

Intro to HTML + CSS

HTML - HyperText Markup language

HTML Structure

HTML is built using `elements`. Some common elements are:

`div`

→ a container of sorts

`span`

→ an inline container

`h1`, `h2`, `h3`, `h4`, `h5`, `h6`

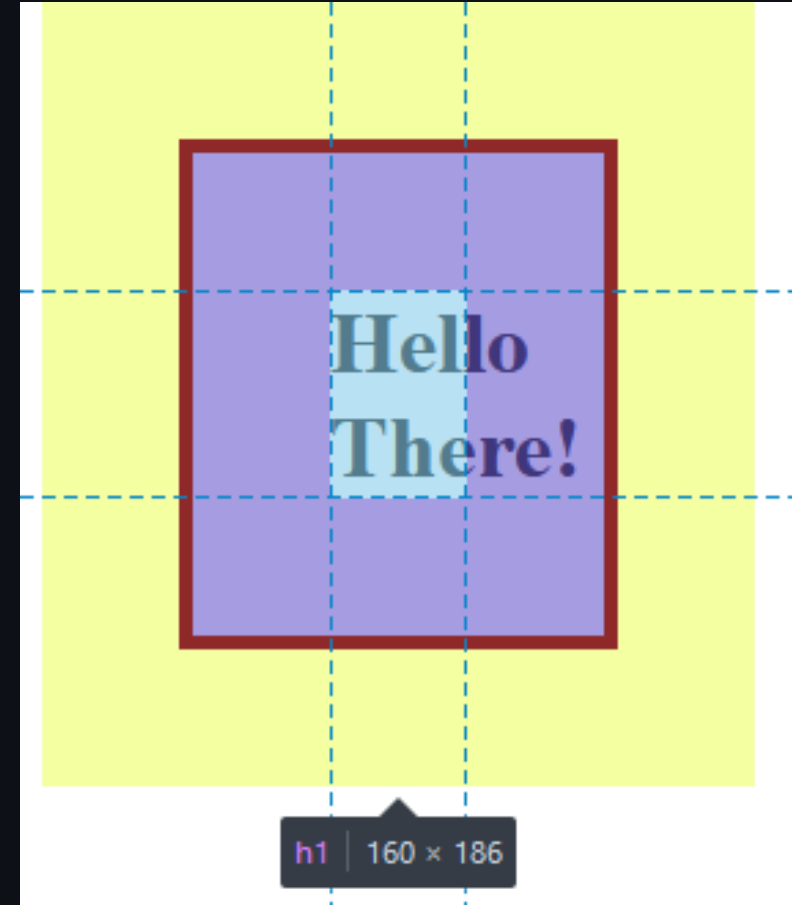
→ headings

Everything is a box... (mostly)

Every HTML element has a rectangular shape. **A box.**

Each box has 4 regions.

- Margin
- Border
- Padding
- Content



CSS - Cascading Style Sheets

Why CSS?

CSS allows us to style our webpages by changing the *properties* of certain HTML elements.

```
/* below is a CSS selector - it selects HTML elements */
body {
  background-color: lightblue; /* this is a CSS property */
}

h1 {
  color: white; /* this is a CSS property */
  text-align: center; /* this is a CSS property */
}

p {
  font-family: verdana;
  font-size: 20px;
}
```

Flexbox

Everything is a box (cont'd)

Below are 3 `<div>` elements, height `100px` with a `1px solid black` border.

Hello there!

Hello there!

Hello there!

Everything is a box (cont'd)

Below are 3 `<h1>` elements, with a `1px solid red` border.

Hello there!

Hello there!

Hello there!

Note how each element fills up all the horizontal space.

Flexbox

An element with `display: flex;` styling forces all its children to take up the remaining horizontal space.

Below are 3 `h3` elements inside a `flexbox` `div` element. Note how taller elements push the flexbox outwards



Diagram illustrating a flexbox container (blue border) containing three `h3` elements (red borders). All three elements are of equal height and width, demonstrating how they share the available horizontal space equally.

Hello there!Hello there!Hello there!



Diagram illustrating a flexbox container (blue border) containing three `h3` elements (red borders). The middle element is significantly taller than the other two. The taller element pushes the other two outwards, making them wrap to the next line, demonstrating how taller elements affect the layout.

Hello there!Hello there!Hello there!

