



**FLEXBOX**



# FLEX BOX

CSS only recently got real layout tools (CSS Flexbox and Grid) - for years we were stuck with Floats / Clears

# FLOAT LAYOUTS



# WHY?

CSS was written a LONG time ago, in a galaxy far far away (called the 1990s).

There was no concept of a multi-device universe or interactive websites. Everything was based around print layouts.



# FLOATS

That why's we have the float-based system - it comes from print design where you "float" images in a sea of words - like in a magazine.



The web doesn't  
**FLOAT**  
anymore







Now we...

**FLEX**

How does  
**FLEXBOX**  
work?



# PARENT (CONTAINER)



Flex containers are the objects that contain the children



# CHILDREN (ITEMS)



Flex items are the children that go inside the parent container



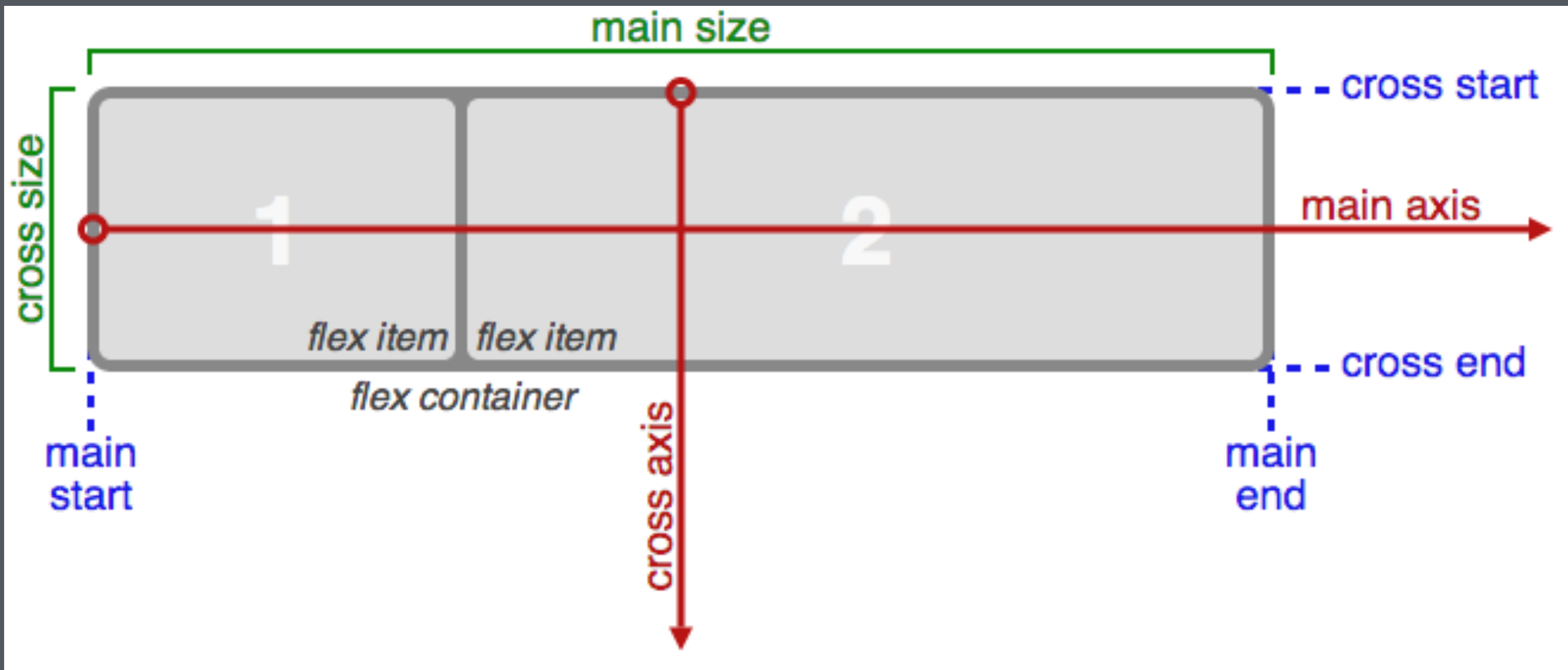
# FLEXBOX IN ACTION





# HOW TO FLEXBOX

## Multi-axis alignment method





# HOW TO FLEXBOX

.items



.container

# HOW TO FLEXBOX

Easy to start, harder to use well

```
.container {  
  display: flex;  
  justify-content: center;  
  align-items: center;  
}  
  
.item {  
  background-color: orange;  
  height: 100px;  
  width: 100px;  
}
```



