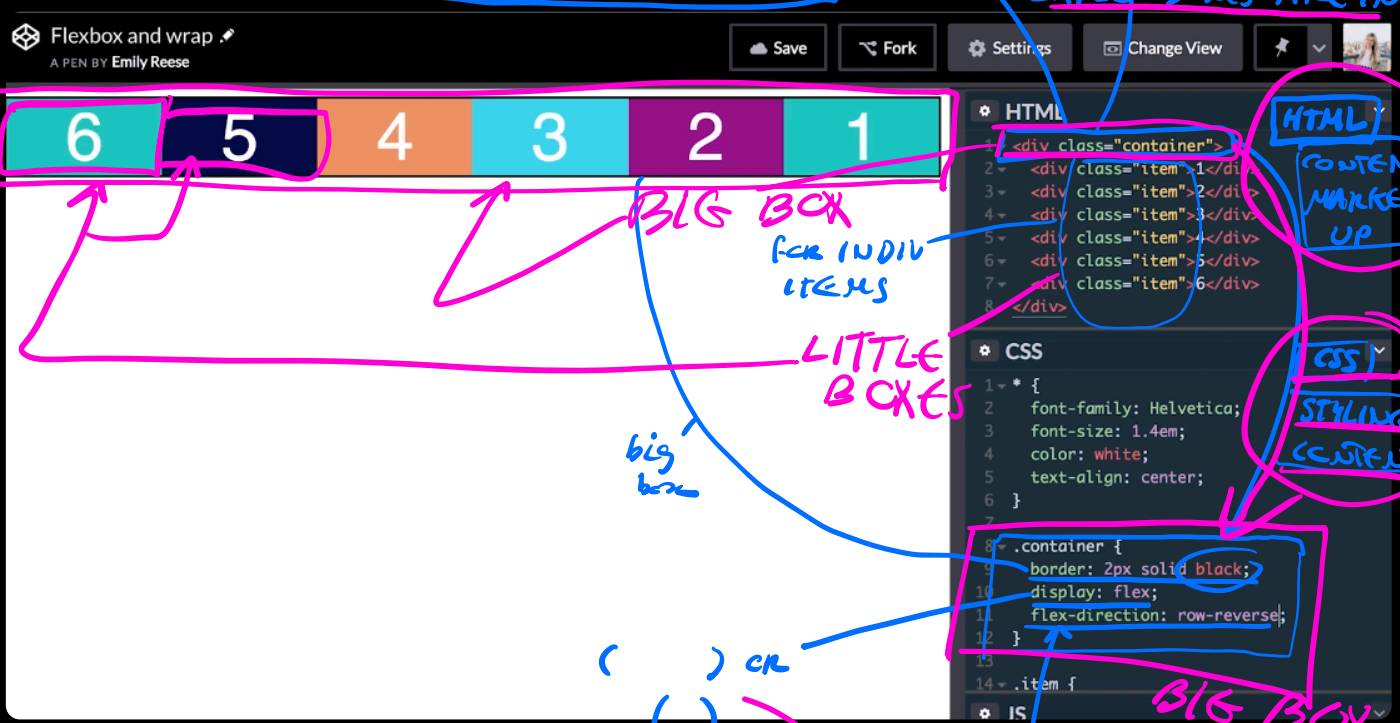
FLEXBOX: STACKING THE LITTLE BOXES IN 1D VECTORS: ( ) OR ( )



FOR BIG BOX LITTLE BOXES ARE IN

BIG BOX FOR INDIV ITEMS

LITTLE BOXES

big box

HTML CONTENT MARKED UP

CSS STYLING CONTENT

( ) OR ( )

BIG BOX STYLING

FLEX DIRECTION = LITTLE BOXES STACKED IN ( )'S VS ( )'S

row = ( )

column = ( )

row-reverse = ( ) little boxes backwards in it

column-reverse = ( )

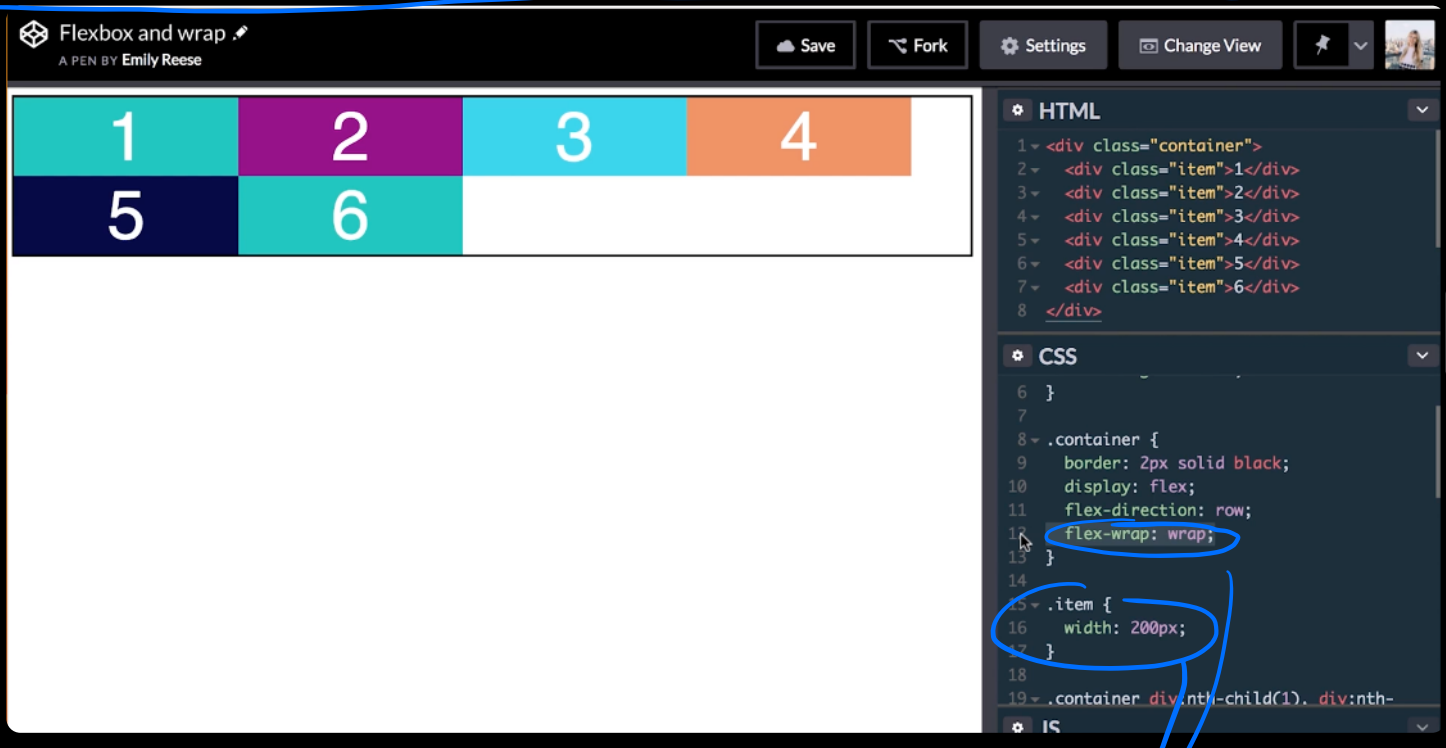
FLEX MEANS BIG BOX = ( ) OR ( ) VECTOR, NOT A GRID

