FLEXBOX

Without FlexBox

- Hard to maintain your layouts, box models, align-items, free space etc.
- You will have to constantly use properties such as float, positional properties, fixed heights, and other tricks to be able to accomplish even simple tasks.

With FlexBox

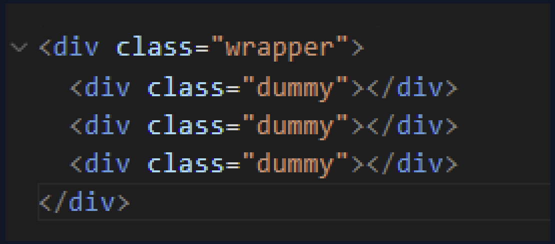
- Removes, the complexity that we had before!
- Easy to maintain and accomplish almost any layout that you wish (for 2D layouts GRID is most preferred)
- Brings extremely powerful tools! Keep things easy

Short Answer: If you want to build something you must learn flexbox!

HOW TO ACTIVATE IT ? HOW IT WORKS ?

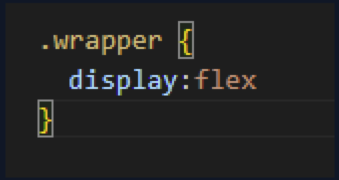
SIMPLE USAGE

HTML



```
<div class="wrapper">
  <div class="dummy"></div>
  <div class="dummy"></div>
  <div class="dummy"></div>
</div>
```

CSS

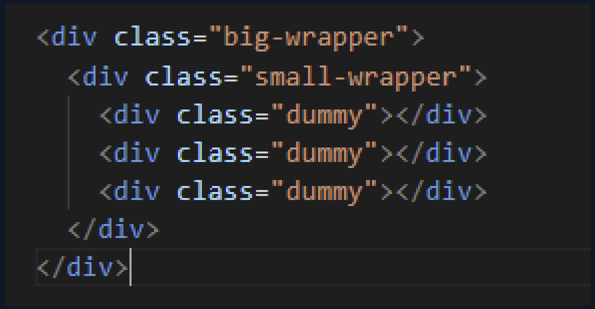


```
.wrapper {
  display:flex
}
```

- “wrapper” div became flex container!
- “dummy” divs became flex items!

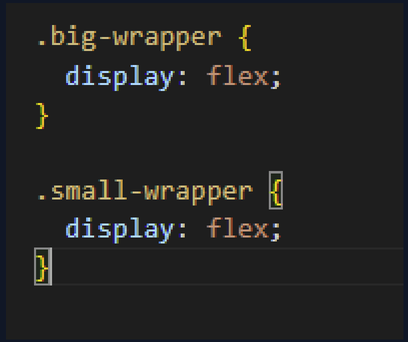
NESTED USAGE

HTML



```
<div class="big-wrapper">
  <div class="small-wrapper">
    <div class="dummy"></div>
    <div class="dummy"></div>
    <div class="dummy"></div>
  </div>
</div>
```

CSS

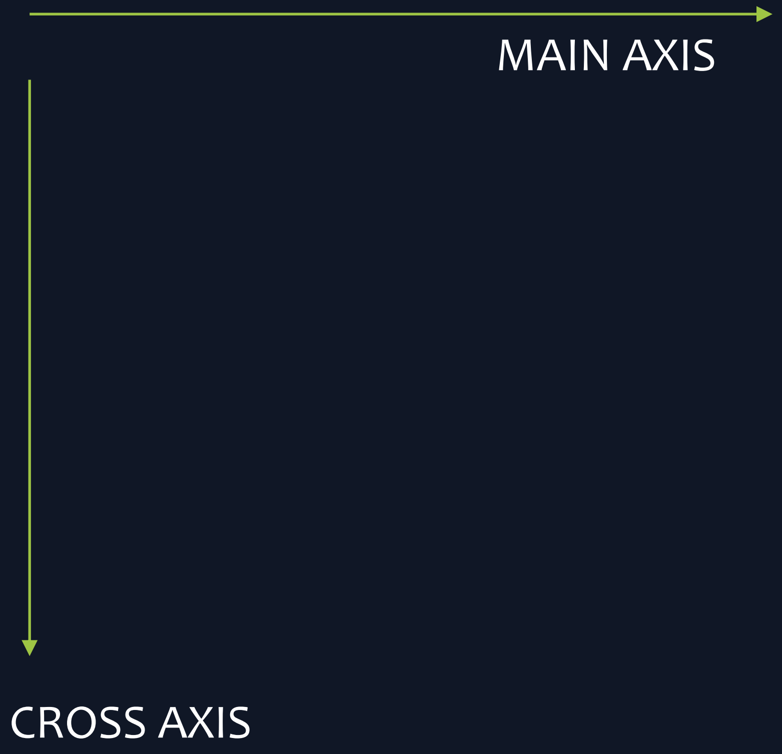


```
.big-wrapper {
  display: flex;
}

.small-wrapper {
  display: flex;
}
```