When building

# FLEXBOX LAYOUTS

Every Container Must Have:

```
display: flex;
```

Which also gives you the following for free:

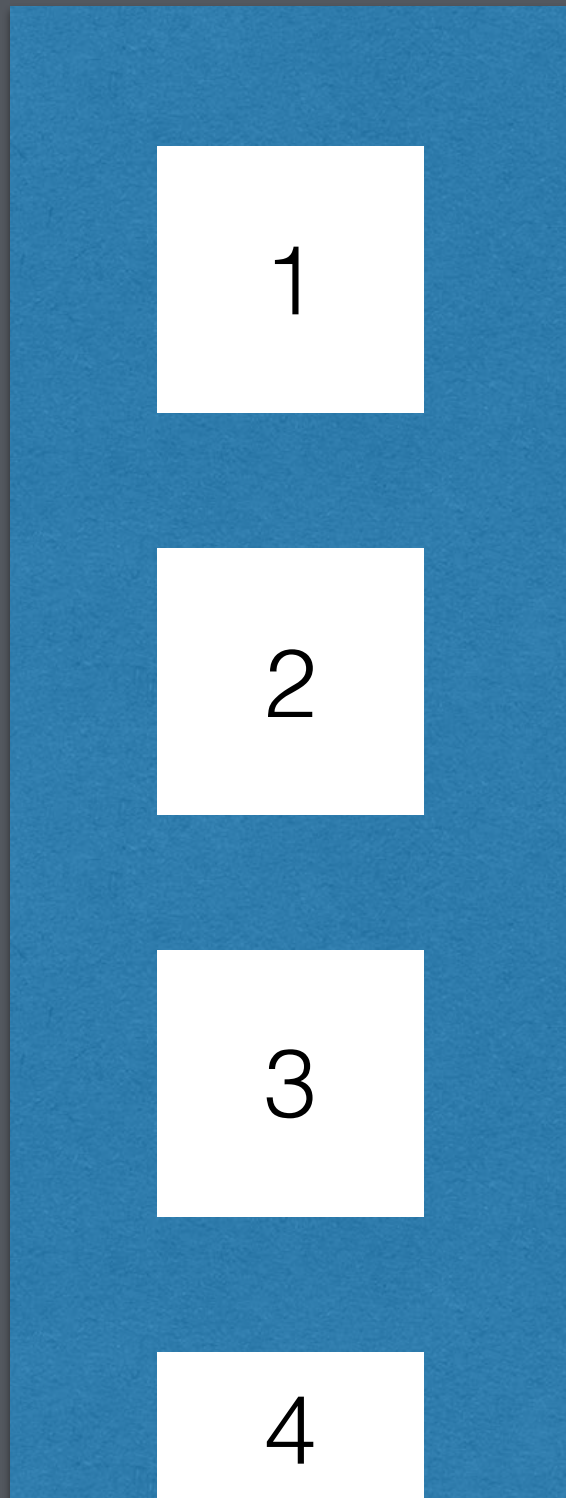
```
flex-direction: row;  
justify-content: flex-start;  
align-items: stretch;
```