column-reverse

not column-reverse

TELLING IT ( ) OR ( )

FLEX-WRAP: NOT FORCING THE LITTLE BOXES TO BE A CERTAIN SIZE



→ CSS is a scripting lan.

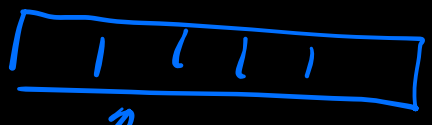
FORCING ALL CN BOXES ALL CN ONE RLCI OR PUSHING BOXES CNTO A NEW LINE

→ without flex-wrap → it resized all the boxes to force them onto one row

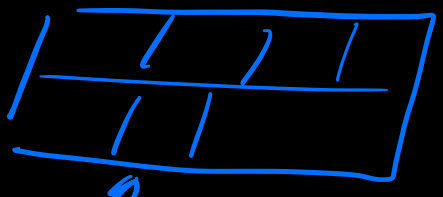
→ wrap is telling it to resize the boxes according to desc (the css for the (boxes) + stack the left over ones on a new row

→ without wrap → it resizes them all to fit them onto the row

→ with wrap → it lets them be a specified space and fits the remaining boxes on a new row

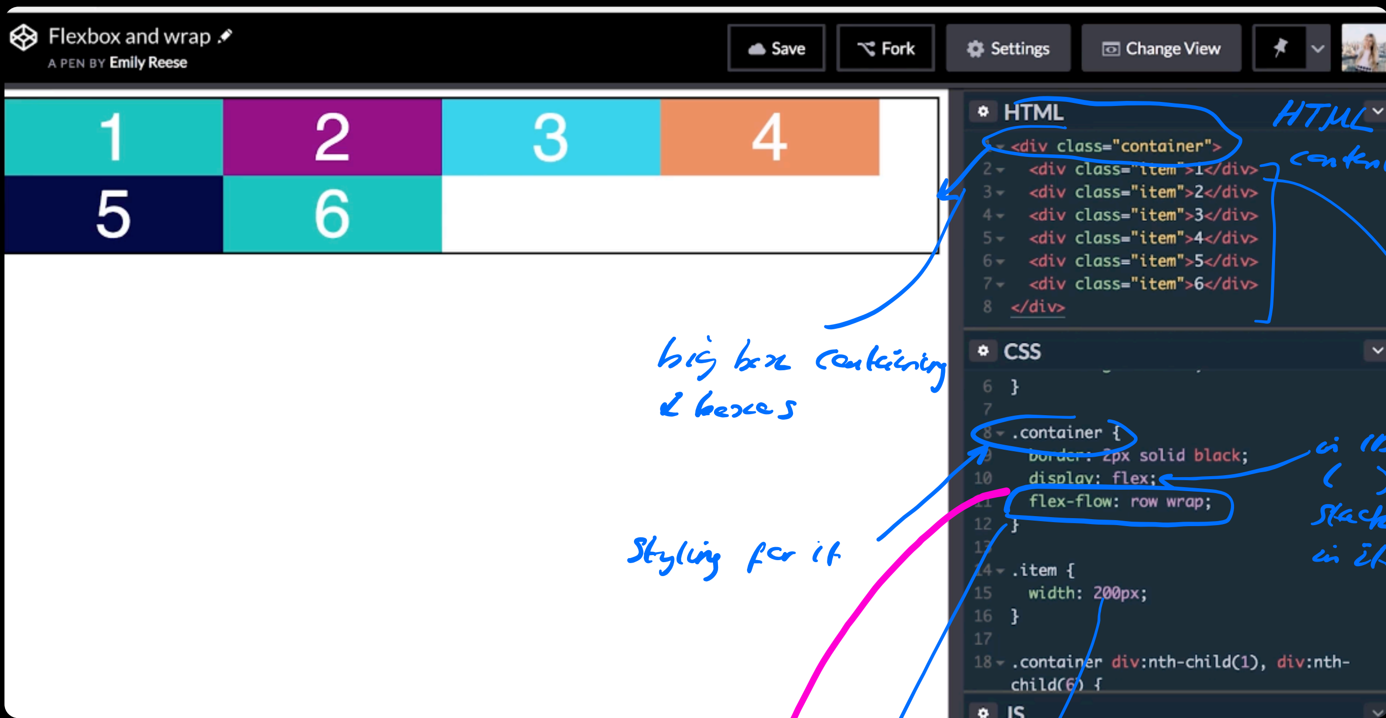
= 

all sized according to screen size

vs 

→ used ( ) or ( ) :: = all set size 1D flexbox

FLEXFLOW: FLEX DIRECTION, FLEX WRAP, COMBINED ONTO ONE CSS LINE IN SHORTHAND



COMBINING IT INTO SHORTHAND FOR ID BIG BOX IN CSS

how the ↓ boxes are stacked in it will be
→ in a row
→ unwrapped (so – not resized to force them all onto one line) – 200px width each (if uses the ↓ box size specified elsewhere + fits the extra space onto a new row