- “big-wrapper” became flex container
- “small-wrapper” became flex item as well as flex container
  - It can use “flex container & flex item” properties.
- “dummy” divs became “flex items”

- You can do this nested as much as you want! There is no limit!
- Understanding what is flex container, what is flex item is important!

HOW TO ACTIVATE IT ? HOW IT WORKS ?



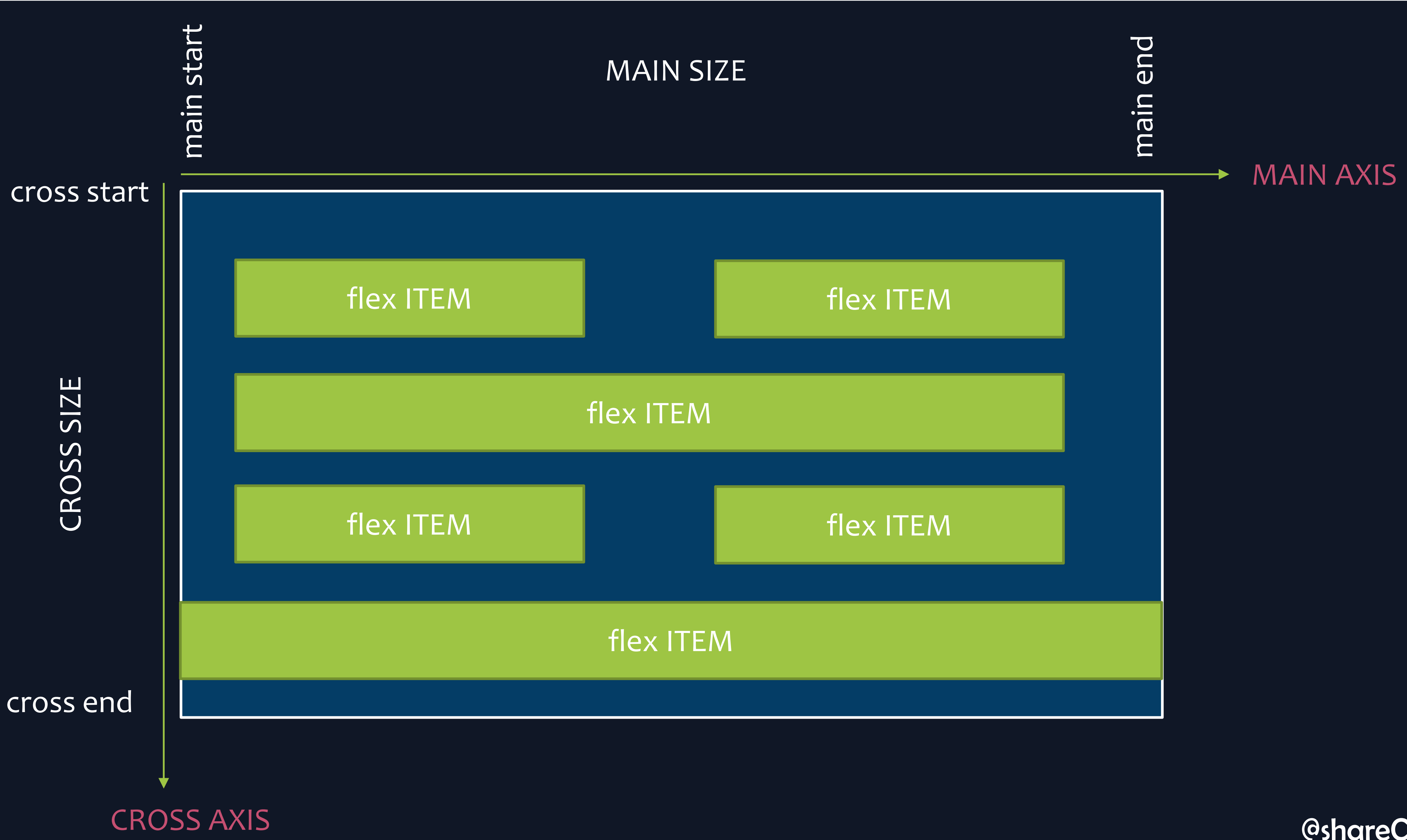
MAIN AXIS

CROSS AXIS

- Flex-direction will change AXISES!
- Some properties will only work on “main axis”
  some will only work on “cross axis”

- Flex-wrap will not swap axises!

FLEX CONTAINER!



@shareCodeCamp