# PARENT PROPERTIES





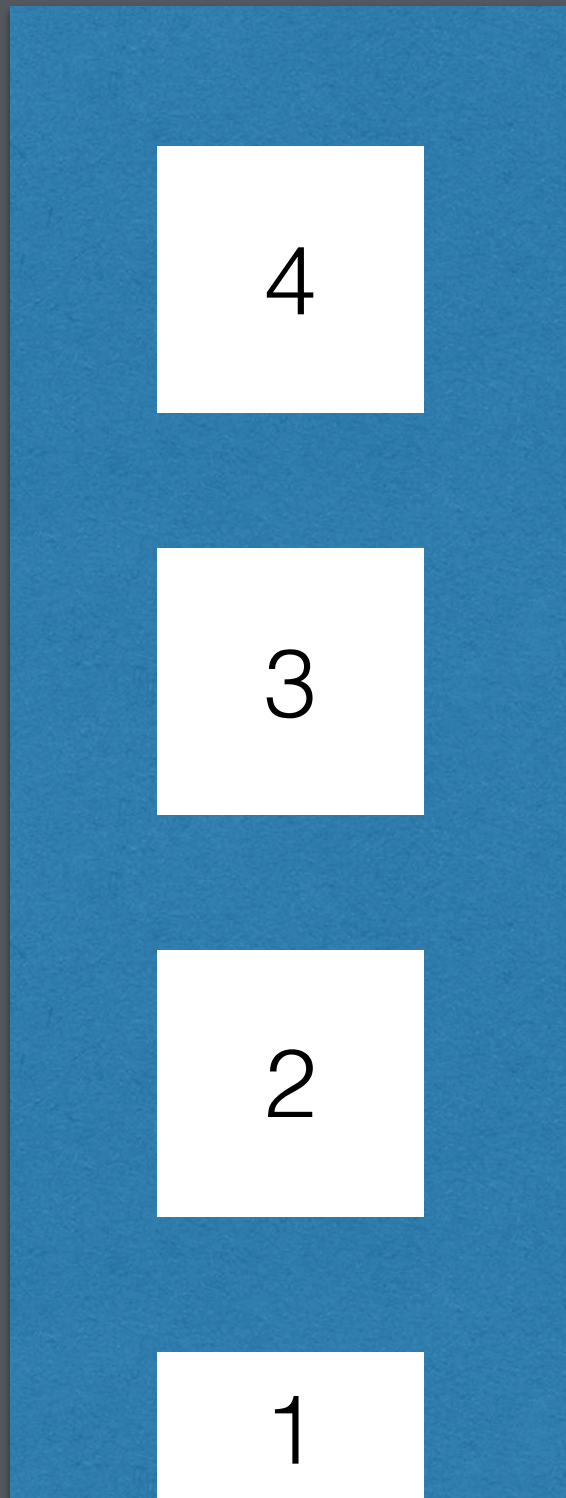
# FLEX-DIRECTION



Think about orientation -  
flexbox layouts are  
inherently vertical or  
horizontal

```
flex-direction: column;
```

# FLEX-DIRECTION



You can easily flip the display order without reordering your HTML!

```
flex-direction: column-reverse;
```



# FLEX-DIRECTION

1

2

3

4

5

You can also do layouts in a row.

```
flex-direction: row;
```

# FLEX-DIRECTION

5

4

3

2

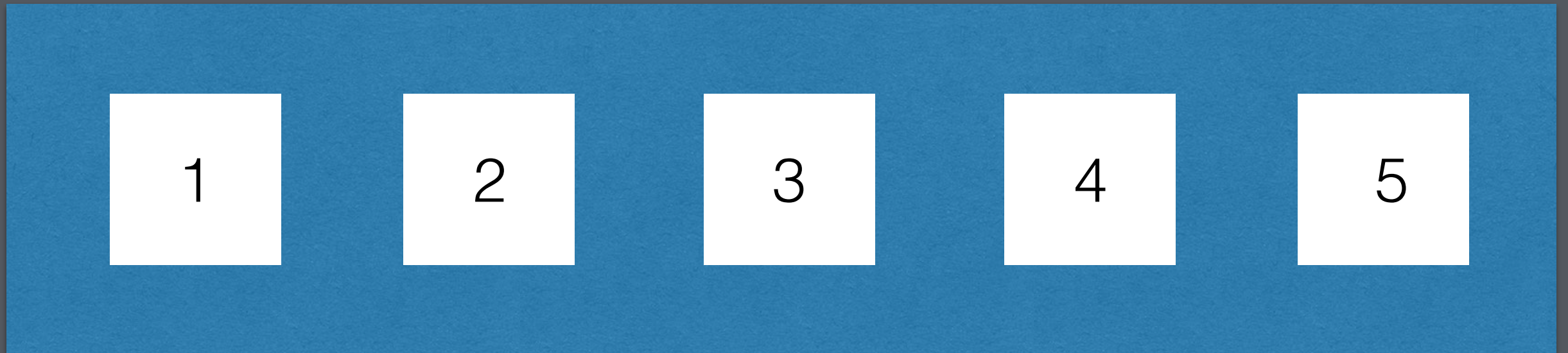
1

You can also flip rows - this is very advantageous for right-to-left languages like Arabic.

```
flex-direction: row-reverse;
```