Flexbox (cont'd)

Hello there!Hello there!Hello there!Hello there!Hello there!Hello there!

Flexbox will by default always preserve fit everything into the available horizontal space. Any overflow is still inserted and the width of each element is reduced, kind of like compressing. This makes your elements more **responsive**

[illegible][illegible]

Flex Wrap

We can wrap this overflow using `flex-wrap: wrap;`

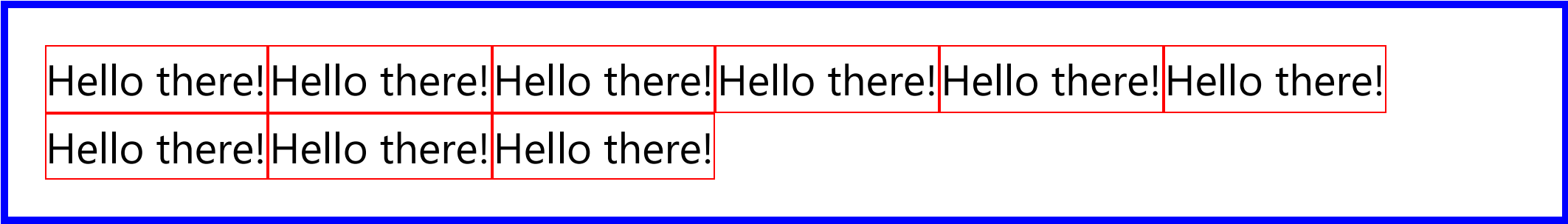
| | | | | | | | | | | | | | | | |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Hello | Hello | Hello | Hello | Hello | Hello | Hello | Hello | Hello | Hello | Hello | Hello | Hello | Hello | Hello | Hello |
| there! | there! | there! | there! | there! | there! | there! | there! | there! | there! | there! | there! | there! | there! | there! | there! |

`flex-wrap` wraps any element that will overflow **content box**.

| | | | | | | |
|--------------|--------------|--------------|--------------|--------------|--------------|--|
| Hello there! | Hello there! | Hello there! | Hello there! | Hello there! | Hello there! | |
| Hello there! | Hello there! | Hello there! | Hello there! | Hello there! | Hello there! | |
| Hello there! | Hello there! | Hello there! | | | | |

Flex Direction

The `flex-direction` dictates the direction of which elements 'queue up'.
By default, this direction is set to `row` or a horizontal order.



| | | | | | |
|--------------|--------------|--------------|--------------|--------------|--------------|
| Hello there! | Hello there! | Hello there! | Hello there! | Hello there! | Hello there! |
| Hello there! | Hello there! | Hello there! | | | |

flex-direction: column

Hello there!

Hello there!

Hello there!

Hello there!