3. Align items and justify content

flex direction = row

→ flexbox is a ↑ box containing the ↓ boxes in a row or a column
→ this video is how to align ↓ boxes in that ↑ er ID flexbox

→ main vs cross axis
↑
L → R = main axis (flex)

"TO "JUSTIFY" A LITTLE BOX"

"justify" is code for force the ↓ boxes along one side of the ↑ box

"justify" centre   "justify" left

→ if the box = a r or c flexbox (vs 2D flex box)

MAIN AX VS CROSS AX IN FLEXBOX

y = main ax
x = cross ax

flexbox = ( ) or ( ) → main / cross ax dep's on which

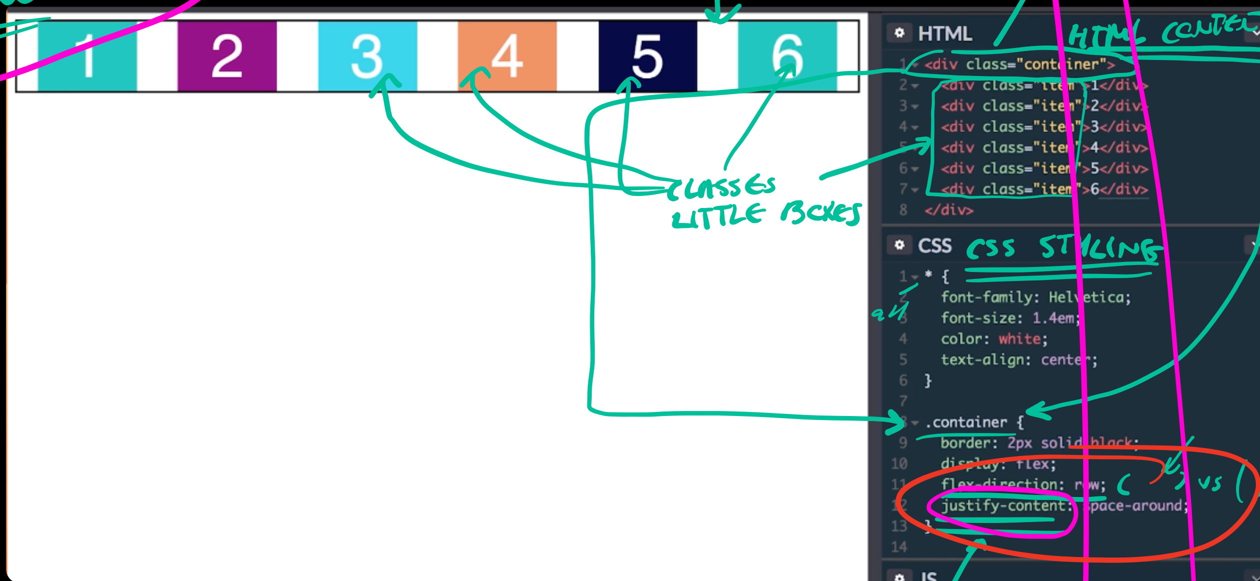
FLEXBOX → the BOX = (O O O) OR ( )

x = main ax
y = cross ax

this is the box

→ el's in it are the boxes
→ in HTML, the box is a class defined in a div wrapping around the entire thing

WHEN THE BIG BOX IS A FLEX BOX WHICH JUST FITS AROUND THE LITTLE BOXES



BIG BOX

HTML CONTENT

CLASSES LITTLE BOXES

CSS STYLING

all

( ) vs ( )

HOW THE LITTLE BOXES MASH IN TO FIT THE BIG BOX, AKA "JUSTIFY"

default
flex-start;

MOVING THEM ALONG THE 2C: MAIN AX.

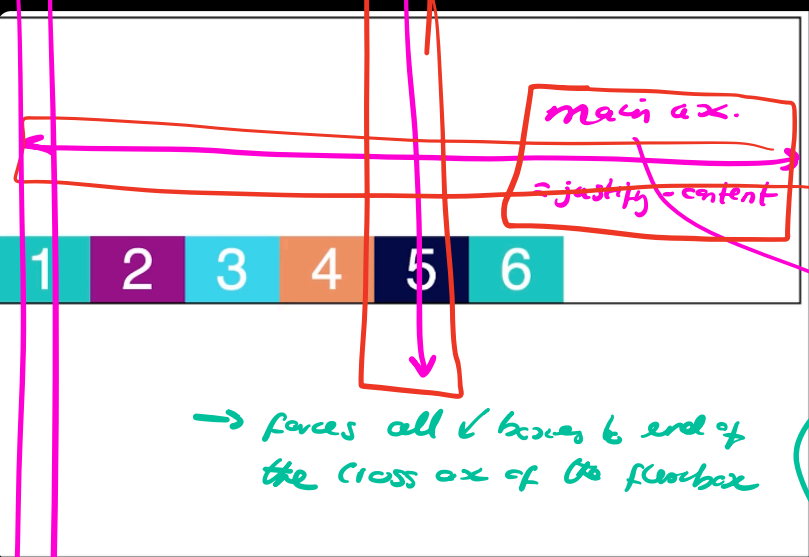
flex-end;

JUSTIFY-CONTENT; ...;