# FLEX-WRAP

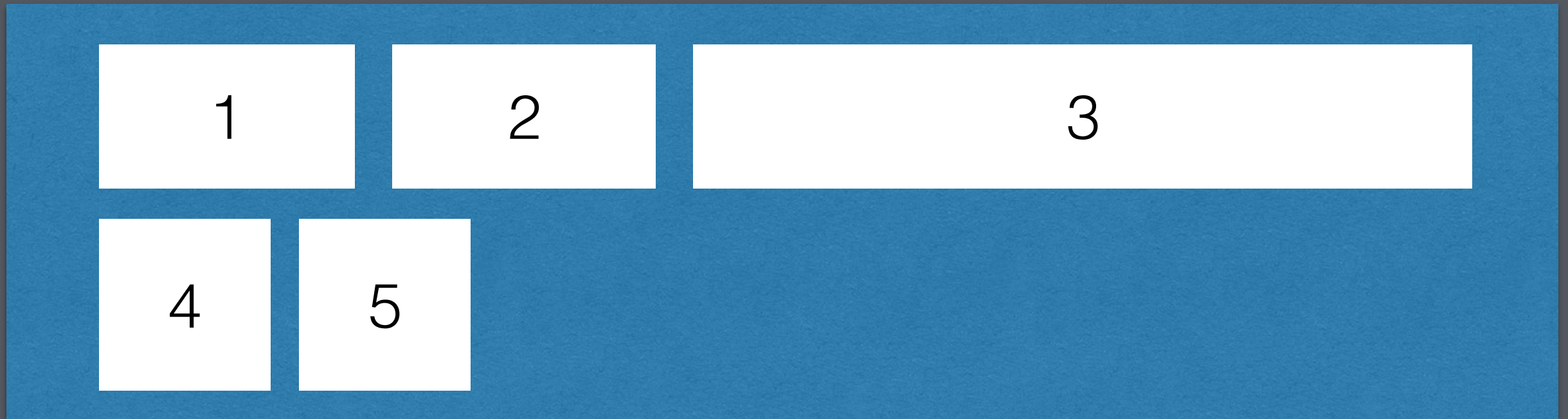


By default, all boxes are stuffed into one row.

```
flex-wrap: nowrap;
```



# FLEX-WRAP

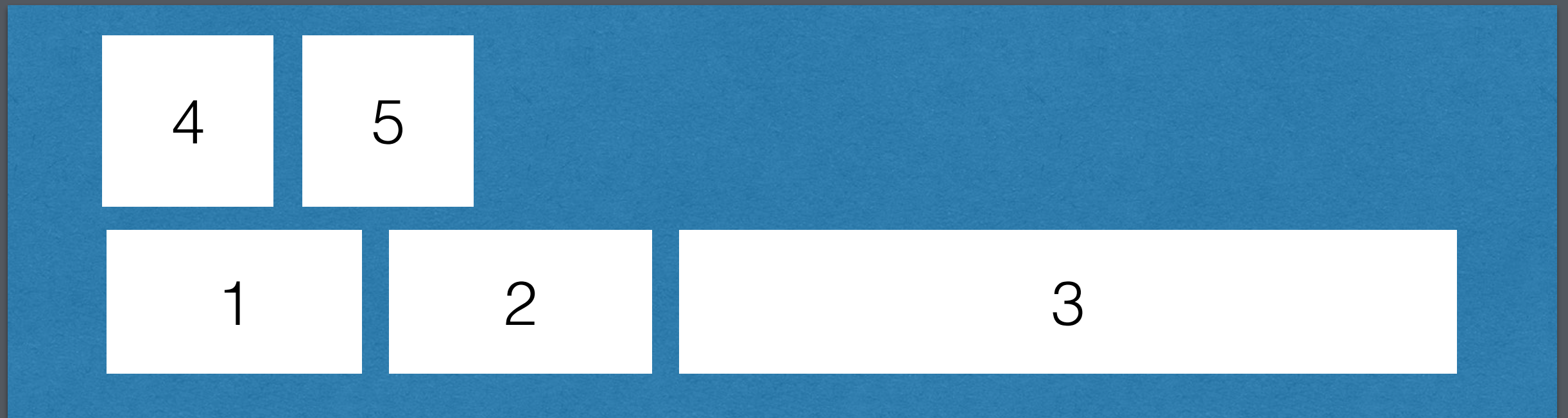


But you can make them  
pop-out into additional rows  
as needed.

```
flex-wrap: wrap;
```



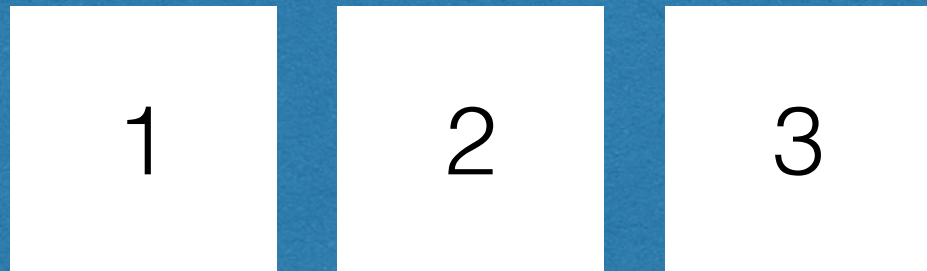
# FLEX-WRAP



They can display right to left as well and bottom-to-top (I find this very confusing personally and don't use it).

```
flex-wrap: wrap-reverse;
```

