Theory: Orientation and display order

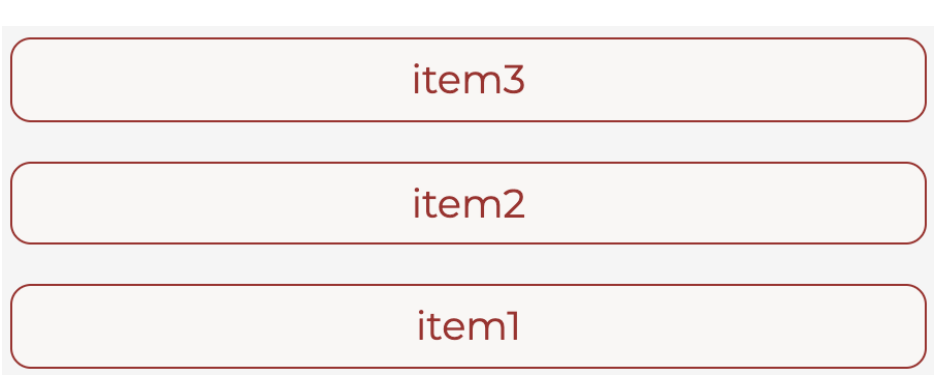
24 minutes 0 / 5 problems solved

As you already know, it's possible to place flex items in any direction along any axis. When is it necessary? Literally always, both for the whole page layout or for the element markup. You may find another solution for this problem, but it most likely will be setting a `width` for all the elements in CSS. Flexbox properties mentioned in this topic make the whole process much easier and more flexible. That's where the name 'flexbox' comes from.

§1. Flex-direction

The `flex-direction` property belongs to the flex-container and is used to determine the direction in which flex items are placed. By default, it has a **row value**. That means all the elements are lining up in a row along the main axis from left to right. Here are other possible values:

- `row` is a default value.
- `row-reverse` value makes flex items line up in a row from right to left along the main axis.
- `column` value makes flex items line up in a column from top to bottom along the cross axis.
- `column-reverse` value makes flex items line up in a column from bottom to top along the cross axis.



And here's the syntax example:

```
.flex-container{
  flex-direction: column;
}
```

§2. Flex-wrap

If there are too many flex items inside one container, extra elements will first shrink to their minimal size (i.e. the size of the content) and then just go out of the borders of the container. Here are the 3 items which barely fit the container:

Current topic:
Orientation and display order Stage 2

Topic depends on:
✕ Introduction to Flexbox Stage 3

Topic is required for:
Flexibility, growth and contraction ratio Stage 2

Table of contents:
↑ Orientation and display order
§1. Flex-direction
§2. Flex-wrap
§3. Flex-flow
§4. Order
§5. Summary
Feedback & Comments

If we add one more item, their width will decrease a bit to fit the container even though it's determined in CSS:

Now, if we add a couple more items, their width will decrease to the size of their content. If there are still too many elements, they will overflow:

`Flex-wrap` is a property that allows to control overflow and make extra items either follow the rules described above or move to the next row (or column, if `flex-direction: column;` applied). It also belongs to the flex container.

`Flex-wrap: nowrap;` is a default value that doesn't let items move to the next row and makes them stay in place even in case of overflow. Below you can see the three remaining properties:

- `nowrap` is a default value.
- `wrap` value makes overflow items move to the next line.

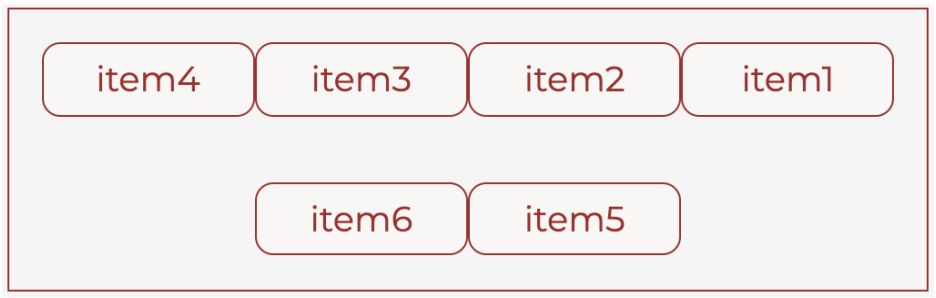
- `wrap-reverse` value makes overflow items move to the next line and places them in the opposite order.

§3. Flex-flow

`Flex-flow` is a shorthand property combining `flex-direction` and `flex-wrap`. According to recommendations from Google on CSS code style, using shorthand properties is a good practice. That's why you should consider using `flex-flow`.

For example, the following HTML code will look like the picture below.

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



However, only if any of the code snippets below are applied.

```
.flex-container{
  flex-direction: row-reverse;
  flex-wrap: wrap;
}
```

```
.flex-container{
  flex-flow: row-reverse wrap;
}
```

§4. Order

`Order property` belongs to flex items. Its value is a digit, its default value is `order: 0;`, and it determines the display order of the items. The lower the value is, the earlier the item is displayed. In the example below all the items have the default order value except for the last one which has `order:-1;` property.

```
<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>
```

```
.item3{
  order: -1;
}

.flex-container{
  border: 1px solid brown;
  color: brown;
  display: flex;
  font-family: 'Montserrat', sans-serif;

  padding: 0.5em 0 0.5em 0;

  width: 30em;

  justify-content: center;

}


.flex-item{

  border: 1px solid brown;

  border-radius: 10px;

  text-align: center;

  height:2em;

  line-height: 2em;

  margin: 0 .5em 0 .5em;

  flex-basis: 4em;

}
```

The result of this code will be the following:

Now let's randomly change the order. The result of this code will look like the picture below.

```
<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>
```

```
.item1{
  order: 2;
}

.item2{
  order: 3;
}

.item3{

  order: 1;

}



.flex-container{

  border: 1px solid brown;

  color: brown;

  display: flex;

  font-family: 'Montserrat', sans-serif;

  padding: 0.5em 0 0.5em 0;

  width: 30em;

  justify-content: center;

}



.flex-item{

  border: 1px solid brown;

  border-radius: 10px;

  text-align: center;

  height:2em;

  line-height: 2em;

  margin: 0 .5em 0 .5em;

  flex-basis: 4em;

}
```

In the example above we used the number of the items as a max value. However, it's not necessary to have an order: N value for the item which has to be displayed Nth. The value just has to be higher than the value of the item which should be displayed before this one and lower than the value of the item which has to be displayed next.

If several items have the same order property value, they will be displayed in the same order as they are declared in HTML.

Here's an example:

```
<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>
```

```
.item1{
  order: 11;
}

.item2{
  order: 30;
}

.item3{

  order: 25;

}



.flex-container{

  border: 1px solid brown;

  color: brown;

  display: flex;

  font-family: 'Montserrat', sans-serif;

  padding: 0.5em 0 0.5em 0;

  width: 30em;

  justify-content: center;

}



.flex-item{

  border: 1px solid brown;

  border-radius: 10px;

  text-align: center;

  height:2em;

  line-height: 2em;

  margin: 0 .5em 0 .5em;

  flex-basis: 4em;

}

```

Below you can see the result:

§5. Summary

Flexbox is controlled through flex container and flex items properties. `Flex-direction` allows to manipulate the axes and directions, `flex-wrap` helps to handle overflow elements, and the `order` property lets us manipulate the display order of the items. Also, there is a shorthand property combining flex-direction and flex-wrap. Now let's get down to practice!

Report a typo

8 users liked this theory. 0 didn't like it. What about you?

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