Frontendβ → CSS → Flexbox → Axis alignment

Theory: Axis alignment

🕓 14 minutes 0 / 5 problems solved

Skip this topic Start practicing

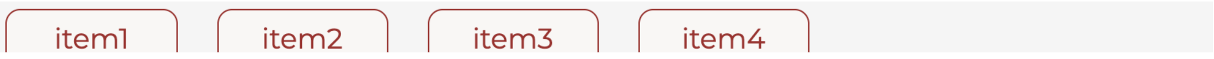
22 users solved this topic. Latest completion was about 18 hours ago.

Flexbox allows adjusting items to fill the container, which can be accomplished not only with `flex-grow` and `flex-shrink` but also with the properties we'll touch on in this topic. Those are more complex, have more advanced settings than already mentioned `flex-grow` and `flex-shrink`, and, honestly, are much more comfortable to work with. These are the reasons why they are oftenly used in real life.

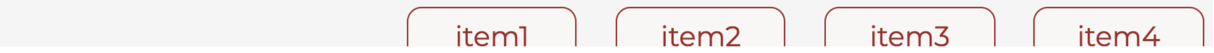
§1. Justify-content

As we already know, if neither `flex-grow` and `flex-shrink` properties nor the `margin` property are applied or by default the flex items are grouped together at the **main-start**. We use `flex-grow` and `flex-shrink` to distribute the remaining space between the items making them grow or shrink. `Justify-content` property allows to distribute the remaining space between items by setting different gaps and margins instead of changing items' size. This property has only several values:

- `justify-content: flex-start;` — the default value: items are stuck to the **main-start** and grouped together



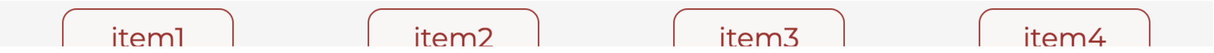
- `justify-content: flex-end;` — items will be grouped together and stuck to the main-end **in the right order**. Please, note, that using this value is not similar to using `flex-direction: row-reverse;`



- `justify-content: center;` — items will be grouped together and aligned to the center of the flex container.

- `justify-content: space-between;` — items will be stretched to full width. The first item will be stuck to the main-start and the last one — to the **main-end**.

- `justify-content: space-around;` — similar to the previous one, but with this value there will be a gap between the main-start and the first element as well as between the main-end and the last element. The gap will be half the size of the gap between the two closest items.



§2. Align-items and align-self

While `justify-content` exists to align items along the main axis, `align-items` aligns them along the cross axis. The concept is the same, but the values are different:

- `align-items: flex-start;` — sticks the items to the **cross-start**.

item1
item2
JET
item4

Current topic:

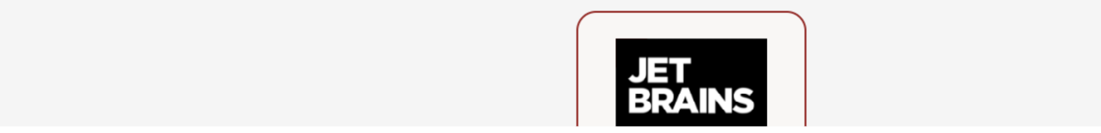
Axis alignment Stage 2

Topic depends on:

✕ Flexibility, growth and contraction ratio Stage 2

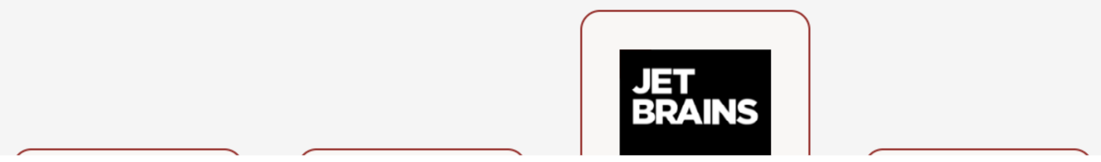
Table of contents:

- `align-items: flex-end;` — sticks the items to the cross-end.



- `align-items: center;` — places the items to the center of the cross-axis.

- `align-items: baseline;` — aligns the items with their baselines. The baseline is an abstract line right above the text.



