CSS FLEXBOX

CSS Flexbox

Before flexbox, website layout is generally done by using float property. But when using float there are some issues like when using float, the items which are floated does not considered in regular flow and issue arrive like background and border shrinking. Hence, we need to clear the float to overcome such types of issue.

Flexbox provided an easy way to layout things on webpage. Using flexbox, we can easily arrange items contained in flexbox container.

Flexbox is a set of CSS properties, which are used together to creates a layout.

Below are the various properties of flexbox with an explanation and examples.

Flexbox layout has two main things.

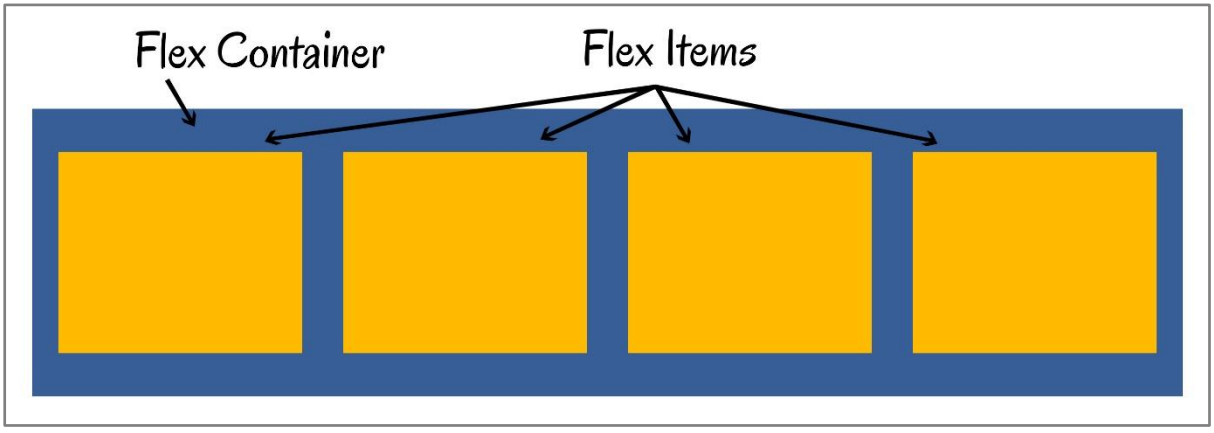
1. Flex container – which contains all the things which need to be laid out.
2. Flex items – these are the individual items of flexbox which need to be laid out.

Properties of Flex Container

Below are the various properties which can be applied to flex container.

Display: flex

To define a flexbox container, we need to set its containers display property to flex. So, when to make a flexbox container, all its direct children will become flex items.



CSS FLEXBOX

Example

```
<div class="container">
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
    <div class="item">4</div>
    <div class="item">5</div>
  </div>
```

```
.container {
  display: flex;
}
```

In above example when we set display property of div with class container to flex, it will become flex container and all its direct child (div with class items) will automatically become flex items.