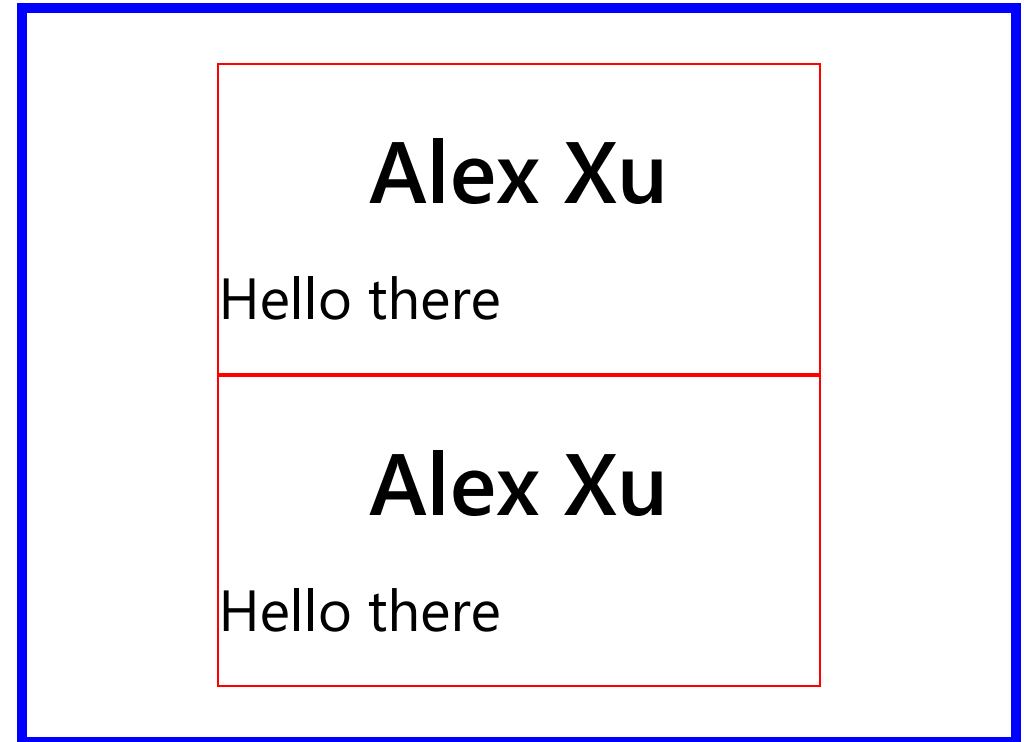
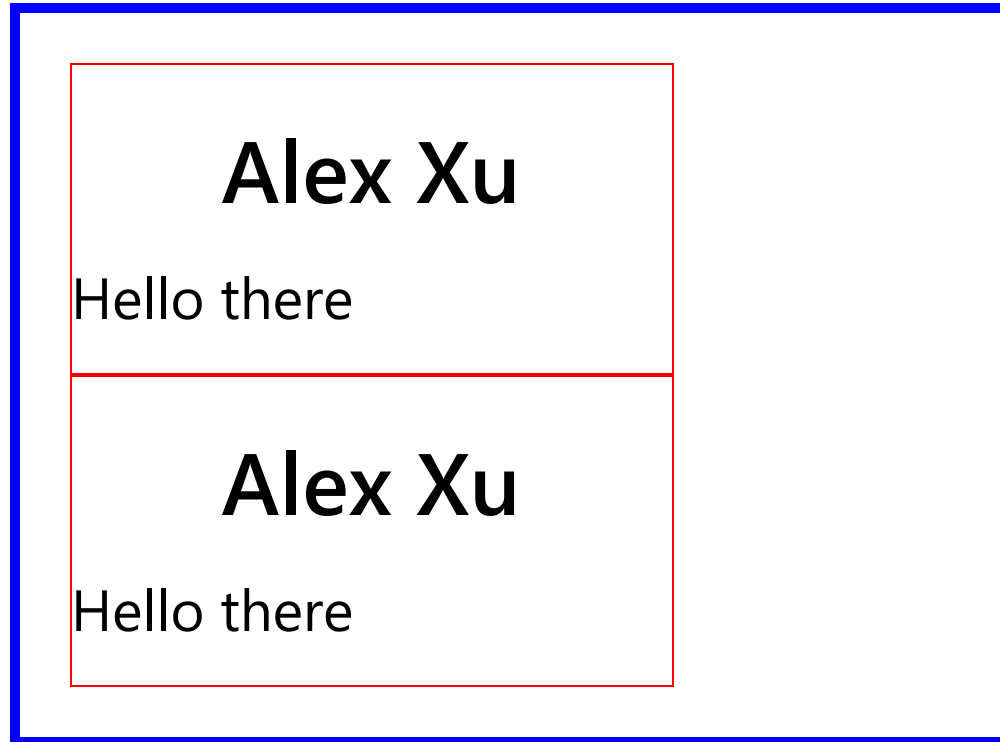
Hello there!

Hello there!

flex-direction: column (cont'd)

You might think `flex-direction: column;` just looks like normal HTML.

But its use case is for aligning more complex, custom elements, especially ones which have a limited width:



alignment and justification

When using flexbox, you can align and justify the content of the flexbox.