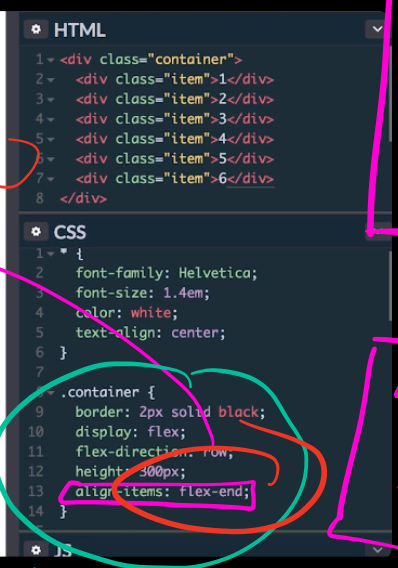
flex-center; ← not centre; (es)

space-around;

cross axe = align-items



→ forces all ↙ boxes to end of the cross ax of the flexbox



in CSS for ↑ box

MOVING THEM ALONG THE Y AX (CROSS AX)

ALIGN-ITEMS: ....;

→ websites = ↑↑ vects

FLEXBOX THOUGHT PROCESS:

- HTML content → ↑ box
  marking it up → ↓ boxes (1D-d)
  → ↓ boxes in ↑ box
- CSS styling → ↑ box

↓ boxes are contained in

-

```
border: 2px solid black;
display: flex;
flex-direction: row;
height: 300px;
align-items: center;
```

( ) or ( ) for ↓ boxes in it

( ) → main axis.

cross axis.

→ makes all ↓ boxes in it into a

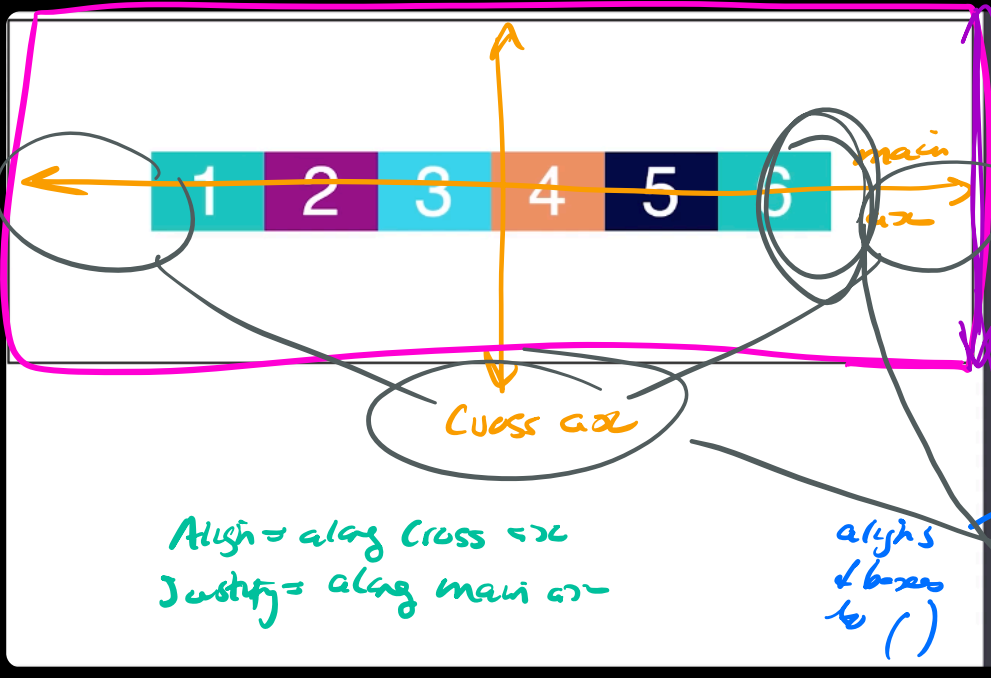
for moving / sticking those boxes around in the Container (↑ box around it in HTML)

→ justify-content is for main axis, align-items is for the cross axis.

→ in this case it's the y → it's centering

along the y.

COMBINE JUSTIFY (FLEXBOX CTRL ON FLEXBOX MAIN AX) + ALIGN (FLEXBOX CTRL ON FLEXBOX CROSS AX)



HTML

```
<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 class="item">6</div>
</div>
```

HTML main ik

CSS

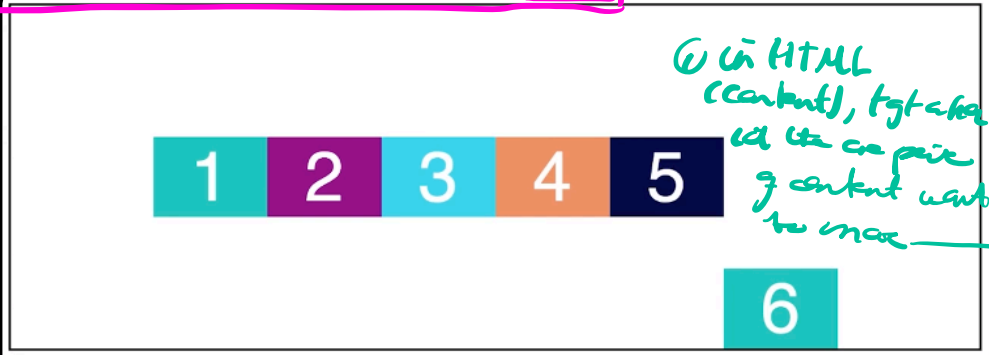
```
font-family: Helvetica;
font-size: 1.4em;
color: white;
text-align: center;
}

.container {
  border: 2px solid black;
  display: flex;
  flex-direction: row;
  height: 300px;
  align-items: center;
  justify-content: center;
}
```

CSS fmatly ik (flexbox)

JS

or ( )
(as CSS grid = both)

TO TGT ONE BOX



① in HTML (content), tgt which el the cursor is & content want to move

② in CSS – style the id for that el. (# = id, ... = class)

③ in this case → align-self: ... ; moves it along y : align= for ⊥ main ax (x = main axe here : ⊥ way container = dgrid)

HTML

```
<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 class="item" id="move-me">6</div>
</div>
```

CSS

```
align-items: center;
justify-content: center;
}

.item {
  width: 100px;
  height: 70px;
}

#move-me {
  align-self: flex-end;
}
```

as HTML < > </ > ".