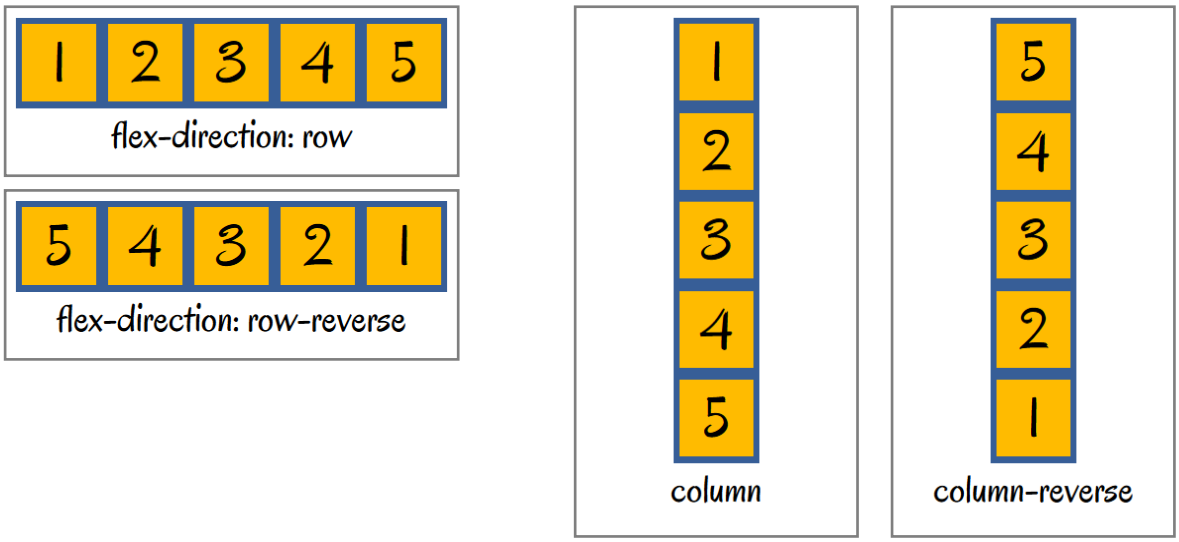
Flex-direction

Flexbox is single direction layout, means all flex items are laid out in single dimension either horizontal or vertical.

Flex-direction property is used to set the main axis of flex items. It can have below values.

- Row – Flex items lay out in horizontal axis from left to right.
- Row-reverse - Flex items lay out in horizontal axis from right to left.
- Column – Flex items lay out in vertical axis from top to bottom.
- Column-reverse – Flex items lay out in vertical axis from bottom to top.

CSS FLEXBOX

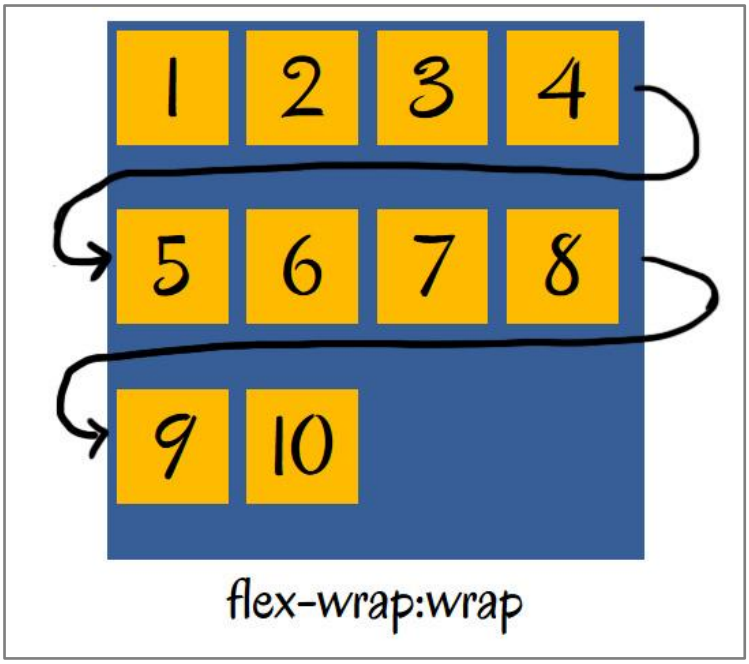


Flex-wrap

By default, all flex item will try to fit into one row/column depend on which property value is selected for flex-direction. And if flex items do not fit into container it will spill out of container.

To prevent this, we can use flex-wrap property. Flex-wrap can have below three values.

- Nowrap – It is default value, and means that flex items do not wrap even if it does not fit into container.
- Wrap – If flex items do not fit into container it will wrap into multiple lines.
- Wrap-reverse – It is like a wrap, but it will wrap flex item in reverse order.



CSS FLEXBOX

Flex-flow

It is a shorthand property to set flex-direction and flex-wrap.