- `align-items: stretch;` — items occupy the same size as the cross axis.

§3. Align-self

While `align-items` declares the behavior of all the items, `align-self` defines the alignment of the single item it's applied to. The values of this property are the same except for one external value:

- `align-self: auto;` makes the item inherit `align-items` value from the parent or makes it `stretch` if the parent doesn't have one.

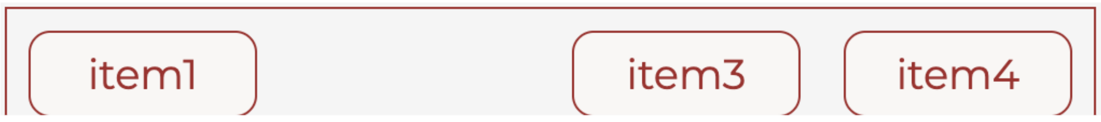
Notice that even if `align-items` is applied to the flex container, `align-self` applied to the element will be counted anyway:

```
<div class="flex-container">
  <div class="flex-item item1">item1</div>
  <div class="flex-item item2">item2</div>
  <div class="flex-item item3">item3</div>
  <div class="flex-item item4">item4</div>
</div>
```

```
.flex-container{
  color: brown;
  display: flex;
  font-family: 'Montserrat', sans-serif;
  padding: .5em 0 .5em 0;
  width: 40em;
  align-items: flex-start;
}

.flex-item{
  border: 1px solid brown;
  border-radius: 10px;
  text-align: center;
  height:2em;
  line-height: 2em;
  margin-right: 0.5em;
  margin-left: 0.5em;
  flex-basis: 6em;
  flex-grow: initial;
}

.item2{
  align-self: flex-end;
}
```

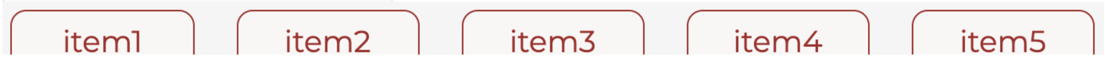


§4. Align-content

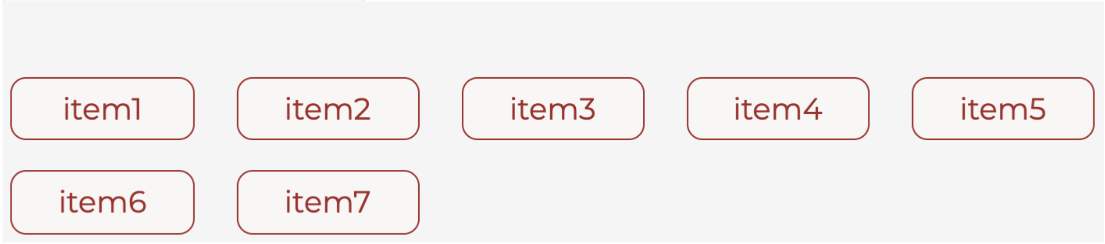
This property defines the distribution of the free space between the lines located along the main axis if there is enough free space on the cross axis. Okay, once again: if there are several lines in one flex container and there are some available pixels between them - `align-content` is to distribute those pixels between those lines. It works similar to `justify-content`, but aligns lines, not items.

Possible values are completely the same as for `justify-content`, but let's look through them again anyway:

- `align-content: flex-start;`



- `align-content: flex-end;`



- `align-content: center;`

- `align-content: space-between;`

- `align-content: space-around;`

§5. Summary

This is the last topic about flexbox module. This time we've leaned how to operate with free space inside of the flex container in any possible case: `justify-content` is for the in-line free space, `align-self` and `align-items` are for the free space on the cross axis and `align-content` is for the free space in case there are several lines.

As it was said earlier in the introduction, flexbox is a really helpful module. With only two types of blocks and properties which you can count on your fingers and which most of the times are easy to use, you can create a perfect layout however you see it.

Report a typo

How did you like the **theory**?

😍 🙂 😐 🙁 😡

Start practicing

This content was created 2 months ago and updated 10 days ago. Share your feedback below in comments to help us improve it!

Comments (0) Hints (0) Useful links (0) Show discussion