OVERVIEW EVERYTHING

FLEX CONTAINER

- display
- justify-content
- align-items
- align-content

- flex-direction
- flex-wrap
- flex-flow*

- row-gap
- column-gap
- gap*

FLEX ITEM

- flex-grow
- flex-shrink
- flex-basis
- flex*
- order
- align-self

* means shorthand

MAIN & CROSS

PROPERTIES FOR MAIN AXIS

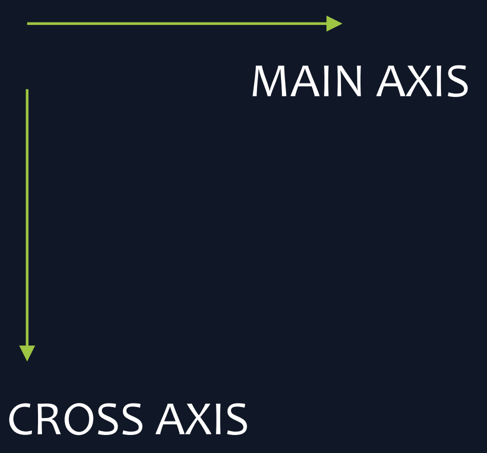
- justify-content

PROPERTIES FOR CROSS AXIS

- align-items
- align-self
- align-content

FLEX DIRECTION

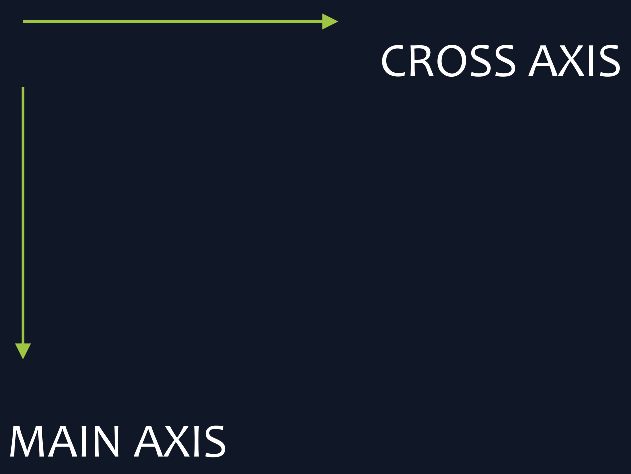
flex-direction: row;
(DEFAULT)



MAIN AXIS

CROSS AXIS

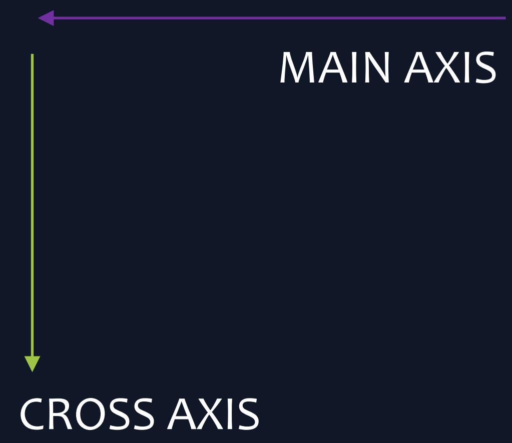
flex-direction: column;