Syntax

```
Flex-flow: <flex-direction value> <flex-wrap> value;
```

Example

```
Flex-flow: column wrap;
```

Justify-content

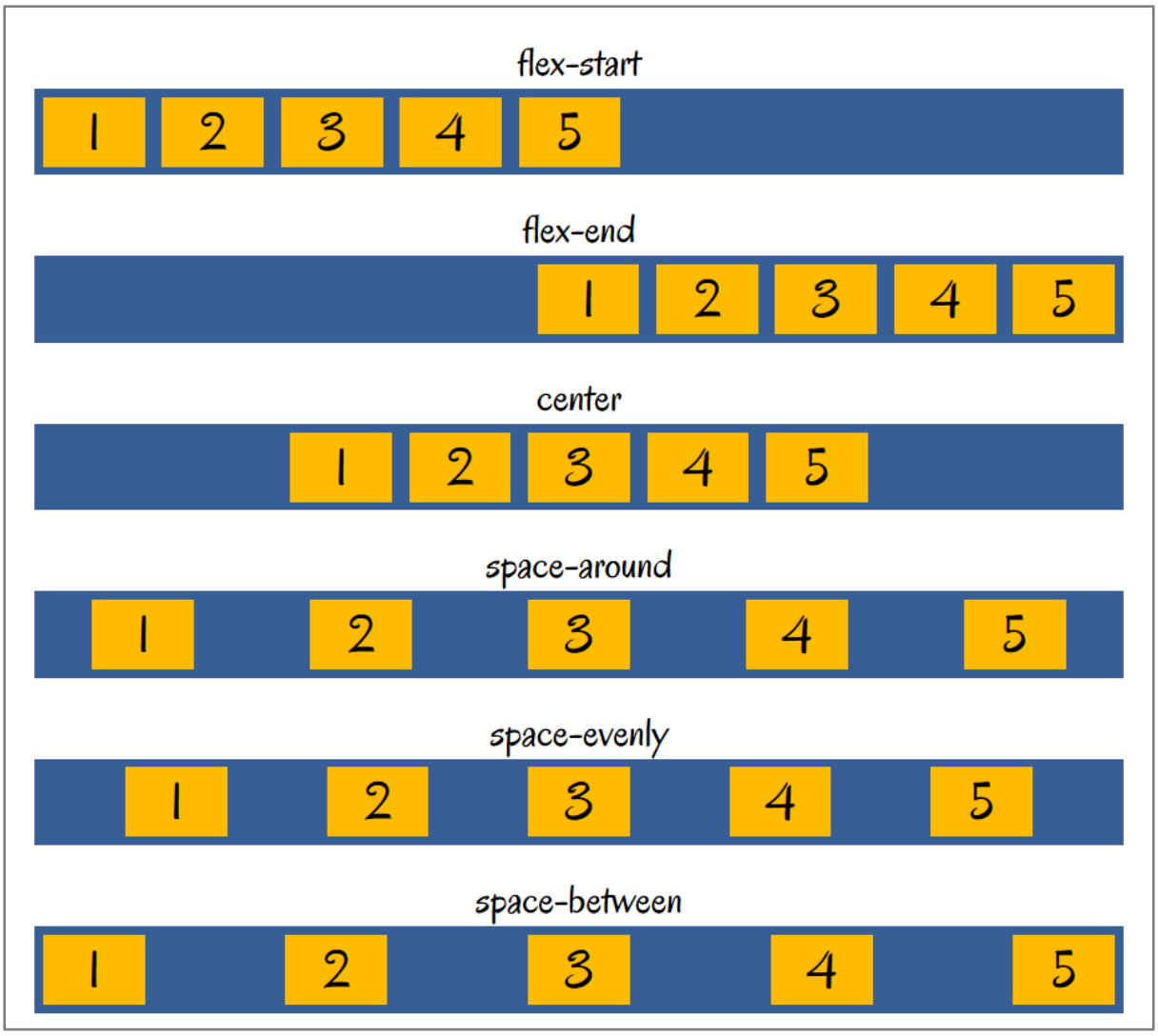
Justify-content property is used to set the alignment of flex items along with main axis. Means if flex-direction property is set to row/row-reverse, it will align item on horizontal axis and if flex-direction property is set to column/column-reverse it will align item on vertical axis.

It is used to set the alignment of flex items and also can be used to distribute flex items within container.

It can have below values.

- Flex-start (default value) – Flex items are aligned to start of container.
- Flex-end – Flex items are aligned to end of container
- Center – Flex items are aligned in center of container.
- Space-around – All flex items will have equal space around them. Since all flex items will have equal amount of space on both side, there are less space before first item and after last item compared to space between two flex items.
- Space-evenly – All flex items are distributed so that spacing between any two flex items (including space on start and end) are equal.
- Space-between – All flex items are distributed within container so that spacing between any two flex items will have equal amount of space. First will not be any space before first item and after last item.