# JUSTIFY-CONTENT



Controls how boxes space  
in flexbox rows/columns.

```
justify-content: flex-start;
```



# JUSTIFY-CONTENT



Centering is easy - note,  
auto margins don't work in  
flex land.

```
justify-content: center;
```

# JUSTIFY-CONTENT



1

2

3

Push everything right,  
similar to `text-align: right;`  
but for layouts!

```
justify-content: flex-end;
```



# JUSTIFY-CONTENT

1

2

3

Pushes stuff as far apart as possible.

```
justify-content: space-between;
```

# JUSTIFY-CONTENT

1

2

3

Centers with respect to total row/column, equal spacing between each item.

```
justify-content: space-around;
```



# **ALIGN-ITEMS**

Controls vertical alignment -  
hurrah! Only took CSS 20  
years...

# ALIGN-ITEMS

1

2

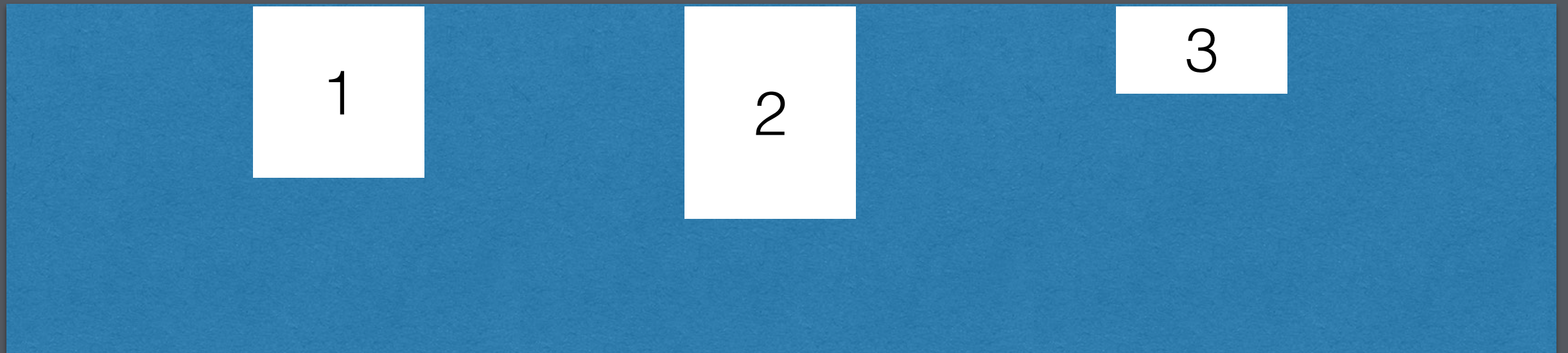
3

Centers children items  
vertically

```
align-items: center;
```



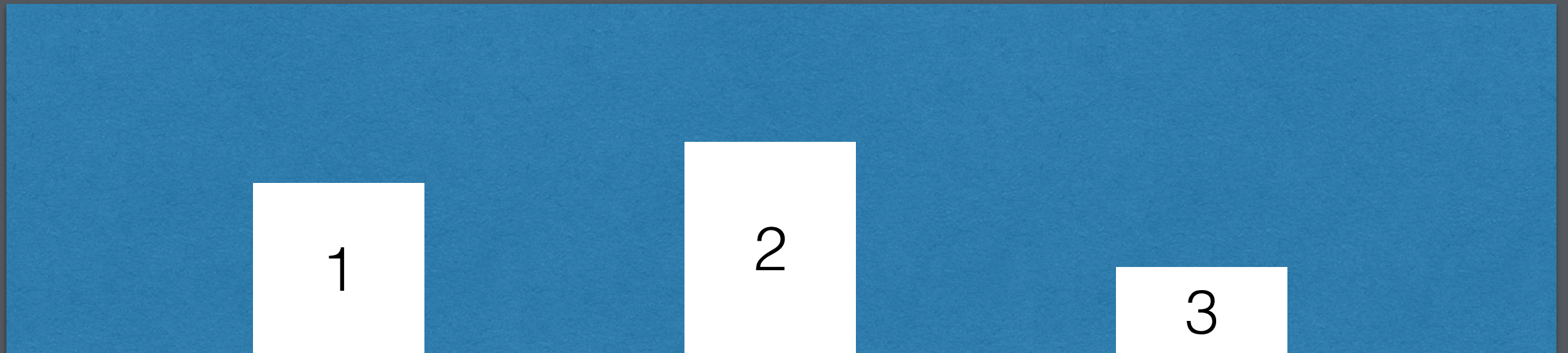
# ALIGN-ITEMS



Top-aligns children, even if  
they have different heights

```
align-items: flex-start;
```