JS

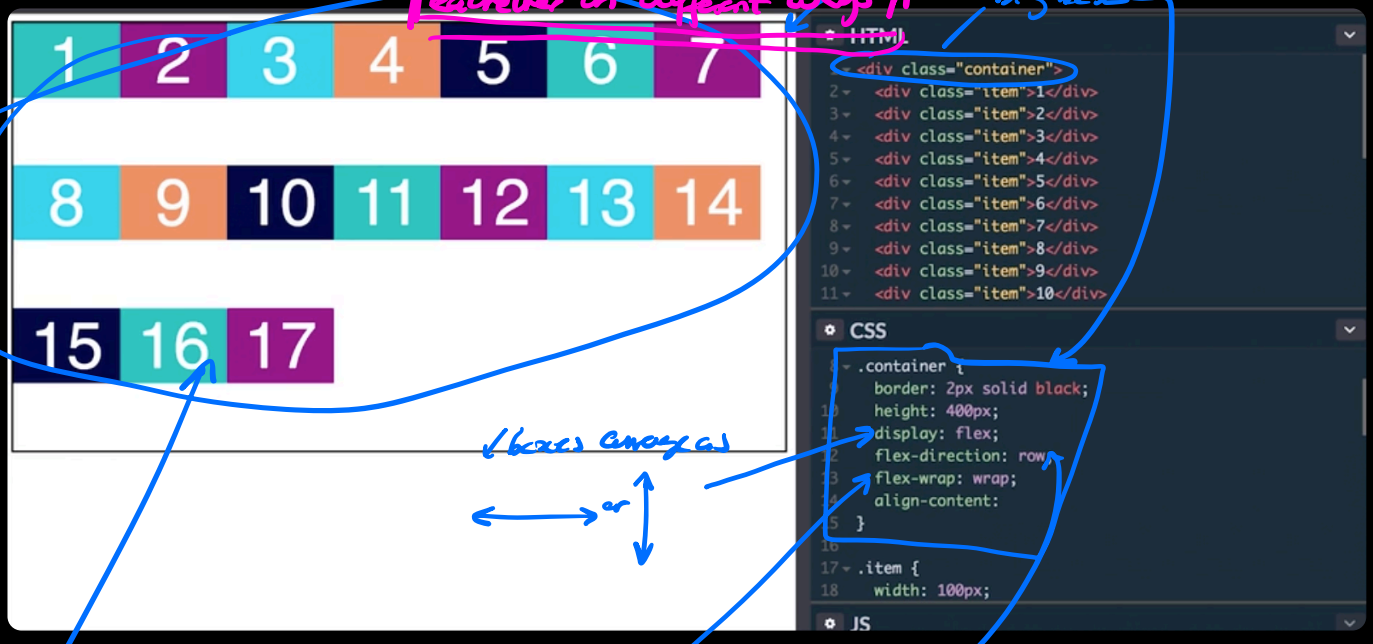
4. Align multiple lines of content

making all of the boxes in box together

→ aligning r's, c's

seperating the boxes apart from each other in different ways

boxes in big box



boxes arranged as

or

boxes arrange as ( )

boxes don't fit onto one row → are .....

this is when how to use align-content

→ to align all of the el's in the box (which is a prop of flexbox)

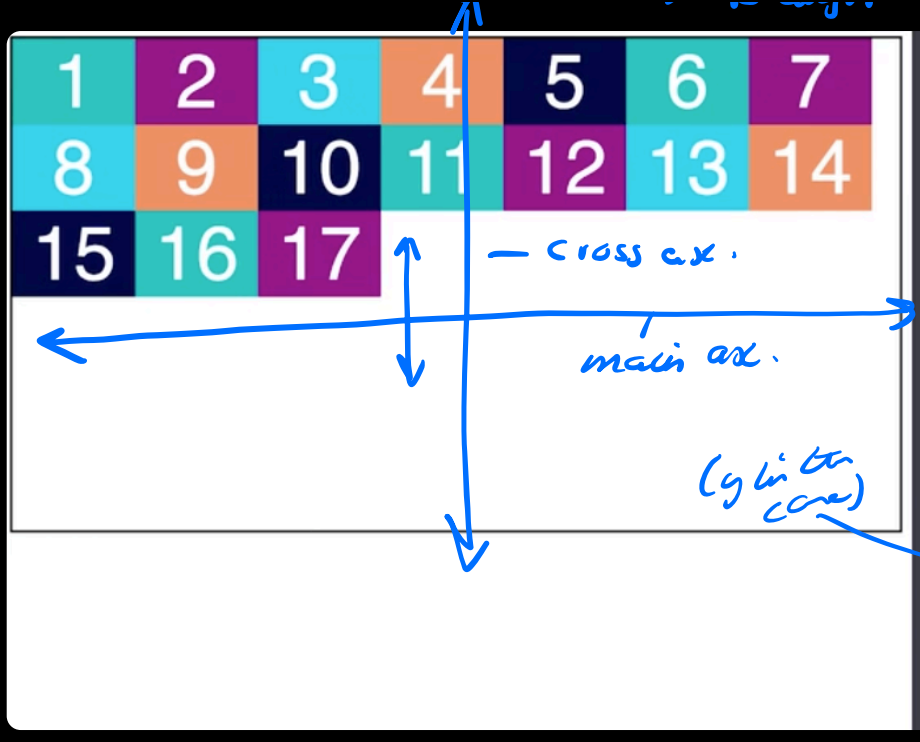
cross axis

→ "align" : . . . ( ) or ( )