CSS FLEXBOX



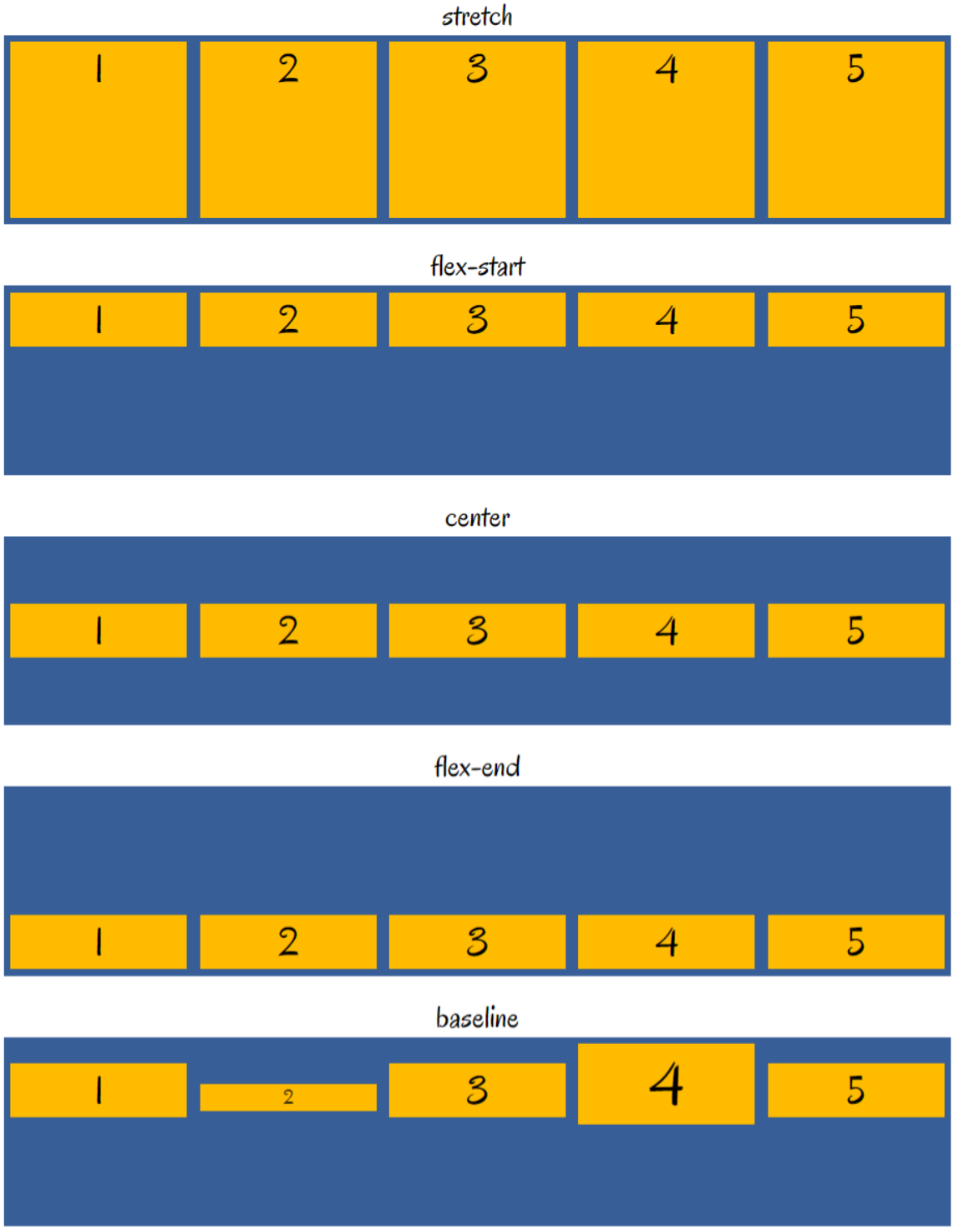
Align-items

Align-items property is used to align the flex items on cross axis. It means, if flex-direction is set to row, align-items will work on vertical (Y) axis and if flex-direction is set to column, align-items will work on horizontal (X) axis.

Align items can have below values.

- Stretch (default value) – All flex items will stretch to fill the flex container.
- Flex-start – Flex items will align to starting of a flex container.
- Center – Flex items will align to center of a flex container.
- Flex-end – Flex items will align to ending of a flex container.
- Base-line – All flex items will align such as their base line will on same line. It is generally used when flex items have text with different font size. If all flex item text have same font size it won't have any effect.

