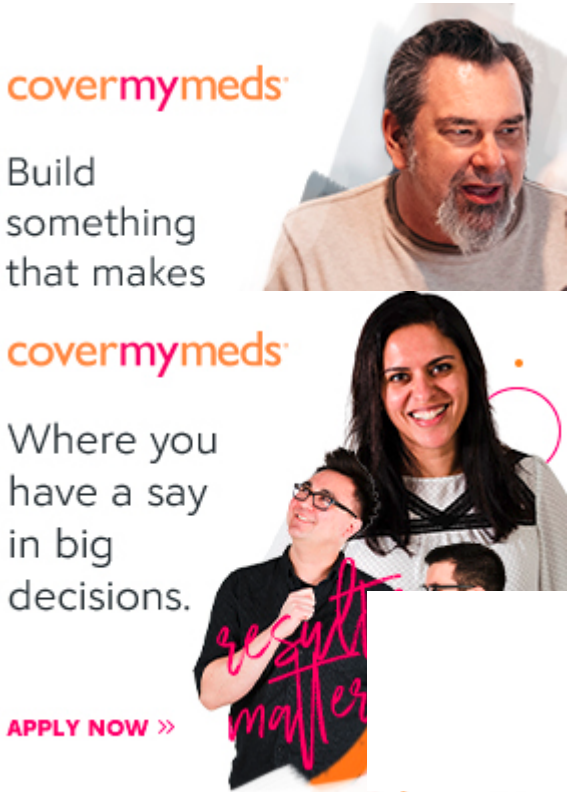
www.akajlm.net

#vue #angular #javascript #node #laravel #css #vs-code

The **Flexbox Layout** officially called CSS Flexible Box Layout Module (https://www.w3.org/TR/css-flexbox/) is new layout module in CSS3 made to improve the items align, directions and order in the container even when they are with dynamic or even unknown size. The prime characteristic of the flex container is the ability to modify the width or height of its children to fill the available space in the best possible way on different screen sizes.



Many designers and developers find this

flexbox layout easier to use, as positioning of the elements is simpler thus more complex layouts can be achieved with less code, leading to simpler development process. Flexbox layout algorithm is direction based unlike the block or inline layout which are vertically and horizontally based. This flexbox layout should be used for small application components, while new CSS Grid Layout Module (https://www.w3.org/TR/css-grid/) is emerging to handle the large scale layouts.

Rather that explaining how the flex properties work, this guide will focus on how the flex properties affect the layout in a visual way.

Table of Contents

Basics

Before we start with describing the flexbox properties, let's give a little introduction of the flexbox model. The flex layout is constituted of parent container referred as *flex container* and its immediate children which are called *flex items*.

CSS3-Flexbox-Model

In the box above you can see the properties and the terminology used to

describe the flex container and its children. For more information on their meaning read the official flexbox model (https://www.w3.org/TR/css-flexbox/#box-model) by W3C.

The flexbox layout went through many iterations and several syntax changes from its initial draft in 2009, so to avoid confusion and make everything clear we'll use only the syntax from the last working draft (Sep 2014). If you need to maintain old browser compatibility you can read this article (https://css-tricks.com/using-flexbox/) on how to do that in the best way.

The browser support for the latest flexbox specification is:

- » Chrome 29+
- » Firefox 28+
- » Internet Explorer 11+
- » Opera 17+
- » Safari 6.1+ (prefixed with `-webkit-`)
- » Android 4.4+
- » iOS 7.1+ (prefixed with `-webkit-`)

You can see detailed browser support and compatibility here (https://caniuse.com/flexbox) .

from Scratch (https://beginnertailwind.com)

Usage

To use flexbox layout just set the `display` property on the parent HTML element:

```css
.flex-containe
  display: -we
  display: fle
}
```

Or if you want to display it like an inline element use:

```css
.flex-containe
  display: -we
  display: ini
}
```

Note: This is the only property you need to set on the parent container and all its immediate children will become automatically flex items.

There are several ways to group the flexbox properties and by far the easiest way I've found to understand the flexbox options and their usage is to divide them in two groups one for the flex

container and one for the flex items. Below are explained all of them and how they affect the layout visually.

