FLEXBOX NOTES

19 December 2022 14:35

FLEXBOX ->

Flexbox is a way to arrange items into rows or columns.

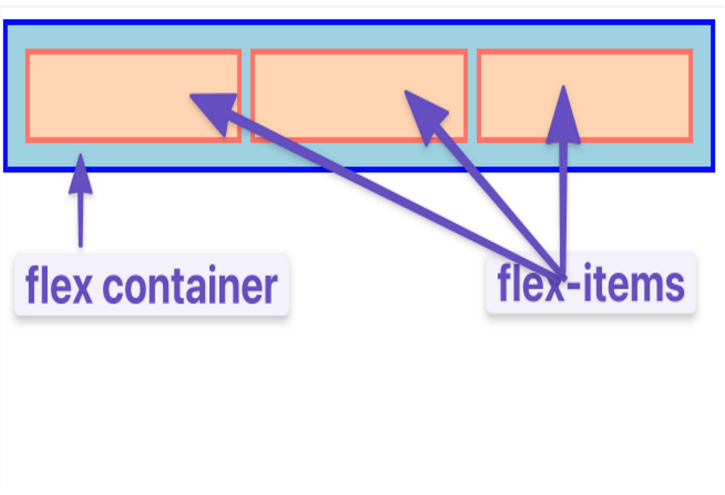
1.FLEX CONTAINER AND FLEX ITEMS - >

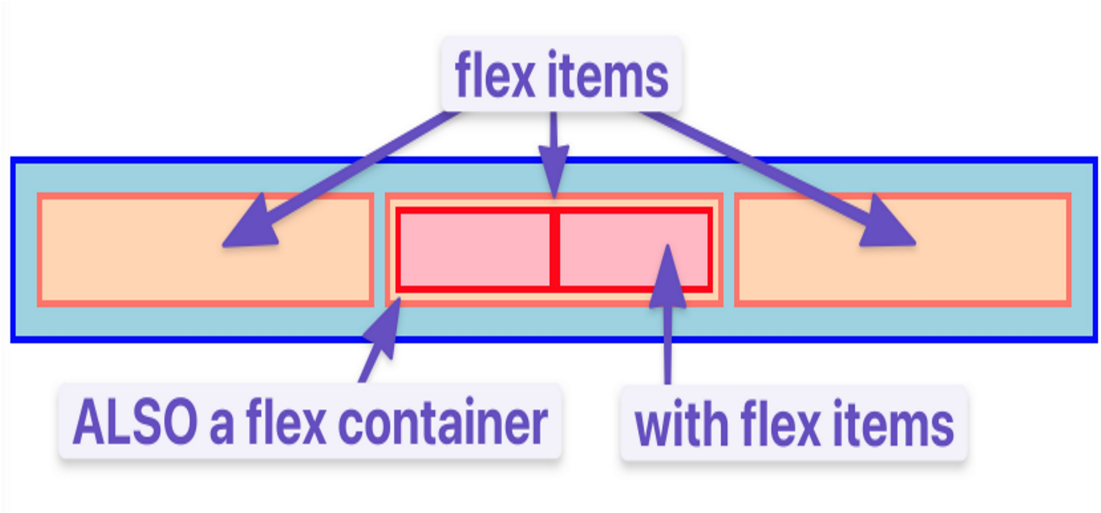
-> flexbox is not just a single CSS property but a whole toolbox o properties that you can use to put things where you need them. Some of these properties belong on the *flex container*, while some go on the *flex items*. This is a simple yet important concept

-> A flex container is any element that has **display: flex** on it. A flex item is any element that lives directly inside of a flex container.

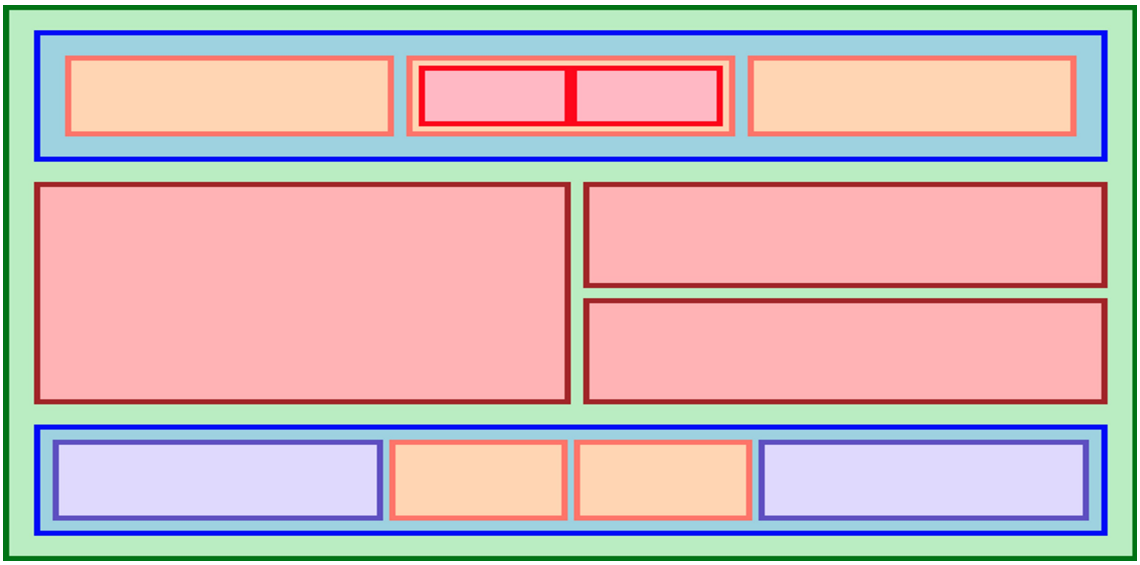
```
.container {
  background: lightblue;
  padding: 16px;
  border: 4px solid blue;
  gap: 8px;
  display: flex;
}

.item {
  background: peachpuff;
  border: 4px solid salmon;
  height: 48px;
  flex: 1;
  padding: 4px;
}
```





-> The following image was achieved using *only* flexbox to arrange, size, and place the various elements. Flexbox is a *very* powerful tool.