# ALIGN-ITEMS

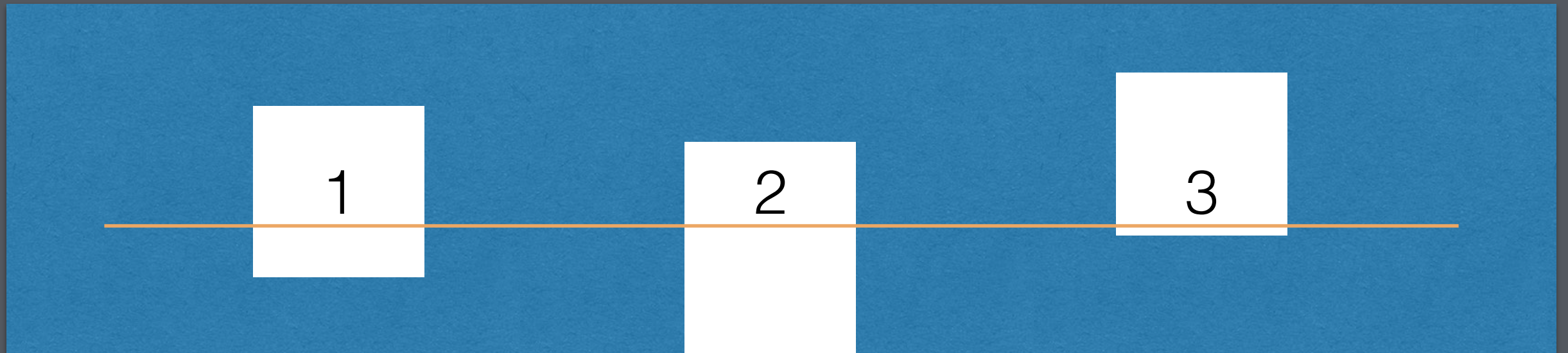


Bottom-aligns children, even  
if they have different heights

```
align-items: flex-end;
```



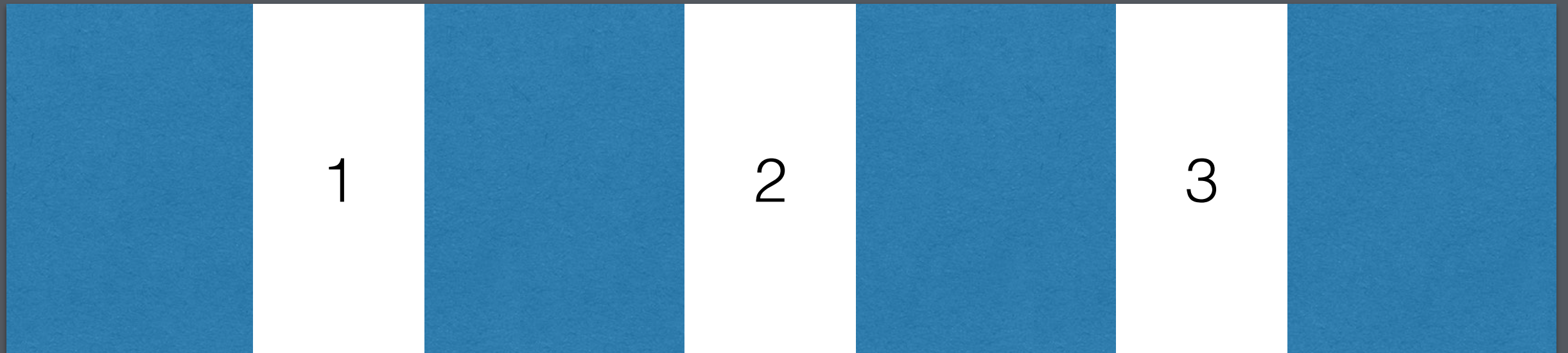
# ALIGN-ITEMS



Aligns children by the middle baseline of the text in the child (note orange line)

```
align-items: baseline;
```

# ALIGN-ITEMS



Stretches children to be the size of the container (no height property)

```
align-items: stretch;
```