A flexbox has **two** axes:

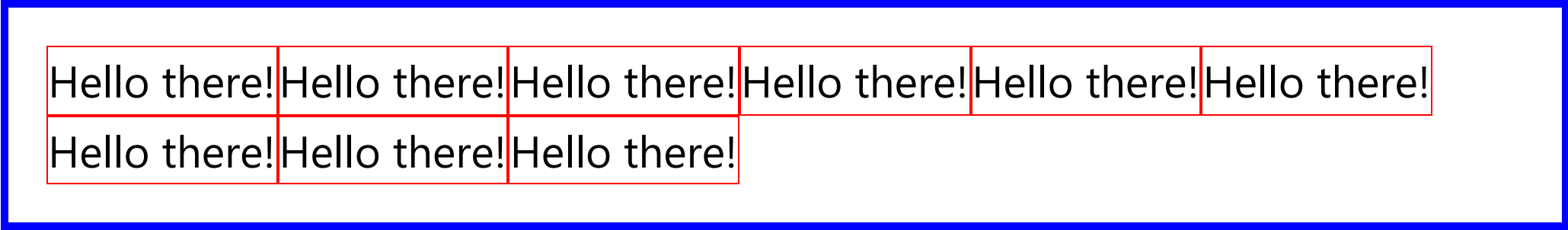
- the main axis
- the cross axis

The **main** axis will always be in the direction given by `flex-direction`

The **cross** axis will always be the opposite direction given by `flex-direction`

alignment and justification (cont'd)

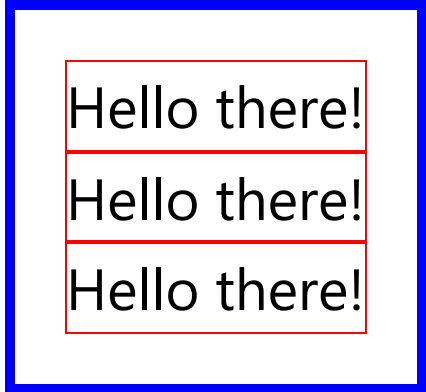
<----- main axis: flex-direction: row ----->



A diagram showing a flex container with a blue border. Inside, there are two rows of text. The first row contains six 'Hello there!' items, each in a red-bordered box. The second row contains three 'Hello there!' items, each in a red-bordered box, followed by empty space. A vertical red line is on the left side of the container.

| | | | | | |
|--------------|--------------|--------------|--------------|--------------|--------------|
| Hello there! | Hello there! | Hello there! | Hello there! | Hello there! | Hello there! |
| Hello there! | Hello there! | Hello there! | | | |

<----- cross axis: flex-direction: column ----->

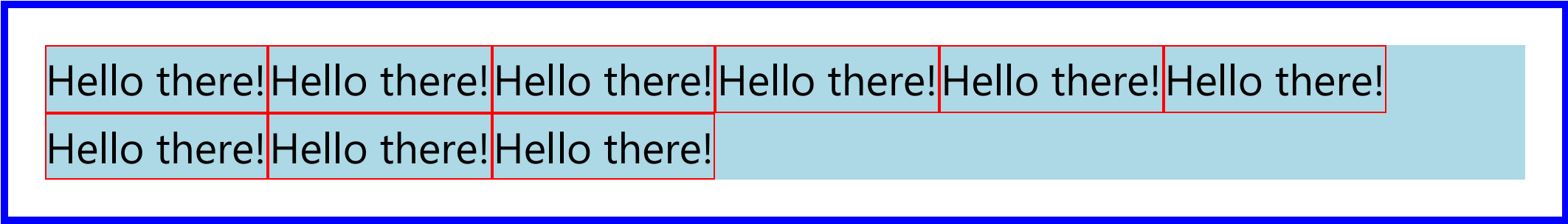


A diagram showing a flex container with a blue border. Inside, there are three 'Hello there!' items, each in a red-bordered box, stacked vertically. A vertical black line is on the left side of the container.

| |
|--------------|
| Hello there! |
| Hello there! |
| Hello there! |

content vs items

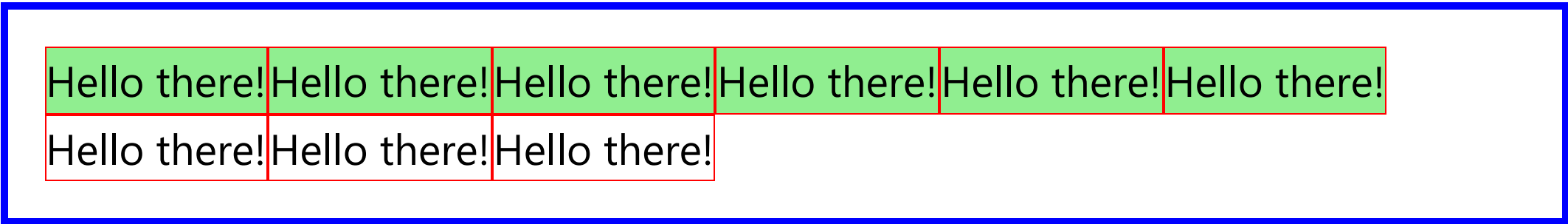
`content` refers to everything inside a flexbox, highlighted in light blue:



A diagram of a flexbox represented by a blue rectangular border. Inside, there are two rows of text. The first row contains six 'Hello there!' strings, each enclosed in a red rectangular box. The second row contains three 'Hello there!' strings, each in a red box, followed by a large, empty light blue rectangular area that fills the rest of the flexbox. This light blue area represents the 'content' of the flexbox.

| | | | | | |
|--------------|--------------|--------------|--------------|--------------|--------------|
| Hello there! | Hello there! | Hello there! | Hello there! | Hello there! | Hello there! |
| Hello there! | Hello there! | Hello there! | | | |

`items` refers to everything inside a flexbox **line**, highlighted in light green:



A diagram of a flexbox represented by a blue rectangular border. Inside, there are two rows of text. The first row contains six 'Hello there!' strings, each enclosed in a red rectangular box. The second row contains three 'Hello there!' strings, each in a red box, followed by a large, empty white rectangular area that fills the rest of the flexbox. The first row of red boxes is highlighted with a light green background, representing the 'items' of the flexbox.

| | | | | | |
|--------------|--------------|--------------|--------------|--------------|--------------|
| Hello there! | Hello there! | Hello there! | Hello there! | Hello there! | Hello there! |
| Hello there! | Hello there! | Hello there! | | | |

content vs items (cont'd)

`content` refers to everything inside a flexbox, highlighted in light blue:

Hello there!

Hello there!

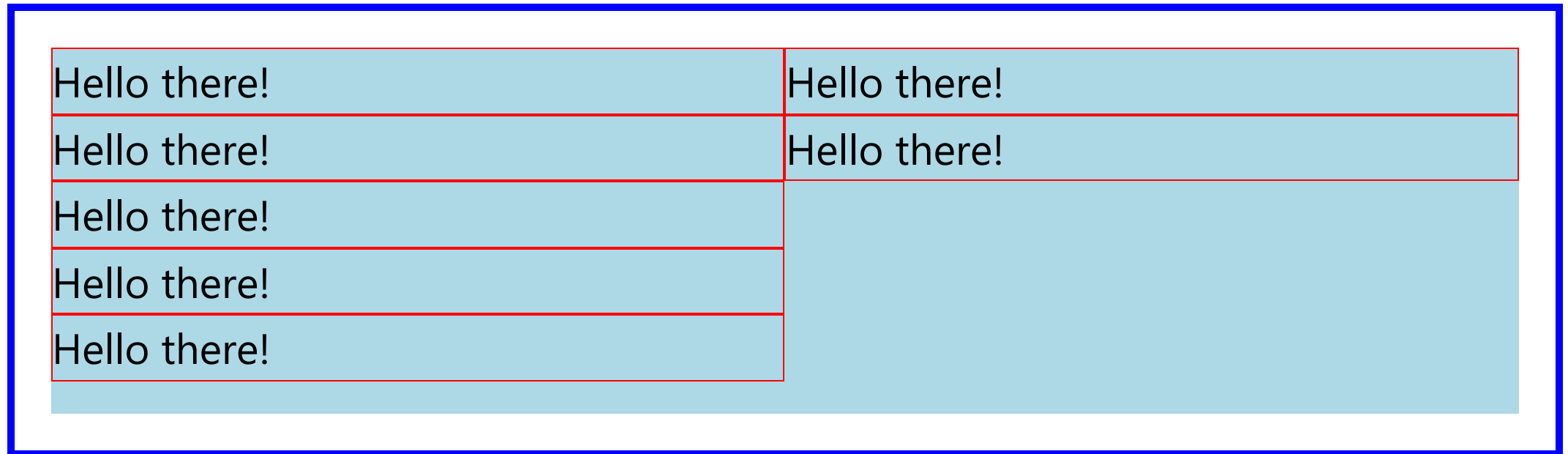
Hello there!

Hello there!

Hello there!

Hello there!

Hello there!



content vs items (cont'd)

`items` refers to everything inside a flexbox **line**, highlighted in light green:

Hello there!