Flexbox container properties

flex-direction

This property specifies how flex items are laid out in the flex container, by setting the direction of the flex container's main axis. They can be laid out in

two main directions, like rows horizontally or like columns vertically.

Values:

```css
.flex-containe
  -webkit-flex
  flex-direct
}
```

With `row` direction the flex items are stacked in a row from left-to-right in `ltr` context

```css
.flex-containe
```

```
    -webkit-flex
    flex-direct
}
```

With `row-reverse` *direction the flex items are stacked in a row from right-to-left in* `ltr` *context*

```
        css

.flex-containe
    -webkit-flex
    flex-direct
}
```

With `column` *direction the flex items are stacked in a column from top-to-bottom*

```css
.flex-containe
  -webkit-flex
  flex-direct
}
```

With `column-reverse` *direction the flex items are stacked in a column from bottom-to-top*

Default value: `row`

Note: `row` and `row-reverse` are dependent of the writing mode so in `rtl` context they will be reversed respectively.

flex-wrap

The initial flexbox concept is the container to set its items in one single line. The `flex-wrap` property controls if the flex container lay out its items in single or multiple lines, and the direction the new lines are stacked in.

Values:

```bash
.flex-containe
  -webkit-flex
  flex-wrap:
}
```

Flex items are displayed in

one row, by default they are shrunk to fit the flex container's width

```
css

.flex-containe
  -webkit-flex
  flex-wrap:
}
```

Flex items are displayed in multiple rows if needed from left-to-right and top-to-bottom

```
css

.flex-containe
  -webkit-flex
  flex-wrap:
```

```
}
```

Flex items are displayed in multiple rows if needed from left-to-right and bottom-to-top

Default value: `nowrap`

Note: These properties are dependent of the writing mode so in `rtl` context they will be reversed respectively.

flex-flow

This property is a shorthand for setting the `flex-direction` and

`flex-wrap` properties.

Values:

```
css

.flex-containe
  -webkit-flex
  flex-flow:
}
```

Default value:

`row nowrap`

justify-content

The `justify-content` property aligns flex items

along the main axis of the current line of the flex container. It helps distribute left free space when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size.

Values:

```
css

.flex-containe
  -webkit-just
  justify-cont
}
```

Flex items are aligned to the left side of the flex container in `ltr` *context*

```css
.flex-containe
  -webkit-just
  justify-cont
}
```

Flex items are aligned to the right side of the flex container in `ltr` *context*

```css
.flex-containe
  -webkit-just
  justify-cont
}
```

Flex items are aligned at

the center of the flex container

```
css

.flex-containe
  -webkit-just
  justify-cont
}
```

Flex items are displayed with equal spacing between them, first and last flex items are aligned to the edges of the flex container

```
css

.flex-containe
  -webkit-just
```

```
    justify-cont
}
```

Flex items are displayed with equal spacing around every flex item, even the first and last flex items

Default value:

`flex-start`

align-items

Flex items can be aligned in the cross axis of the current line of the flex container, similar to `justify-content` but in the perpendicular direction. This property sets the default alignment

for all flex items, including the anonymous ones.

Values:

```css
.flex-containe
  -webkit-alig
  align-items.
}
```

*Flex items fill the whole height _(*or width_)_ from ***cross start*** to ***cross end*** of the flex container_

```css
.flex-containe
```

```
  -webkit-alig
  align-items:
}
```

*Flex items are stacked to the _**cross start*** of the flex container_

```
        css

.flex-containe
  -webkit-alig
  align-items:
}
```

*Flex items are stacked to the _**cross end*** of the flex container_

```css
.flex-containe
  -webkit-alig
  align-items.
}
```

*Flex items are stacked to the center of the _**cross axis** of the flex container_*

```css
.flex-containe
  -webkit-alig
  align-items.
}
```

Flex items are aligned in a

way that their baselines are aligned

Default value: `stretch`

Note: Read more details about how baselines are calculated here (https://www.w3.org/TR/css-flexbox/#flex-baselines) .

align-content

The `align-content` property aligns a flex container's lines within the flex container when there is extra space in the cross-axis, similar to how `justify-content` aligns

individual items within the main-axis.