# CHILDREN PROPERTIES





# ORDER

3

1

2

Change the order in which  
children render within a row.

```
order: 3;  
order: 1;  
order: 2;
```



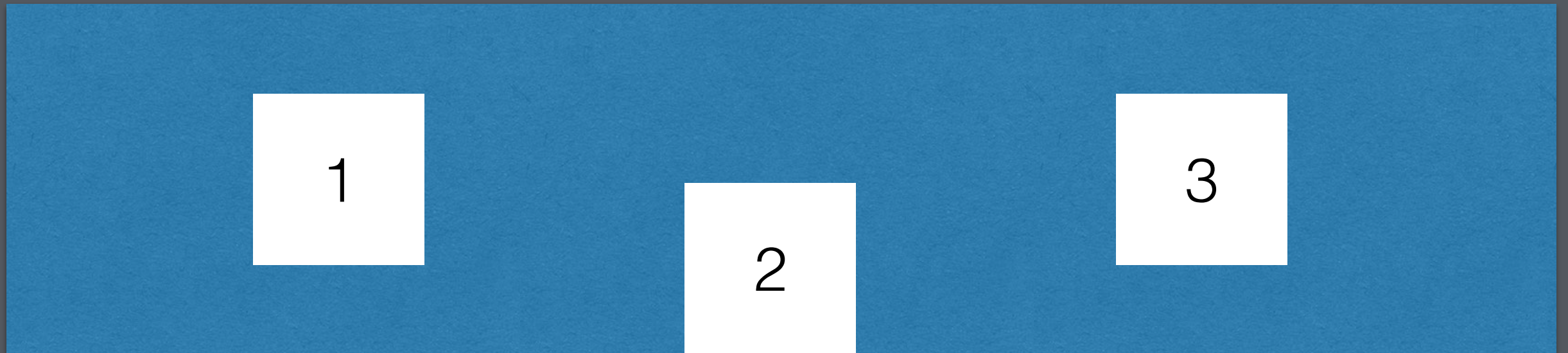
# FLEX-GROW



Controls the rate at which individual children grow in width across viewports

```
flex-grow: 2;  
flex-grow: 0.5;  
flex-grow: 1;
```

# ALIGN-SELF



Works like align-items but for an individual child, so you can override on a per-item basis.