CROSS AXIS

MAIN AXIS

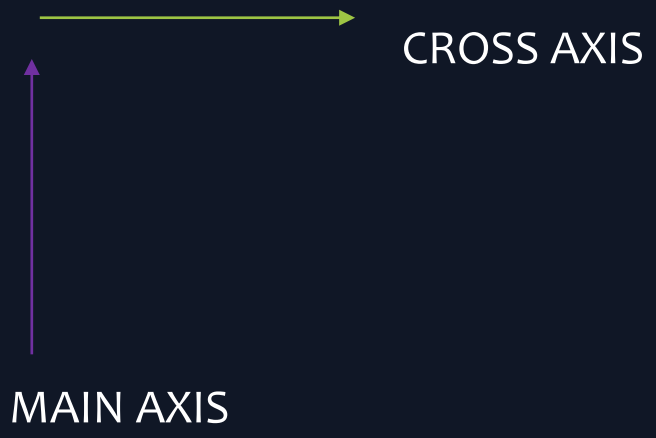
flex-direction: row-reverse;



MAIN AXIS

CROSS AXIS

flex-direction: column-reverse;

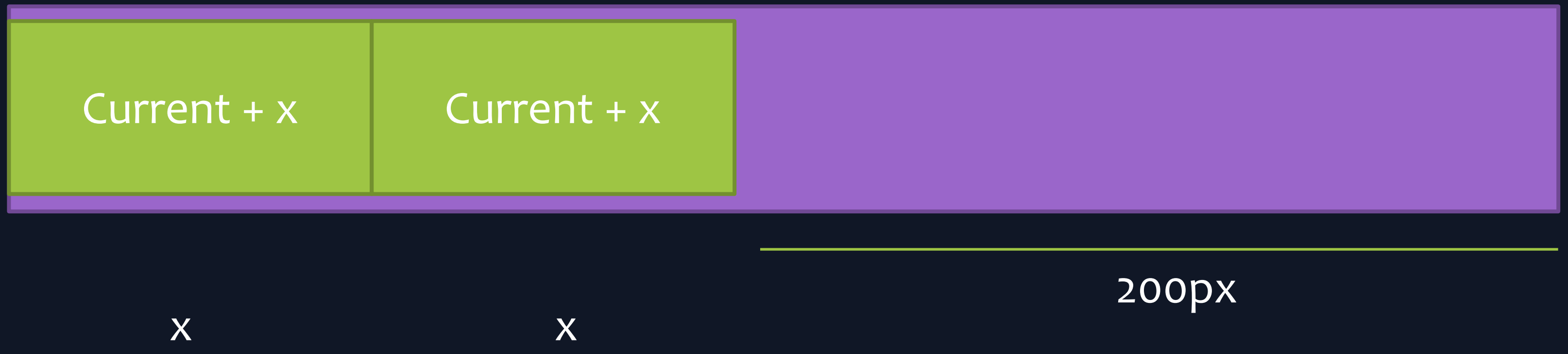


CROSS AXIS

MAIN AXIS

@shareCodeCamp

FLEX GROW



Current + x

Current + x

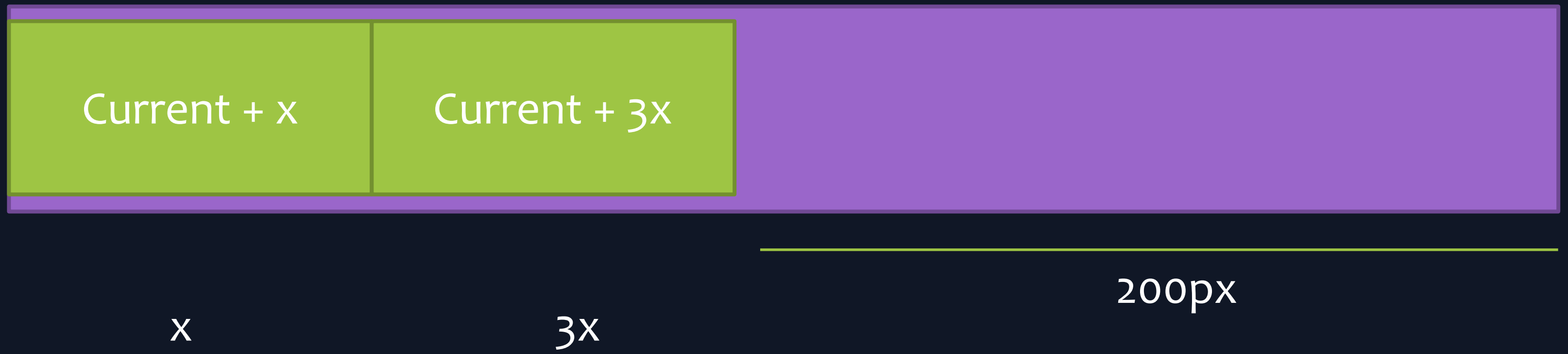
200px

x

x

2x = 200

x = 100



Current + x

Current + 3x

200px

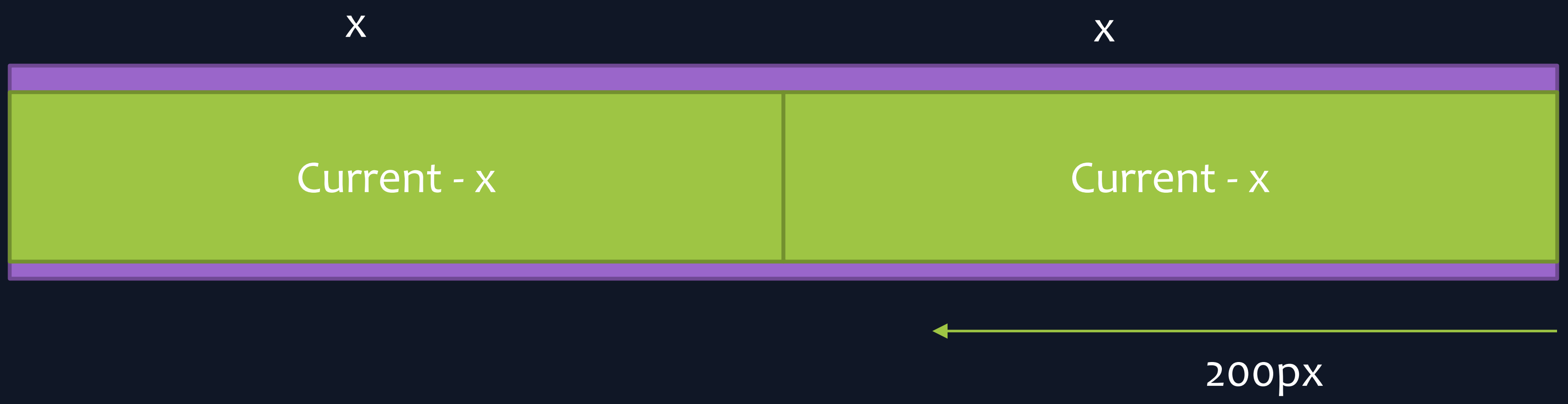
x

3x

4x = 200

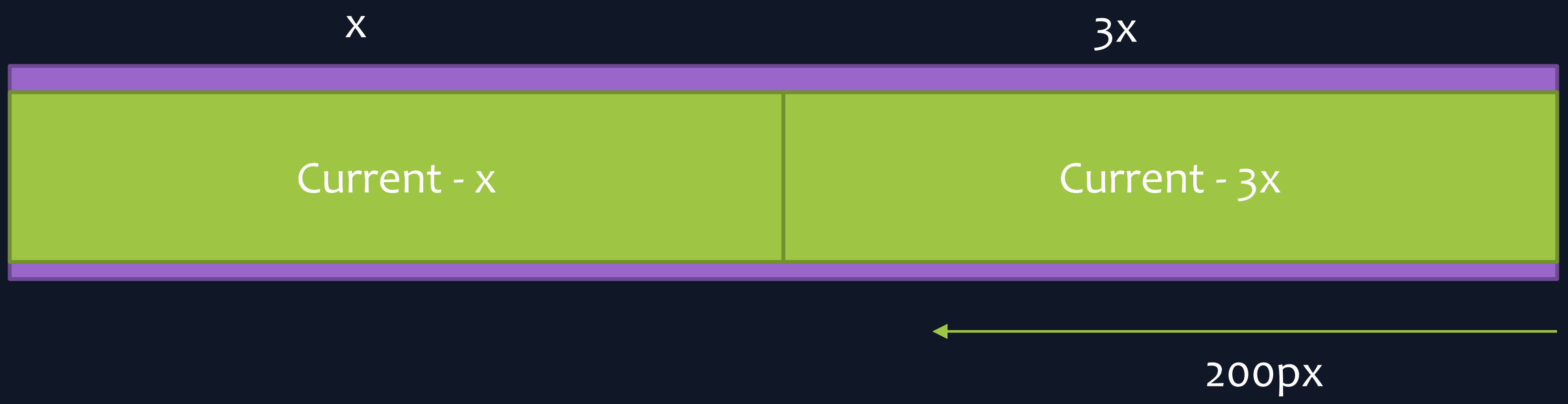
x = 50

FLEX SHRINK