CSS FLEXBOX



CSS FLEXBOX

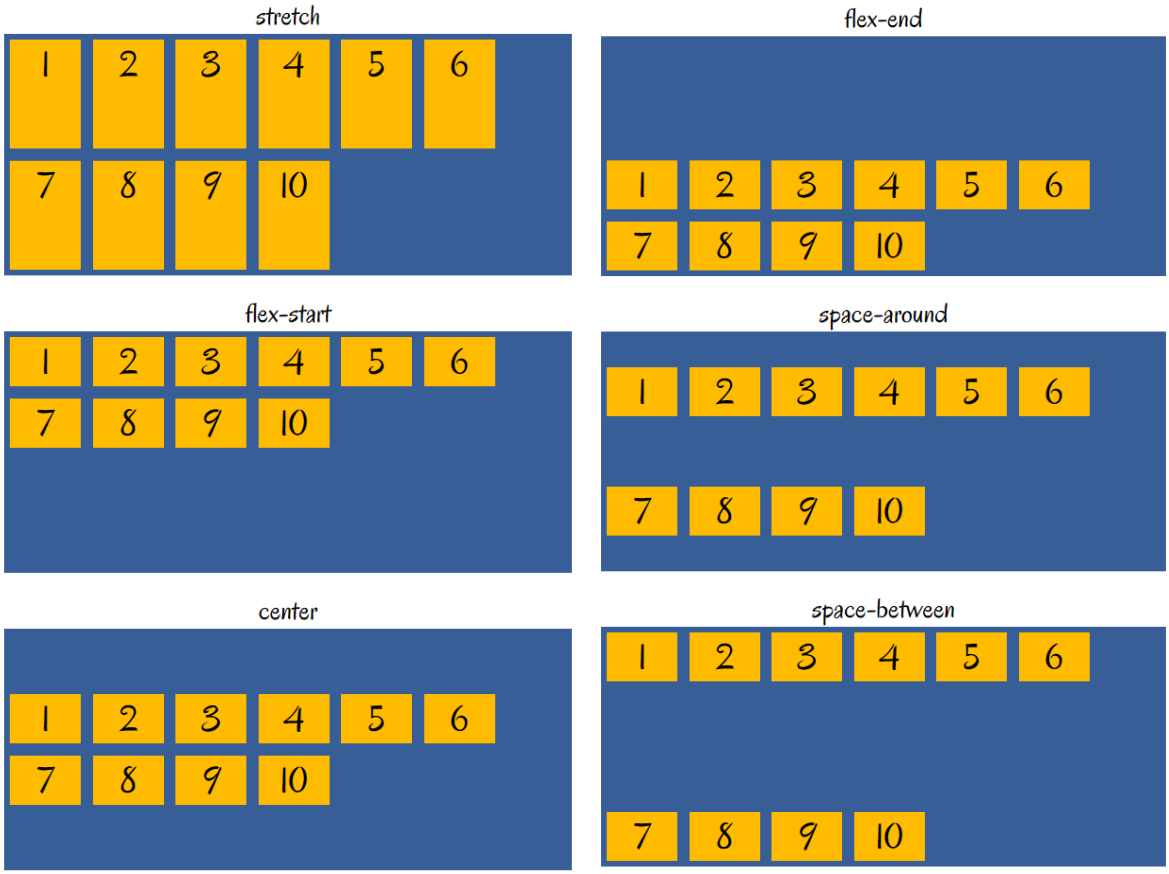
Align-content

Align-content is used to align flex container lines on cross axis where flex items are spread across multiple lines.

Align-content property will not have any effect if flex items are spread only in single line.

Align-content can have below values.

- Stretch (default) – Flex container lines is stretched to fill the container.
- Flex-start – Flex container lines is aligned to start of flex container.
- Center – Flex container lines is aligned to center of flex container.
- Flex-end – Flex container liens is aligned to end of flex container.
- Space-around – Flex container lines are aligned such that all lines have equal amount of spacing on both sides.
- Space-between – Flex container lines are aligned such that all lines have equal amount of spacing between each line. First and last lines is aligned to flex-start and flex-end.



CSS FLEXBOX

Properties of Flex Items

Below are the various properties which can be applied to individual flex items. These properties can be used to modify the behaviour of individual flex items.

Order

Order property is used to change the default order of flex items. By default, all the flex items are laid out as they are written in mark up. If we want to change the order, we can use order property.

The default value for order property is 0. So if you assign positive value to some flex item it will be moved to at the end and if you assign negative value to some flex item it will be moved to front.

You can also assign order to all flex items according to how you want them to be displayed.