main a

"align" is for align the cross axis - justify is main axis

→ to align boxes according to y axe

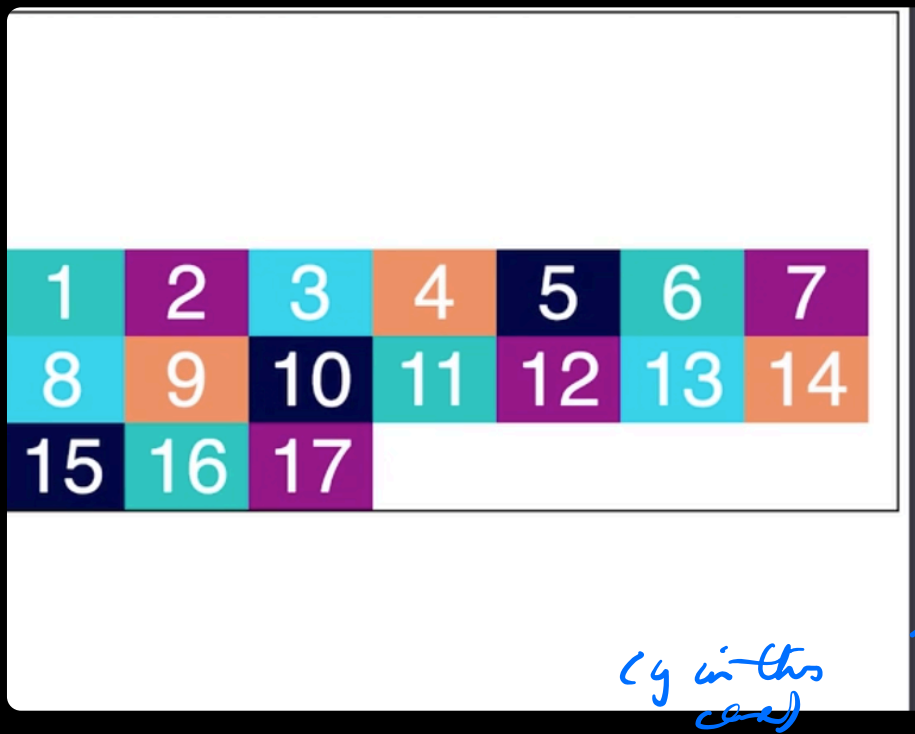


HTML

```
<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 class="item">6</div>
  <div class="item">7</div>
  <div class="item">8</div>
  <div class="item">9</div>
  <div class="item">10</div>
```

CSS

```
.container {
  border: 2px solid black;
  height: 400px;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-content: flex-start;
}

.item {
  width: 100px;
```

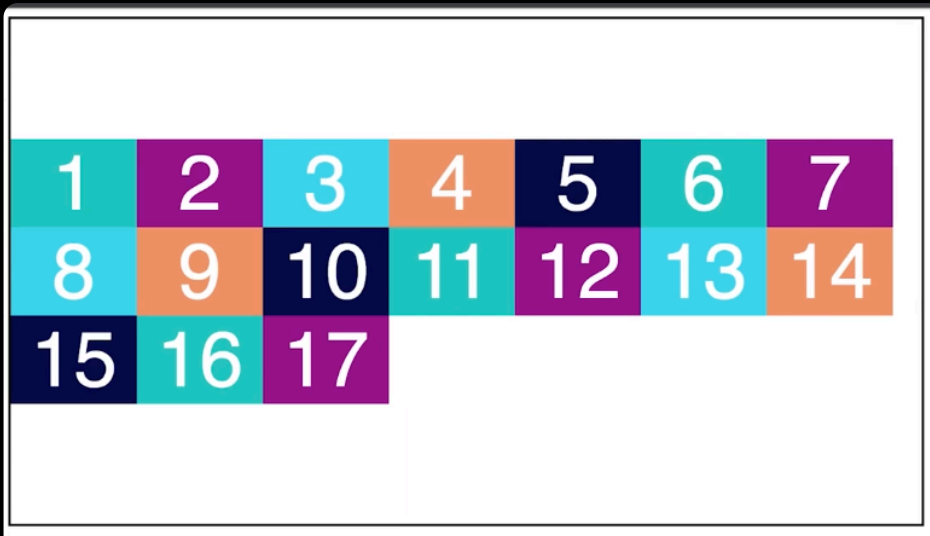


HTML

```
<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 class="item">6</div>
  <div class="item">7</div>
  <div class="item">8</div>
  <div class="item">9</div>
  <div class="item">10</div>
```

CSS

```
.container {
  border: 2px solid black;
  height: 400px;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-content: flex-end;
}

.item {
  width: 100px;
```



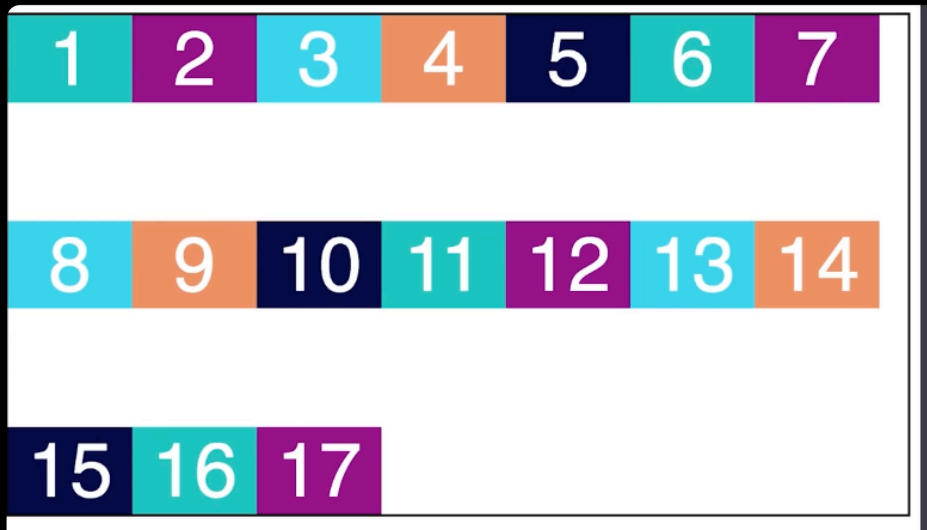
HTML

```
<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 class="item">6</div>
  <div class="item">7</div>
  <div class="item">8</div>
  <div class="item">9</div>
  <div class="item">10</div>
```

CSS

```
.container {
  border: 2px solid black;
  height: 400px;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-content: center;
}

.item {
  width: 100px;
```