2x = 200

x = 100

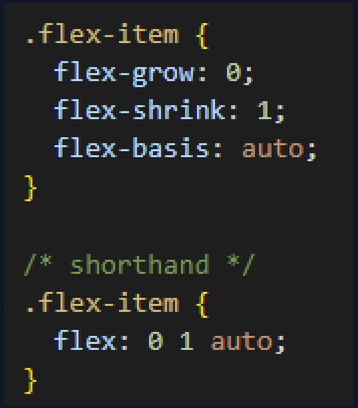


4x = 200

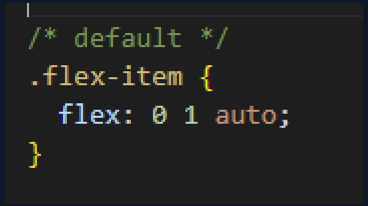
x = 50

FLEX (SHORTHAND)

flex: <flex-grow> <flex-shrink> <flex-basis>

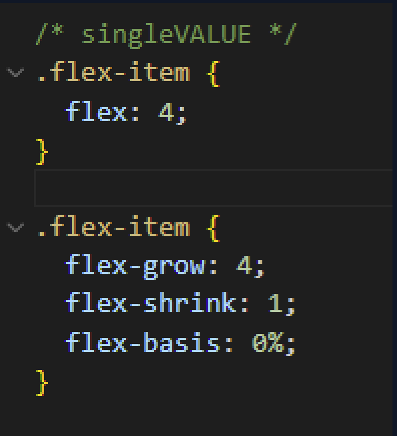


```css
.flex-item {
  flex-grow: 0;
  flex-shrink: 1;
  flex-basis: auto;
}

/* shorthand */
.flex-item {
  flex: 0 1 auto;
}
```

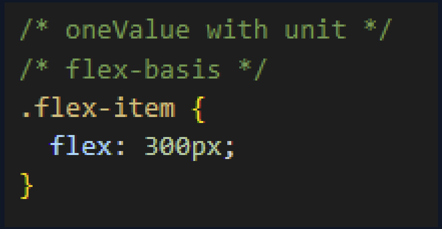


```css
/* default */
.flex-item {
  flex: 0 1 auto;
}
```

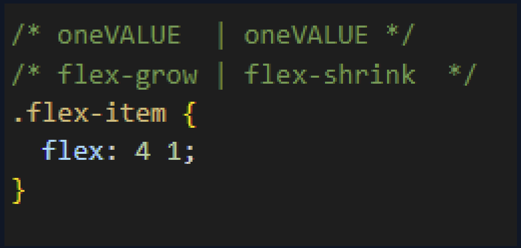
USAGE CASES



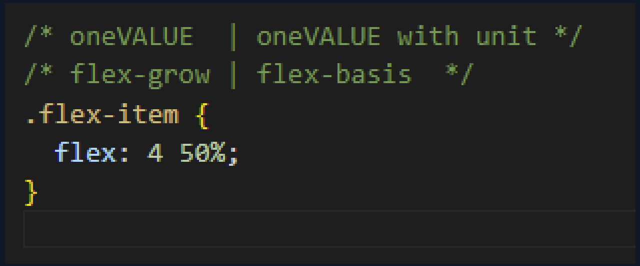
```css
/* singleVALUE */
.flex-item {
  flex: 4;
}

.flex-item {
  flex-grow: 4;
  flex-shrink: 1;
  flex-basis: 0%;
}
```



```css
/* oneValue with unit */
/* flex-basis */
.flex-item {
  flex: 300px;
}
```



```css
/* oneVALUE  | oneVALUE */
/* flex-grow | flex-shrink  */
.flex-item {
  flex: 4 1;
}
```



```css
/* oneVALUE  | oneVALUE with unit */
/* flex-grow | flex-basis  */
.flex-item {
  flex: 4 50%;
}
```

@shareCodeCamp