Flex Box

The Flexbox Layout (Flexible Box) module (currently a W3C Candidate Recommandation) aims at providing a more efficient way to lay out, align and distribute space among items in a container, even when their size is unknown and/or dynamic (thus the word “flex”).

The main idea behind the flex layout is to give the container the ability to alter its items' width/height (and order) to best fill the available space (mostly to accomodate to all kind of display devices and screen sizes). A flex container expands items to fill available free space, or shrinks them to prevent overflow.

Most importantly, the flexbox layout is direction-agnostic as opposed to the regular layouts (block which is vertically-based and inline which is horizontally-based). While those work well for pages, they lack flexibility (no pun intended) to support large or complex applications (especially when it comes to orientation changing, resizing, stretching, shrinking, etc.).

Note: Flexbox layout is most appropriate to the components of an application, and small-scale layouts, while the Grid layout is intended for larger scale layouts.

Basics

- Box Model
- Floats
- Clear Fix
- Flex Box
- Homework

Since flexbox is a whole module and not a single property, it involves a lot of things including its whole set of properties. Some of them are meant to be set on the container (parent element, known as “flex container”) whereas the others are meant to be set on the children (said “flex items”).

If regular layout is based on both block and inline flow directions, the flex layout is based on “flex-flow directions”. Please have a look at this figure from the specification, explaining the main idea behind the flex layout.

Basically, items will be layed out following either the main axis (from main-start to main-end) or the cross axis (from cross-start to cross-end).

- **main axis** - The main axis of a flex container is the primary axis along which flex items are laid out. Beware, it is not necessarily horizontal; it depends on the flex-direction property (see below).
- **main-start** | main-end - The flex items are placed within the container starting from main-start and going to main-end.
- **main size** - A flex item’s width or height, whichever is in the main dimension, is the item’s main size. The flex item’s main size property is either the ‘width’ or ‘height’ property, whichever is in the main dimension.
- **cross axis** - The axis perpendicular to the main axis is called the cross axis. Its direction depends on the main axis direction.
- **cross-start** | cross-end - Flex lines are filled with items and placed into the container starting on the cross-start side of the flex container and going toward the cross-end side.
- **cross size** - The width or height of a flex item, whichever is in the cross dimension, is the item’s cross size. The cross size property is whichever of ‘width’ or ‘height’ that is in the cross dimension.

Properties

display: flex | inline-flex; (Applies to: parent flex container element)

This defines a flex container; inline or block depending on the given value. Thus, it enables a flex context for all its direct children.

```
display: other values | flex | inline-flex;
```

Please note that:

- CSS columns have no effect on a flex container
- float, clear and vertical-align have no effect on a flex item
- flex-direction (Applies to: parent flex container element)

This establishes the main-axis, thus defining the direction flex items are placed in the flex container.

```
flex-direction: row | row-reverse | column | column-reverse
```

- row (default): left to right in ltr; right to left in rtl
- row-reverse: right to left in ltr; left to right in rtl
- column: aligns the flex items top to bottom
- column-reverse: aligns the flex items bottom to top
- flex-wrap (Applies to: parent flex container element)

This defines whether the flex container is single-line or multi-line, and the direction of the cross-axis, which determines the direction new lines are stacked in.

flex-wrap: nowrap | wrap | wrap-reverse

```
nowrap (default): single-line / left to right in ltr; right to left in rtl
```

- wrap: multi-line / left to right in ltr; right to left in rtl
- wrap-reverse: multi-line / right to left in ltr; left to right in rtl
- flex-flow (Applies to: parent flex container element)

This is a shorthand `flex-direction` and `flex-wrap` properties, which together define the flex container's main and cross axes. Default is `row nowrap`;

```
flex-flow: <‘flex-direction’> || <‘flex-wrap’>
```

justify-content (Applies to: parent flex container element)

This defines the alignment along the main axis. It helps distribute extra free space leftover when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

justify-content: flex-start | flex-end | center | space-between | space-around

- flex-start (default): items are packed toward the start line
- flex-end: items are packed toward to end line
- center: items are centered along the line
- space-between: items are evenly distributed in the line; first item is on the start line, last item on the end line
- space-around: items are evenly distributed in the line with equal space around them

align-items (Applies to: parent flex container element)

This defines the default behaviour for how flex items are laid out along the cross axis on the current line. Think of it as the justify-content version for the cross-axis (perpendicular to the main-axis).

```
align-items: flex-start | flex-end | center | baseline | stretch
```

- flex-start: cross-start margin edge of the items is placed on the cross-start line
- flex-end: cross-end margin edge of the items is placed on the cross-end line
- center: items are centered in the cross-axis
- baseline: items are aligned such as their baselines align
- stretch (default): stretch to fill the container (still respect min-width/max-width)

align-content (Applies to: parent flex container element)

This aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis.

Note: this property has no effect when the flexbox has only a single line.

```
align-content: flex-start | flex-end | center | space-between | space-around | stretch
```

- flex-start: lines packed to the start of the container
- flex-end: lines packed to the end of the container
- center: lines packed to the center of the container
- space-between: lines evenly distributed; the first line is at the start of the container while the last one is at the end
- space-around: lines evenly distributed with equal space between them
- stretch (default): lines stretch to take up the remaining space

order (Applies to: child element / flex item)

By default, flex items are layed out in the source order. However, the order property controls the order in which they appear in their container.

```
order: <integer>
```

flex-grow (Applies to: child element / flex item)

This defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.

If all items have flex-grow set to 1, every child will set to an equal size inside the container. If you were to give one of the children a value of 2, that child would take up twice as much space as the others.

```
flex-grow: <number> (default 0)
```

Negative numbers are invalid.

flex-shrink (Applies to: child element / flex item)

This defines the ability for a flex item to shrink if necessary.

```
flex-shrink: <number> (default 1)
```

Negative numbers are invalid.

flex-basis (Applies to: child element / flex item)

This defines the default size of an element before the remaining space is distributed.

```
flex-basis: <length> | auto (default auto)
```

flex (Applies to: child element / flex item)

This is the shorthand for flex-grow, flex-shrink and flex-basis. The second and third parameters (flex-shrink, flex-basis) are optional. Default is 0 1 auto.

```
flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
```

align-self (Applies to: child element / flex item)

This allows the default alignment or the one specified by align-items to be overridden for individual flex items.

Please see the align-items explanation to understand the available values.

```
align-self: auto | flex-start | flex-end | center | baseline | stretch
```

Source

all content is from http://css-tricks.com/snippets/css/a-guide-to-flexbox/