JS



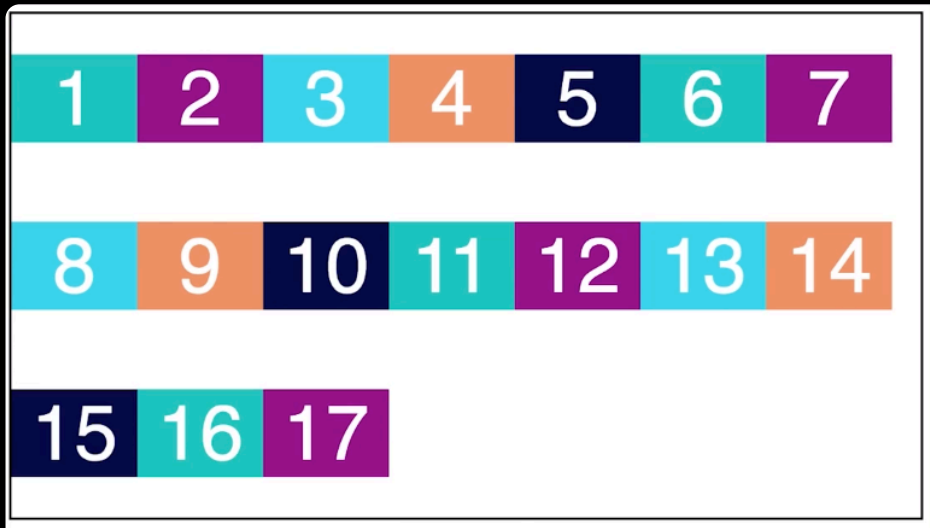
HTML

```
<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 class="item">6</div>
  <div class="item">7</div>
  <div class="item">8</div>
  <div class="item">9</div>
  <div class="item">10</div>
```

CSS

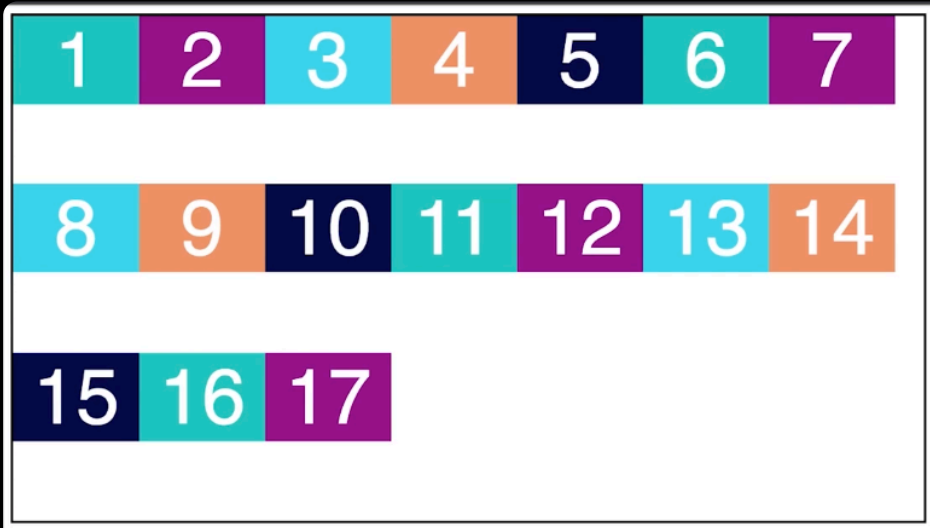
```
.container {
  border: 2px solid black;
  height: 400px;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-content: space-between;
}

.item {
  width: 100px;
```

JS




```html
<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 class="item">6</div>
  <div class="item">7</div>
  <div class="item">8</div>
  <div class="item">9</div>
  <div class="item">10</div>
```

```css
.container {
  border: 2px solid black;
  height: 400px;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  align-content: space-around;
}

.item {
  width: 100px;
```



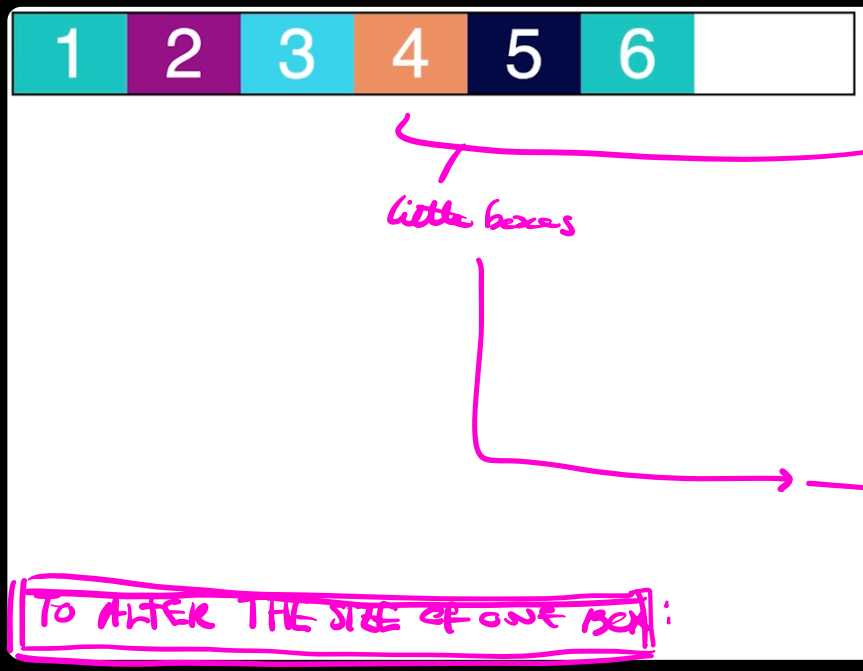

```html
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
  <div class="item">6</div>
  <div class="item">7</div>
  <div class="item">8</div>
  <div class="item">9</div>
  <div class="item">10</div>
```

```css
.container {
  border: 2px solid black;
  height: 400px;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
/*   align-content: stretch; */
}

.item {
  width: 100px;
```