Values:

```
css

.flex-containe
  -webkit-alig
  align-conten
}
```

Flex items are displayed with distributed space after every row of flex items

```
css

.flex-containe
  -webkit-alig
```

```
  align-conte
}
```

*Flex items are stacked toward the _**cross start*** of the flex container_

```
        css

.flex-containe
  -webkit-alig
  align-conte
}
```

*Flex items are stacked toward the _**cross end*** of the flex container_

```css
.flex-containe
  -webkit-alig
  align-conter
}
```

Rows of flex items are stacked in the center of the ***_cross axis*** of the flex container_

```css
.flex-containe
  -webkit-alig
  align-conter
}
```

Rows of flex items are displayed with equal spacing between them, first and last rows are aligned to the edges of the flex container

```css
.flex-containe
  -webkit-alig
  align-conter
}
```

Flex items are displayed with equal spacing around every row of flex items.

Default value: `stretch`

Note: This property has only effect when the flex

container has multiple lines of flex items. If they are placed in single line this property has no effect on the layout.

Note for flex containers

- » all of the `column-*` properties have no effect on a flex container.
- » the `::first-line` and `::first-letter` pseudo-elements do not apply to flex containers.

Flexbox item properties

order

The order property controls the order in which children of a flex container appear inside the flex container. By default they are ordered as initially added in the flex container.

Values:

```
css

.flex-item {
```

```
    -webkit-orde
    order:
}
```

Flex items can be reordered with this simple property, without restructuring the HTML code

Default value: 0

flex-grow

This property specifies the flex grow factor, which determines how much the flex item will grow relative to the rest of the flex items in the flex container when positive

free space is distributed.

Values:

```css
.flex-item {
  -webkit-flex
  flex-grow:
}
```

If all flex items have same value for `flex-grow` *than all items have same size in the container The second flex item takes up more space relative to the size of the other flex items*

Default value: 0

Note: Negative numbers are invalid.

flex-shrink

The `flex-shrink` specifies the flex shrink factor, which determines how much the flex item will shrink relative to the rest of the flex items in the flex container when negative free space is distributed.

Values:

```css
.flex-item {
  -webkit-flex
  flex-shrink:
}
```

*By default all flex items can be shrunk, but if we set it to 0 _(*don't shrink_)_ they will maintain the original size_

Default value: 1

Note: Negative numbers are invalid.

flex-basis

This property takes the same values as the `width` and `height` properties, and specifies the initial main size of the flex item, before free space is distributed according to the flex factors.

Values:

```css
.flex-item {
  -webkit-flex
  flex-basis:
}
```

`flex-basis` ___ is specified for the 4th flex item and dictates the initial size of the element_

Default value: `auto`

Note: There is a naming issue (https://www.w3.org/TR/css-flexbox/#valdef-flex-basis-main-size) with the `auto` value which will be resolved in future.

flex

This property is the shorthand for the `flex-grow`, `flex-shrink` and `flex-basis` properties. Among other values it also can be set to `auto` (`1 1 auto`) and `none` (`0 0 auto`).

```css
.flex-item {
  -webkit-flex
  flex:
}
```

Default value: `0 1 auto`

Note: W3C encourages to use the flex shorthand rather than the separate component properties, as the shorthand correctly resets any unspecified components to accommodate common uses (https://www.w3.org/TR/css-flexbox/#flex-common).

align-self

This `align-self` property allows the default alignment (or the one specified by `align-items`) to be overridden for individual flex items.

Refer to `align-items` explanation for flex container to understand the available values.

Values:

```
css

.flex-item {
  -webkit-alig
  align-self:
}
```

The 3rd and 4th flex items have overridden alignment through the `align-self` *property*

Default value: `auto`

Note: The value of `auto` for `align-self` computes to the value of `align-items` on the element's parent, or `stretch` if the element has no parent.

Note for flex items

- » `float` , `clear` and `vertical-align` have no effect on a flex item, and do not take

it out-of-flow.

Flexbox playground

Here's a flex playground where you can play with the different flex properties and explore the power of the flexbox layout. Combine several flex properties to get complex layouts.

See the Pen
Flexbox Properties Demonstration
(https://codepen.io/justd/pen/yydezN/)
by Dimitar (