Flexbox Solves Problems

Why use Flexbox

Flex box enhances our ability as designers and coders to be more responsive. Which means that we are able to write for different break points without needing extra code for media queries.

Flexboxelement image

A Flexbox layout consists of a **flex container** or box that holds **flex items**. The flex container can be laid out horizontally or vertically. This is referred to as the **main axis**.

The direct children of a flex container are laid out along the main axis. These children can “flex” their sizes, growing to fill unused space in the container, or shrinking to avoid overflowing.

By nesting multiple flex containers with different orientations, you can achieve complex layouts.

Flexbox.png

That one line of code does all the following:

* Treats .flex-container as a flex container.
* Treats all direct children of .flex-container as flex items.
* Flex items will be laid out in a horizontal line.
* Flex items will be laid out in source order.
* Flex items will be laid out starting from the left side of the flex container.
* Flex items will be sized based on their regular widthproperties.
* If there’s not enough space for all the flex items, they will be allowed to shrink horizontally until they all fit.
* If they need to shrink, each item will shrink equally.
* Flex items will all stretch vertically to match the height of the tallest flex item.

It is amazing that one line of code can offer the browser so much information.

I would like to show you some Flexbox examples. Navigation is always a tricky part of any design and Flexbox can help. Alignment of elements can become sticky but Flexbox can help. Centering an element vertically was scary before Flexbox but can be done easily now. Change your mind on the order you want elements to follow- no problem you have Flexbox.

Navigation

Navigation.png

Flexbox is a versatile layout module with which we can create one-dimensional layouts that require flexibility, such as responsive menus. Using flexbox’s ordering, alignment, and sizing properties, we can build navigation bars that adapt their layouts to the viewport size while keeping the HTML outline logical and accessible.

How to create a responsive navigation bar with flexbox. Flexbox navigation can have three different layouts, depending on the viewport size:

1. a **mobile layout** in which only the logo and a toggle button will be visible by default and users can open and close the menu using the toggle,
2. a **tablet layout** in which we will show two call-to-action buttons between the logo and toggle in the default state and the rest of the menu will remain toggleable,
3. a **desktop layout** in which all the menu items, except for the toggle button, will be visible on the screen.

Vertical centering

Vertical.png

Flexbox doesn’t use the concepts of “horizontal” and “vertical” centering. It works with main and cross axes which can look like horizontal and vertical centering. The direction of the main and cross axes depends on the value of the flex-direction property.

In row-based flexbox layouts the main axis runs in a horizontal direction, while the cross axis is vertical. In column-based flexbox layouts the main axis runs vertically, while the cross axis is horizontal.

There are two kinds of centering:

1. Centering along the main axis or -block axis.
2. Centering along the cross axis or -inline axis.

If you want an element that is perfectly centered on the screen, you need to use the [justify-content](https://developer.mozilla.org/en-US/docs/Web/CSS/justify-content) property to center the flex items along the main axis. The default value is flex-start which aligns all the items inside the flex container to the beginning of the main axis.

Because, flexbox is a one-dimensional layout, the cross axis has a secondary role. Flex items flow along the main axis. The cross axis doesn’t change the direction of the flow, just adjusts the items on the screen. There are three CSS properties you can use for centering along the cross axis:

1. [align-items](https://developer.mozilla.org/en-US/docs/Web/CSS/align-items) for single-line centering of all the flex items,
2. [align-self](https://developer.mozilla.org/en-US/docs/Web/CSS/align-self) for single-line centering of an individual flex item,
3. [align-content](https://developer.mozilla.org/en-US/docs/Web/CSS/align-content" \t "_self) for multi-line centering of all the flex items (this property only works when flex items wrap into multiple lines).

You can use the [align-items](https://developer.mozilla.org/en-US/docs/Web/CSS/align-items) property to align items along the cross axis. If you use it together with the center property, it centers items vertically when flex-direction is row or row-reverse, and horizontally when flex-direction is columnor column-reverse.

Alignment

Alignment.png

Order

Order.png

In addition to reversing the order in which flex items are visually displayed, you can target individual items and change where they appear in the visual order with the [order](https://developer.mozilla.org/en-US/docs/Web/CSS/order) property.

The order property is designed to lay the items out in *ordinal groups*. What this means is that items are assigned an integer that represents their group. The items are then placed in the visual order according to that integer, lowest values first. If more than one item has the same integer value, then within that group the items are laid out as per source order.

If your items were links or some other element that the user could tab to, then the tabbing order would be the order that these items appear in the document source — not your visual order.

If you are using a reverse value, or otherwise reordering your items, you should consider whether you actually need to change the logical order in the source. The specification continues with a warning not to use reordering to fix issues in your source:

“Authors *must not* use order values of flex-flow/flex-direction as a substitute for correct source ordering, as that can ruin the accessibility of the document.”

Flex items have a default order value of 0, therefore items with an integer value greater than 0 will be displayed after any items that have not been given an explicit order value.

You can also use negative values with order, which can be quite useful. If you want to make one item display first, and leave the order of all other items unchanged, you can give that item an order of -1. As this is lower than 0 the item will always be displayed first.

Use of the order property has exactly the same implications for accessibility as changing the direction with flex-direction. Using order changes the order in which items are painted, and the order in which they appear visually. It does not change the sequential navigation order of the items. Therefore if a user is tabbing between the items, they could find themselves jumping around your layout in a very confusing way.

By tabbing around any of the live examples on this page, you can see how order is potentially creating a strange experience for anyone not using a pointing device of some kind. To read more about this disconnect of visual order and logical order and some of the potential problems it raises for accessibility.

How to build a responsive navigation bar with flexbox

Flex box layout

<https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Ordering_Flex_Items>

flexbox order property

<https://www.ostraining.com/blog/webdesign/css-flexbox-7-the-order-property/>

Flexbox navbar

<https://freshman.tech/flexbox-navbar/>

How to build a responsive navigation bar

<https://coder-coder.com/responsive-navigation-bar-flexbox-vs-css-grid/>

How to build a responsive navigation bar with flexbox

<https://webdesign.tutsplus.com/tutorials/how-to-build-a-responsive-navigation-bar-with-flexbox--cms-33535>

Justify-content

<https://developer.mozilla.org/en-US/docs/Web/CSS/justify-content>

Manning Publications

<https://www.manning.com/books/css-in-depth?query=css#toc>

Ultimate guide to flexbox centering

<https://onextrapixel.com/flexbox-centering-guide/#:~:text=There%20are%20multiple%20use%20cases%20when%20it%20can,with%20just%20a%20few%20lines%20of%20CSS%20code>.