Content

- display: flex
- flex-direction
- justify-content
- align-items
- wrap
- shrink
- expand
- flex-basis
- align-self

CSS Flexbox: Use display: flex to Position Two BoxesPassed

This section uses alternating challenge styles to show how to use CSS to position elements in a flexible way. First, a challenge will explain theory, then a practical challenge using a simple tweet component will apply the flexbox concept.

Placing the CSS property ***display: flex;*** on an element allows you to use other flex properties to build a responsive page.

==========

CSS Flexbox: Use the flex-direction Property to Make a Row

Adding display: flex to an element turns it into a flex container. This makes it possible to align any children of that element into rows or columns. You do this by adding the flex-direction property to the parent item and setting it to ***row*** or ***column***. Creating a row will align the children horizontally, and creating a column will align the children vertically.

Other options for flex-direction are ***row-reverse*** and ***column-reverse***.

Note: The default value for the flex-direction property is row.

==========

CSS Flexbox: Align Elements Using the justify-content Property

Sometimes the flex items within a flex container do not fill all the space in the container. It is common to want to tell CSS how to align and space out the flex items a certain way. Fortunately, the ***justify-content*** property has several options to do this. But first, there is some important terminology to understand before reviewing those options.

Here is a useful image showing a row to illustrate the concepts below.

Recall that setting a flex container as a row places the flex items side-by-side from left-to-right. A flex container set as a column places the flex items in a vertical stack from top-to-bottom. For each, the direction the flex items are arranged is called the main axis. For a row, this is a horizontal line that cuts through each item. And for a column, the main axis is a vertical line through the items.

There are several options for how to space the flex items along the line that is the main axis. One of the most commonly used is justify-content: center;, which aligns all the flex items to the center inside the flex container. Others options include:

flex-start: aligns items to the start of the flex container. For a row, this pushes the items to the left of the container. For a column, this pushes the items to the top of the container. This is the default alignment if no justify-content is specified.

flex-end: aligns items to the end of the flex container. For a row, this pushes the items to the right of the container. For a column, this pushes the items to the bottom of the container.

space-between: aligns items to the center of the main axis, with extra space placed between the items. The first and last items are pushed to the very edge of the flex container. For example, in a row the first item is against the left side of the container, the last item is against the right side of the container, then the remaining space is distributed evenly among the other items.

space-around: similar to space-between but the first and last items are not locked to the edges of the container, the space is distributed around all the items with a half space on either end of the flex container.

space-evenly: Distributes space evenly between the flex items with a full space at either end of the flex container

An example helps show this property in action. Add the CSS property justify-content to the #box-container element, and give it a value of center.

==========

CSS Flexbox: Align Elements Using the align-items Property

The align-items property is similar to justify-content. Recall that the justify-content property aligned flex

items along the main axis. For rows, the main axis is a horizontal line and for columns it is a vertical line.

Flex containers also have a cross axis which is the opposite of the main axis. For rows, the cross axis is vertical and for columns, the cross axis is horizontal.

CSS offers the align-items property to align flex items along the cross axis. For a row, it tells CSS how to push the items in the entire row up or down within the container. And for a column, how to push all the items left or right within the container.

The different values available for ***align-items*** include:

flex-start: aligns items to the start of the flex container. For rows, this aligns items to the top of the container. For columns, this aligns items to the left of the container.

flex-end: aligns items to the end of the flex container. For rows, this aligns items to the bottom of the container. For columns, this aligns items to the right of the container.

center: align items to the center. For rows, this vertically aligns items (equal space above and below the items). For columns, this horizontally aligns them (equal space to the left and right of the items).

stretch: stretch the items to fill the flex container. For example, rows items are stretched to fill the flex container top-to-bottom. This is the default value if no align-items value is specified.

baseline: align items to their baselines. Baseline is a text concept, think of it as the line that the letters sit on.

==========

CSS Flexbox: Use the flex-wrap Property to Wrap a Row or Column

CSS flexbox has a feature to split a flex item into multiple rows (or columns). By default, a flex container will fit all flex items together. For example, a row will all be on one line.

However, using the ***flex-wrap*** property tells CSS to wrap items. This means extra items move into a new row or column. The break point of where the wrapping happens depends on the size of the items and the size of the container.

CSS also has options for the direction of the wrap:

nowrap: this is the default setting, and does not wrap items.

wrap: wraps items from left-to-right if they are in a row, or top-to-bottom if they are in a column.

wrap-reverse: wraps items from right-to-left if they are in a row, or bottom-to-top if they are in a column.

==========

Above Flex properties belongs to CONTAINER and below properties belongs to ITEMS of Flex CONTAINER.

==========

CSS Flexbox: Use the flex-shrink Property to Shrink ItemsPassed

So far, all the properties in the challenges apply to the flex container (the parent of the flex items). However, there are several useful properties for the flex items.

The first is the flex-shrink property. When it's used, it allows an item to shrink if the flex container is too small. Items shrink when the width of the parent container is smaller than the combined widths of all the flex items within it.

The flex-shrink property takes numbers as values. The higher the number, the more it will shrink compared to the other items in the container. For example, if one item has a flex-shrink value of 1 and the other has a flex-shrink value of 3, the one with the value of 3 will shrink three times as much as the other.

==========

CSS Flexbox: Use the flex-grow Property to Expand Items

The opposite of flex-shrink is the flex-grow property. Recall that flex-shrink controls the size of the items when the container shrinks. The flex-grow property controls the size of items when the parent container expands.

Using a similar example from the last challenge, if one item has a flex-grow value of 1 and the other has a flex-grow value of 3, the one with the value of 3 will grow three times as much as the other.

==========

CSS Flexbox: Use the flex-basis Property to Set the Initial Size of an Item

The flex-basis property specifies the initial size of the item before CSS makes adjustments with flex-shrink or flex-grow.

The units used by the flex-basis property are the same as other size properties (px, em, %, etc.). The value auto sizes items based on the content.

==========

CSS Flexbox: Use the order Property to Rearrange Items

The order property is used to tell CSS the order of how flex items appear in the flex container. By default, items will appear in the same order they come in the source HTML. The property takes numbers as values, and negative numbers can be used.

==========

CSS Flexbox: Use the align-self Property

The final property for flex items is align-self. This property allows you to adjust each item's alignment individually, instead of setting them all at once. This is useful since other common adjustment techniques using the CSS properties float, clear, and vertical-align do not work on flex items.

align-self accepts the same values as align-items and will override any value set by the align-items property.