2.Growing And Shrinking ->

The flex shorthan :

-> The flex declaration is actually a shorthand for 3 properties that you can set on a flex item
-> These properties affect how flex items size themselves within their container

-> Shorthand properties are CSS properties that let you set the values of multiple other CSS properties simultaneously. Using a shorthand property, you can write more concise (and often more readable) stylesheets, saving time and energy.

-> In this case, flex is actually a shorthand for flex-grow, flex-shrink and flex-basis.

```
div {
  flex: 1;
}
```

-> In the above screenshot, flex: 1 equates to: flex-grow: 1, flex-shrink: 1, flex-basis: 0.

A. Flex-Grow :

-> flex-grow expects a single number as its value, and that number is used as the flex-item's "growth factor"

B. Flex-Shrink :

-> flex-shrink is similar to flex-grow, but sets the "shrink factor" of a flex item. flex-shrink only ends up being applied if the size of all flex items is larger than their parent container

C .Flex-Basis :

-> flex-basis simply sets the initial size of a flex item, so any sort of flex-growing or flex-shrinking starts from that baseline size. The shorthand value defaults to flex-basis: 0%

Important Note About Flex-Basis:
There is a difference between the default value of flex-basis and the way the flex shorthand defines it if no flex-basis is given. The actual default value for flex-basis is auto, but when you specify flex: 1 on an element, it interprets that as flex: 1 1 0. If you want to only adjust an item's flex-grow you can simply do so directly, without the shorthand. Or you can be more verbose and use the full 3 value shorthand flex: 1 1 auto, which is also

equivalent to using flex: auto.

Flex initial : flex: 0 1 auto;

Flex auto : flex : 1 1 auto;

Flex none : flex 0 0 auto;

3.Axes

-> flex-containers as having 2 axes: the main axis and the cross axis

-> flex-direction: row puts the main axis horizontal (left-to-right), and column puts the main axis vertical (top-to-bottom)

4. Alignment

->justify-content: space-between;
->justify-content: flex-start;
->justify-content: flex-end;
->align-item: center;
->align-item: flex-start;
->align-self: flex-start;
->justify-content: space-around;
->justify-content: space-evenly;

5. Gap

->gap : 8px;

/////Important notes about flexbox://///

-> Flexbox is all about *flexibility*. We can control whether items grow or shrink, how the extra space is distributed, and more.

Flexbox direction

-> we can flip to a column with the flex-direction property:
Show Primary Axis

->With flex-direction: row, the primary axis runs horizontally, from left to right. When we flip to flex-direction: column, the primary axis runs vertically, from top to bottom.

1. **Primary axis:** Children will be bunched up at the start of the container.
2. **Cross axis:** Children will stretch out to fill the entire container.

Alignment

->We can change how children are distributed along the primary axis using the **justify-content** property

->flex-direction also matter

->flex-start, center, and flex-end), or we can spread them apart (with space-between, space-around, and space-evenly).

->align-item: flex-start,stretch,baseline,flex-end,center;
->align-self:
->justify-self:

->

- **justify** — to position something along the primary axis.

  .container { justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right ... + safe | unsafe; }

- **align** — to position something along the cross axis

  .container { align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start | self-end + ... safe | unsafe; }

- **content** — a group of “stuff” that can be distributed.

  .container { align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline + ... safe | unsafe; }

- **items** — single items that can be positioned individually.

Growing and shrinking

->about 3 properties: flex-grow, flex-shrink, and flex-basis.

//flex-basis//

->**To put it simply:** In a Flex row, flex-basis does the same thing as width. In a Flex column, flex-basis does the same thing as height.

//flex-grow//

->By default, elements in a Flex context will shrink down to their minimum comfortable size along the primary axis. This often creates extra space.

//flex-shrink//

->
I had an epiphany a while back about flex-shrink: we can think of it as the “inverse” of flex-grow. They're two sides of the same coin:

-
-
- flex-grow controls how the extra space is distributed when the items are smaller than their container.
- flex-shrink controls how space is removed when the items are bigger than their container.

**gap **

-> gap: 73px;

wrapping

-> flex-wrap
->.container { flex-wrap: nowrap | wrap | wrap-reverse; }
->

- nowrap (default): all flex items will be on one line
- wrap: flex items will wrap onto multiple lines, from top to bottom.
- wrap-reverse: flex items will wrap onto multiple lines from bottom to top.

Align-items
Ans align-content ----use in the flebox

To summarize what's happening here:

- flex-wrap: wrap gives us two rows of stuff.
- Within each row, align-items lets us slide each individual child up or down
- Zooming out, however, we have these two rows within a single Flex context! The cross axis will now intersect two rows, not one. And so, we can't move the rows individually, we need to distribute them *as a group*.
- Using our definitions from above, we're dealing with content, not items. But we're also still talking about the cross axis! And so the property we want is align-content