Hello there!

Hello there!

Hello there!

Hello there!

Hello there!

Hello there!


Hello there!

Hello there!

justify

The `justify` prefix refers to the main direction. It always refers to the **main axis**.

```
flex-direction: row & justify-content: center
```



Hello there! Hello there! Hello there! Hello there! Hello there! Hello there!

Hello there! Hello there! Hello there!

justify (cont'd)

`flex-direction: column` & `justify-content: center`

Hello there!

Hello there!

Hello there!

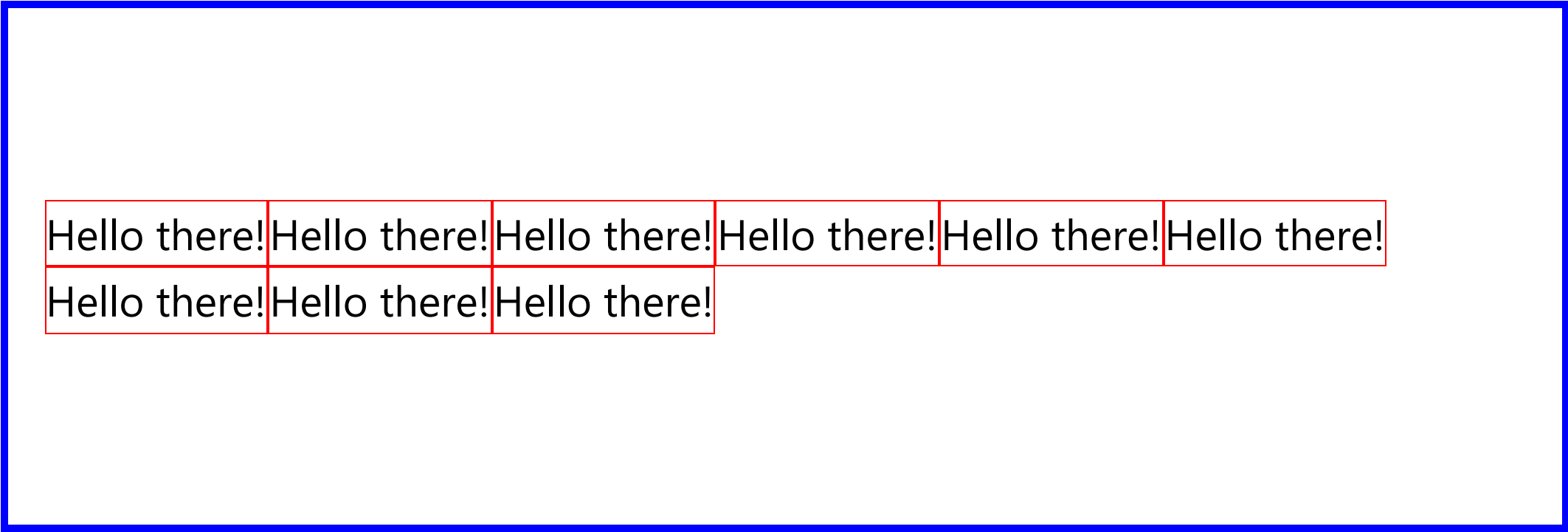
Hello there!

Hello there!

align

The `align` prefix refers to the cross direction. It always refers to the **cross axis**.

`flex-direction: row` & `align-content: center`



| | | | | | |
|--------------|--------------|--------------|--------------|--------------|--------------|
| Hello there! | Hello there! | Hello there! | Hello there! | Hello there! | Hello there! |
| Hello there! | Hello there! | Hello there! | | | |

align (cont'd)

`flex-direction: row` & `align-items: center`

Hello there!Hello there!Hello there!Hello there!Hello there!Hello there!

Hello there!Hello there!Hello there!

align (cont'd)

`flex-direction: column` & `align-content: center`

| | |
|--------------|--------------|
| Hello there! | Hello there! |
| Hello there! | Hello there! |
| Hello there! | Hello there! |
| Hello there! | |
| Hello there! | |
| Hello there! | |

align (cont'd)

`flex-direction: column` & `align-items: center`

Hello there!
Hello there!
Hello there!
Hello there!
Hello there!
Hello there!

Hello there!
Hello there!
Hello there!



Homework

→ go read this: <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>