Example

```
<div class="container">
  <div class="item item1">1</div>
  <div class="item item2">2</div>
  <div class="item item3">3</div>
  <div class="item item4">4</div>
  <div class="item item5">5</div>
</div>
```

```
.item5 { order: 1; }
.item3 { order: 2; }
.item2 { order: 3; }
.item1 { order: 4; }
.item4 { order: 5; }
```

CSS FLEXBOX

In above example, there are 5 flex items with class item1 to item5. Using that class we have assigned different order to each flex item to change the default order in which they are displayed.

After setting the order property like above, all flex item will be display as order displayed in below output.

Flex-grow

Flex-grow property allows the items to grow if there is an available space. It accepts the unitless values which serve as a proportional value.

If all flex-items' flex-grow value is set to 1, then all flex-item will grow to fill the container have equal width/height if flex-direction is set to row/column respectively.

Hence, we can change the width/height of individual flex items using flex-grow property.

Default

```
<div class="container">
  <div class="item item1">1</div>
  <div class="item item2">2</div>
  <div class="item item3">3</div>
  <div class="item item4">4</div>
  <div class="item item5">5</div>
</div>
```

```
.item4 {
  flex-grow: 6;
}
```

In above example we have changed the flex-grow property of flex item 4. So, if there is an available space flex item 4 will grow 6 times compared to other flex items.

Flex-shrink

Flex-shrink property allows to set how an individual flex item will shrink in proportion to other flex items. Hence if there is no enough room for all flex items in container, then item which have higher flex-shrink value will shrink more compared to other flex items.

```
<div class="container">
  <div class="item item1">1</div>
  <div class="item item2">2</div>
  <div class="item item3">3</div>
  <div class="item item4">4</div>
  <div class="item item5">5</div>
</div>
```

```
.item4 {
  flex-shrink: 2;
}
```

Here Item4 will shrink more compared to other flex item because we have set the flex-shrink value to 2 in item4.

CSS FLEXBOX

Flex-basis

Best use of flex-basis is with in conjunction of flex-grow and flex-shrink. Lets consider below case.

Case 1:

```
.item1 {
  flex-grow: 1;
  flex-shrink: 0;
  flex-basis: 200px;
}
```

Let's say that item1 have flex-basis of 200px, flex-grow of 1 and flex-shrink of 0. So here flax-basis will act like minimum width, it means minimum width of item1 will be 200px but if there is a space it will be allowed grow more than 200px as flex-grow is set to 1.

Case 2:

```
.item1 {
  flex-grow: 0;
  flex-shrink: 1;
  flex-basis: 200px;
}
```

Let's say that item1 have flex-basis of 200px, flex-grow of 0 and flex-shrink of 1. So here flex-basis will act like maximum width, it means that maximum width of item1 will be 200px but if there not enough room it will be allowed to shrink less then 200px as flex-shrink is set to 1.

CSS FLEXBOX

Case 3:

```
.item1 {
  flex-grow: 0;
  flex-shrink: 0;
  flex-basis: 200px;
}
```

Let's say that item1 have flex-basis of 200px, flex-grow of 0 and flex-shrink of 0. So here flex-basis will act like width of an item, it means that width of an item1 will be exactly 200px and is not allowed to grow or shrink as flex-grow and flex-shrink is set to 0.

Flex

Flex is a shorthand property for flex-grow, flex-shrink and flex-basis.

Default value for flex property is **0 1 auto**.

```
Syntax

Flex: flex-grow flex-shrink flex-basis
```

Here second and third parameter (flex-shrink and flex-basis) is optional.

Align-self

Default alignment of flex item is set using align-items property. Align-self property is used to override the default alignment if an individual flex item.

CSS FLEXBOX

```
<div class="container">
  <div class="item item1">1</div>
  <div class="item item2">2</div>
  <div class="item item3">3</div>
  <div class="item item4">4</div>
  <div class="item item5">5</div>
</div>
```

```
.container {
  display: flex;
  flex-direction: row;
  align-items: flex-start;
}

.item4 {
  align-self: center;
}
```

In above example default alignment of all flex item is set to flex-start using align-items. So, all flex items will be aligned at flex-start. And in item4 we have used align-self to center to align only item4 to center instead of flex-start. Below is the output of code.

CSS FLEXBOX

Examples

Checkout various examples of Flexbox at My Codepen

Link to My Flexbox Codepen Collection:

https://codepen.io/collection/neZrNb/

Link to My Codepen:

https://codepen.io/sumitpmakwana/