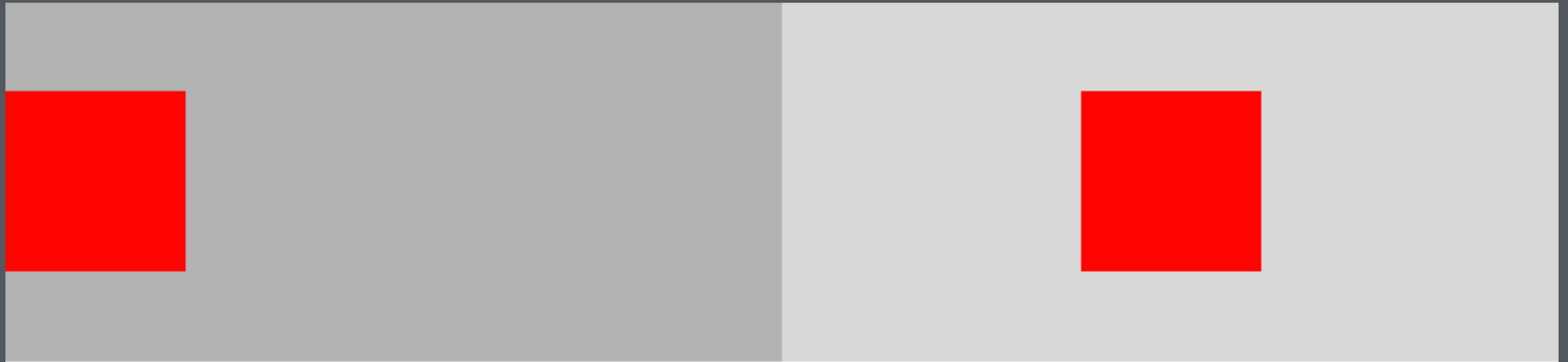
```
align-self: flex-end;
```



# DOUBLE FLEX?



# YES YOU CAN



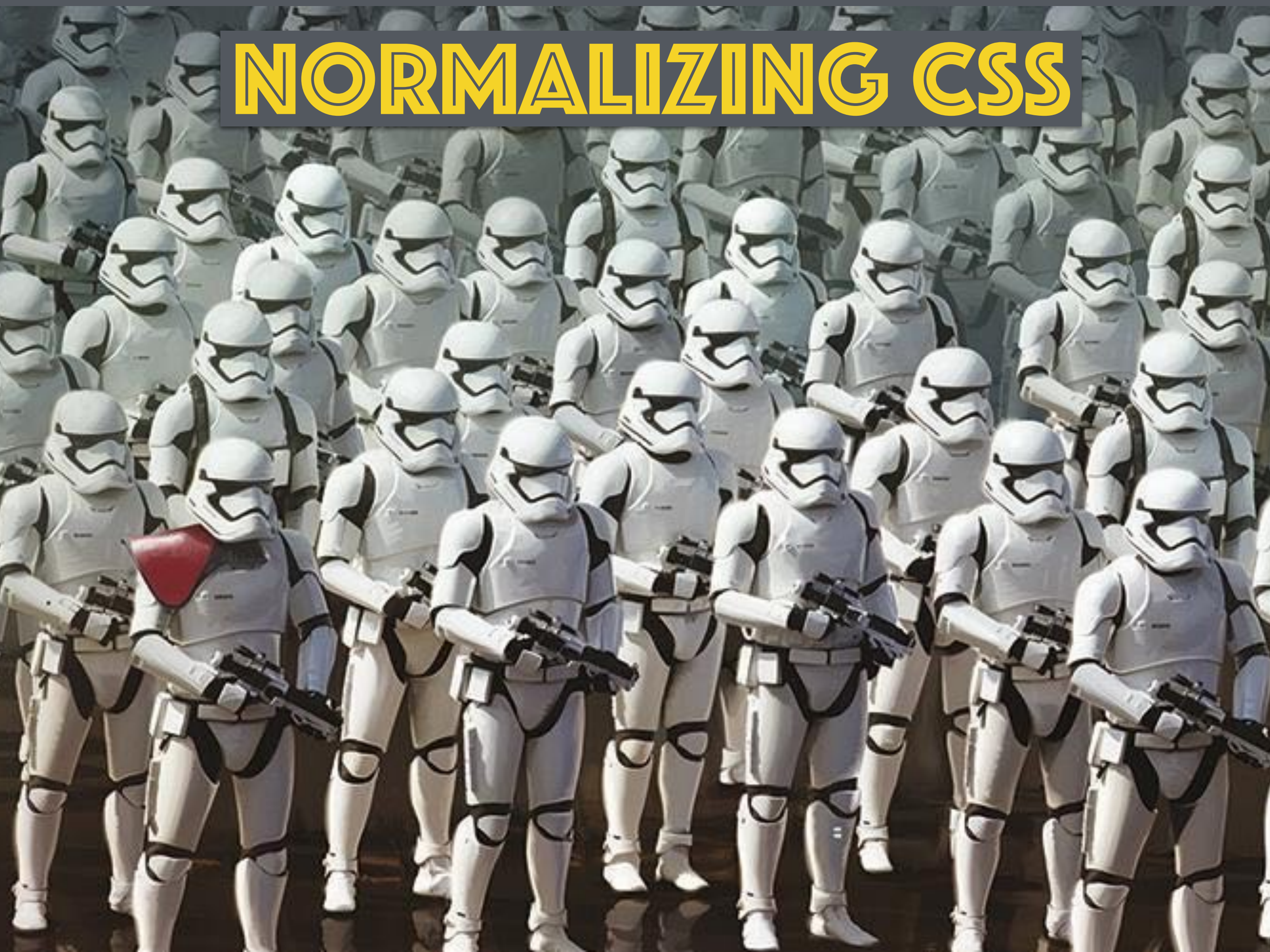
You can nest a flex container inside of a flex container - no problem.

# YOUR TURN

Flex on Assignment #3



# NORMALIZING CSS





# NORMALIZING OUTPUT

- Browsers are all a little unique in how they render things
- Very smart people have compared and contrasted these very minor differences and fixed them for you - how nice.
- There are many of them but I'm going to make your life simple and point you to the best one:

Normalize.css: <http://necolas.github.io/normalize.css/>

# HOW TO USE NORMALIZE

```
<head>  
  <title>Something Unique</title>  
  <link rel="stylesheet" href="css/normalize.css">  
  <link rel="stylesheet" href="css/main.css">  
</head>
```

- 1) Download normalize
- 2) Place normalize CSS before your external CSS
- 3) Code away like normal

Let's check it out in action:

<http://codepen.io/staypuftman/pen/VjPEpJ>



**USE NORMALIZE ON  
THIS WEEKS HW**

# NEXT TIME

HW #2 - Practice Flexbox Layouts  
Next time, we'll explore CSS Grid