

AxesFoundations Course

Introduction

Let's see how the orientation of items within a flex container can be controlled using the flex-direction property.

Lesson overview

This section contains a general overview of topics that you will learn in this lesson.

- You'll learn about the 2 "axes" of a flex container.
- You'll learn how to change those axes to arrange your content in columns instead of rows.

Axes

vertically, and some rules change a bit depending on which direction you are working with. The default direction for a flex container is horizontal, or row, but you can change the direction to vertical, or column. The direction can be specified in CSS like so:

The most confusing thing about flexbox is that it can work either horizontally or

```
1 .flex-container {
2   flex-direction: column;
3 }
```

No matter which direction you're using, you need to think of your flex-containers as having 2 axes: the main axis and the cross axis. It is the direction of these axes that changes when the flex-direction is changed. In *most* circumstances, flex-direction: row puts the main axis horizontal (left-to-right), and column puts the main axis vertical (top-to-bottom).

In other words, in our very first example, we put <code>display: flex</code> on a div and it arranged its children horizontally. This is a demonstration of <code>flex-direction: row</code>, the default setting. The following example is very similar. If you uncomment the line that says <code>flex-direction: column</code>, those divs will stack vertically.

One thing to note is that in this example, <code>flex-direction: column</code> would not work as expected if we used the shorthand <code>flex: 1</code>. Try it out now (i.e. go change the flex value on the <code>flex: 1 auto; line</code>). Can you figure out why it does not work if <code>flex: 1</code> is used? The divs collapse, even though they <code>clearly</code> have a <code>height</code> defined there.

The reason for this is that the flex shorthand expands <code>flex-basis</code> to <code>0</code>, which means

that all <code>flex-grow</code> ing and <code>flex-shrink</code> ing would begin their calculations from <code>0</code>. Empty divs by default have 0 height, so for our flex items to fill up the height of their container, they don't actually need to have any height at all.

The example above fixed this by specifying <code>flex: 1 1 auto</code>, telling the flex items to

parent .flex-container, or by using flex-grow: 1 instead of the shorthand.

Another detail to notice: when we changed the flex-direction to column, flex-basis refers to height instead of width. Given the context this may be obvious, but it's

default to their given height. We could also have fixed it by putting a height on the

We've strayed from the point slightly... We were talking about flex-direction and axes. To bring it back home, the default behavior is flex-direction: row which arranges things horizontally. The reason this often works well without changing other details in the CSS is because block-level elements default to the full width of their parent. Changing things to vertical using flex-direction: column adds complexity because block-level elements default to the height of their content, and in this case there *is* no content.

save worrying about that until you are ready to start making a website in Arabic or Hebrew."

Knowledge check

"There are situations where the behavior of flex-direction could change if you are

using a language that is written top-to-bottom or right-to-left, but you should

The following questions are an opportunity to reflect on key topics in this lesson. If you can't answer a question, click on it to review the material, but keep in mind you

something to be aware of.

are not expected to memorize or master this knowledge.
 How do you make flex items arrange themselves vertically instead of

- <u>horizontally?</u>
 <u>In a column flex-container, what does flex-basis refer to?</u>
- In a row flex-container, what does flex-basis refer to?
 Why do the previous two questions have different answers?
- Additional resources

This section contains helpful links to related content. It isn't required, so consider it supplemental.

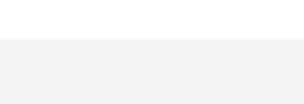
Improve on GitHub

• This <u>flexbox visual cheatsheet</u> has some useful references to flex and its properties.

- For an interactive demo, check out this <u>Scrim on Flexbox axes</u>. Note that this Scrim requires logging into Scrimba in order to view.

Mark Complete

Report an issue

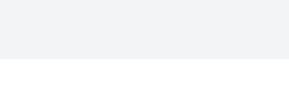


→ Next Lesson

The Odin Project is funded by the community. Join us in empowering learners around the globe by supporting The Odin Project!

Support us!

around the globe by supporting The Odin Project!



Donate now →

See lesson changelog



View Course

Learn more



About usSupportGuidesLegalAboutFAQCommunity guidesTermsTeamContributeInstallation guidesPrivacyBlogContact us

Success Stories

High quality coding education maintained by an open source community.