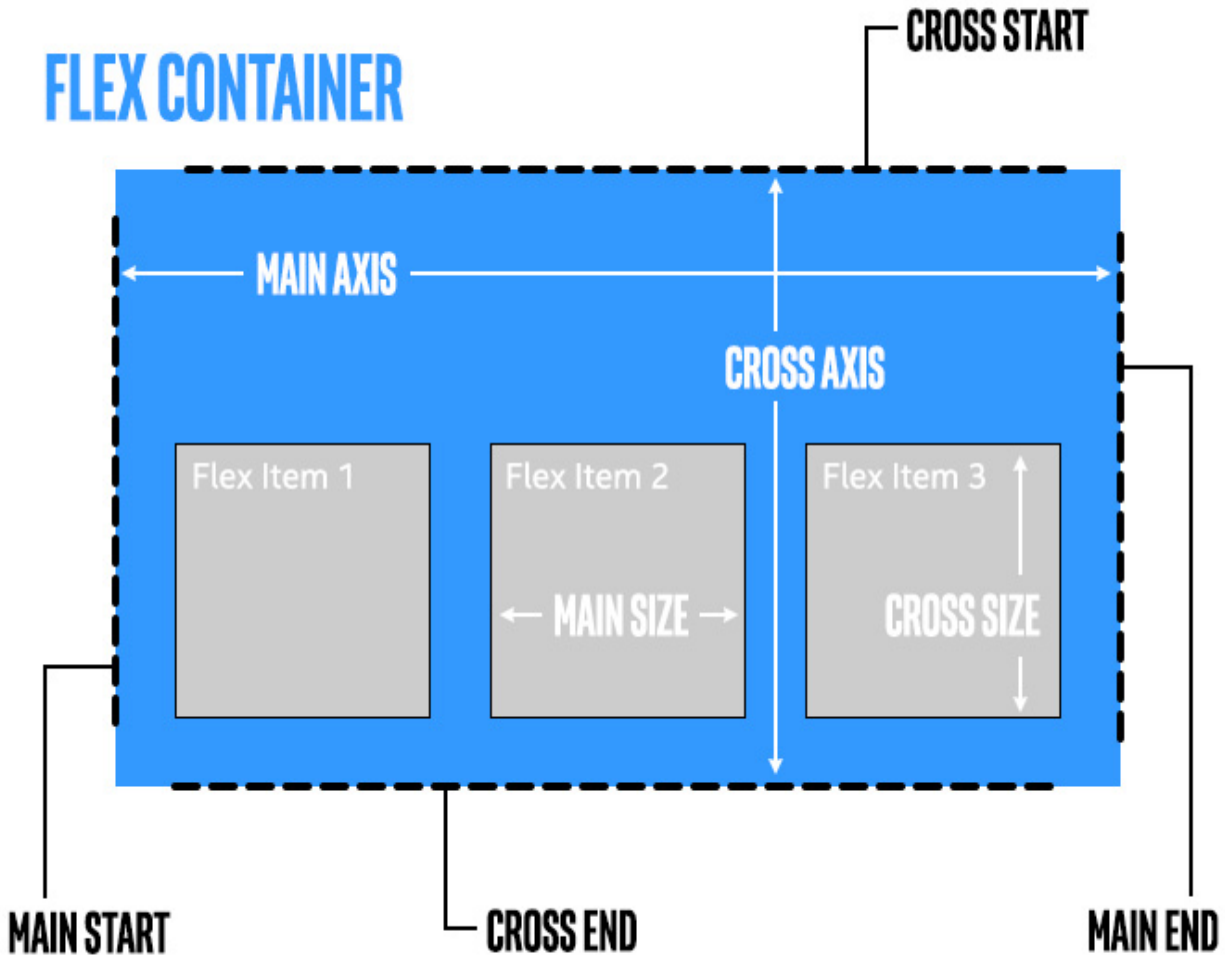
Before we step deeper into flexbox, there are a few concepts we should make sure to understand.

### Main axis and cross axis

Every flexbox has two axes. The "main" axis is the major axis along which all the flex items are being laid. So, when the `flex-direction` is `row`, the main axis is horizontal, running left to right. When the `flex-direction` is `column`, the main axis is vertical, running top to bottom. (Quick quiz: what if the `flex-direction` is `row-reverse`?).

The "cross" axis runs perpendicular to the main axis. It is the direction that items might wrap. So with `flex-flow: row wrap;` the cross axis is vertical and runs top to bottom. And with `flex-flow:row wrap-reverse;` it is vertical running bottom to top.

### Start and end




In the illustration above, we see the main and cross axes as they would be for `flex-flow:row wrap;`. And in the same illustration, we also see the start and end points for both the main and cross axes. Take a moment and visualize how both the axes *and* the start and end points would change for each of these combinations for `flex-flow` :

- `row wrap`
- `row wrap-reverse`
- `row-reverse wrap`
- `row-reverse wrap-reverse`
- `column wrap`
- `column wrap-reverse`
- `column-reverse wrap`

- `column-reverse wrap-reverse`

## main axis for sizing, cross axis for alignment

The terms *"main"* and *"cross"* appear in the descriptions of flexbox and in multiple online tutorials you might find. However, they are **not** used in the names of any CSS properties or values. The following properties control behavior along the main axis. We are already familiar with all of them, except `justify-content`.

- `flex`
- `flex-grow`
- `flex-shrink`
- `flex-basis`
- `justify-content`

All the properties above control how a flex item might be *sized*, except `justify-content`, which controls how a flexbox container spaces out and positions flex items.

And, obversely, the following list of properties controls behavior along the cross axis:

- `align-content`
- `align-items`
- `align-self`

These properties all govern how a flex item might be *aligned* or positioned along the cross axis. They also support a simple `stretch` value, which we will see when we cover them.

**Important**: Flexbox items can have their size and position influenced on the main axis, with `flex-grow`, and others. But on the cross axis, with the exception of a coarse `stretch` option, we can only influence their position. This distinction is very important. In the cross axis direction, our ability to influence the sizing is limited. When not using `stretch`, we get the normal sizing behavior (block level elements take vertical size of content, etc.) or use the regular sizing properties (`min-height`, `max-height`, `height`, etc.)

## flex-start and flex-end are contextual

We will begin to see the values `flex-start` and `flex-end` in the next section as we look at alignment and justification. Just as the terms "main" and "cross" do not appear as CSS terms, be aware that `flex-start` and `flex-end` are contextual. When used with the `justify-content` property, `flex-start` and `flex-end` refer to the "main start" and "main end" sides (as in the illustration above). When used with any of the flexbox align properties, `flex-start` and `flex-end` refer to the "cross start" and "cross end" sides.