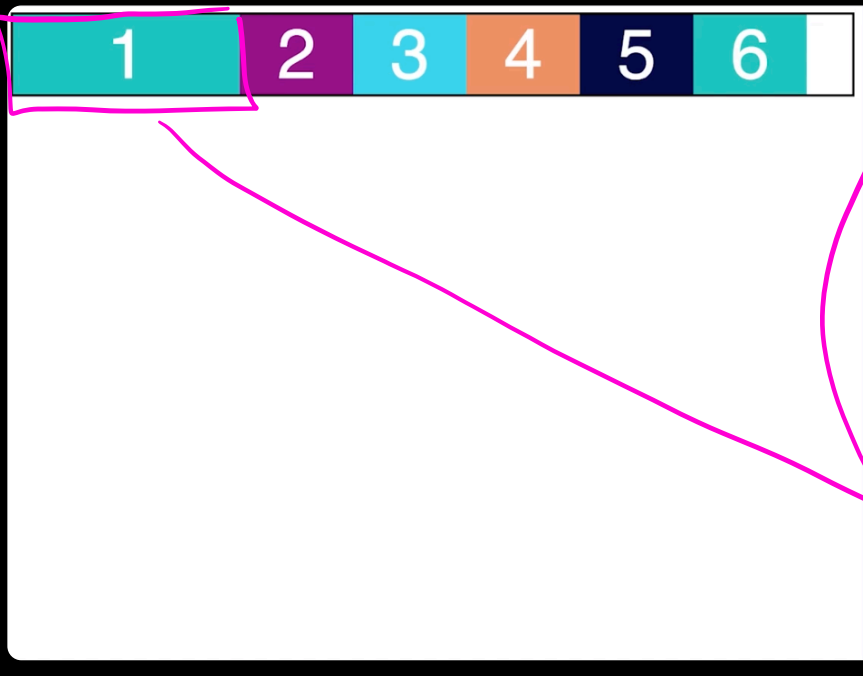
5. Adjust element dimensions

→ when resizing the page → all boxes in 1 flexbox

NORMALLY:



TO ALTER THE SIZE OF ONE BOX:

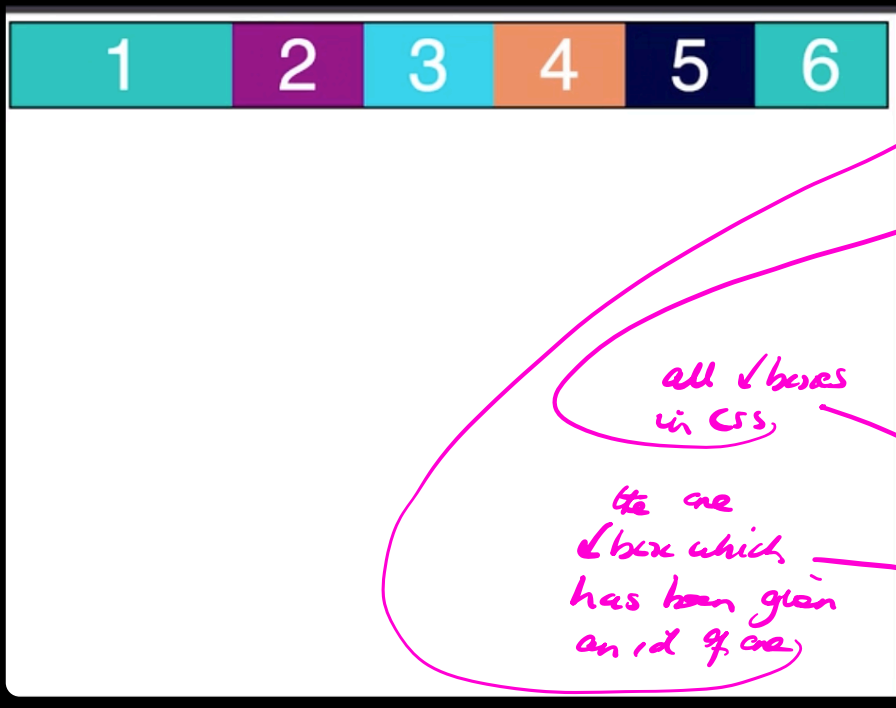


TO A SIZE OF BOXES RELATIVE TO EACH OTHER

Flexbox: flex-basis

A PEN BY Emily Reese

Save | Fork | Settings | Change View



HTML

```
<div class="container">
  <div class="item" id="one">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
  <div class="item">6</div>
</div>
```

- LITTLE BOXES CONTENT = HTML

CSS

```
  border: 2px solid black;
  display: flex;
  flex-direction: row;
}

.item {
  flex-grow: 1;
}

#one {
  flex-grow: 2;
}
```

all ✓ boxes in CSS

the one ✓ box which has been given an id of one

CSS ✓ BOX FORMATTING

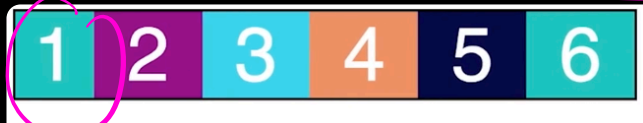
JS

the no's show the ratio's of the sizes of the ✓ boxes relative to eachother

→ and fill out the entire box

→ one box has a flex-grow of 2, and the rest have one of 1 → so the ratio's of their sizes are 2 : the rest

TO CONTROL THE CONTAINER SHRINK RATE



HTML

```
<div class="container">
  <div class="item" id="one">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
  <div class="item">4</div>
  <div class="item">5</div>
  <div class="item">6</div>
</div>
```

CSS

```
  border: 2px solid black;
  display: flex;
  flex-direction: row;
}

.item {
  flex-basis: 100px;
}

#one {
  flex-shrink: 3;
}
```

all containers (✓ boxes) have a default px size of 100px - but when the screen size is shrunk box 1 shrinks 3x faster than the others

JS

TO COMBINE FLEX GROW, SHRINK, BASIS

when the ↑box shrinks → how fast the width of each ↓box shrinks relative to the others

flex: grow shrink basis

when all boxes spread out to full screen width → ratio of the width of that ↓box cf'd to others

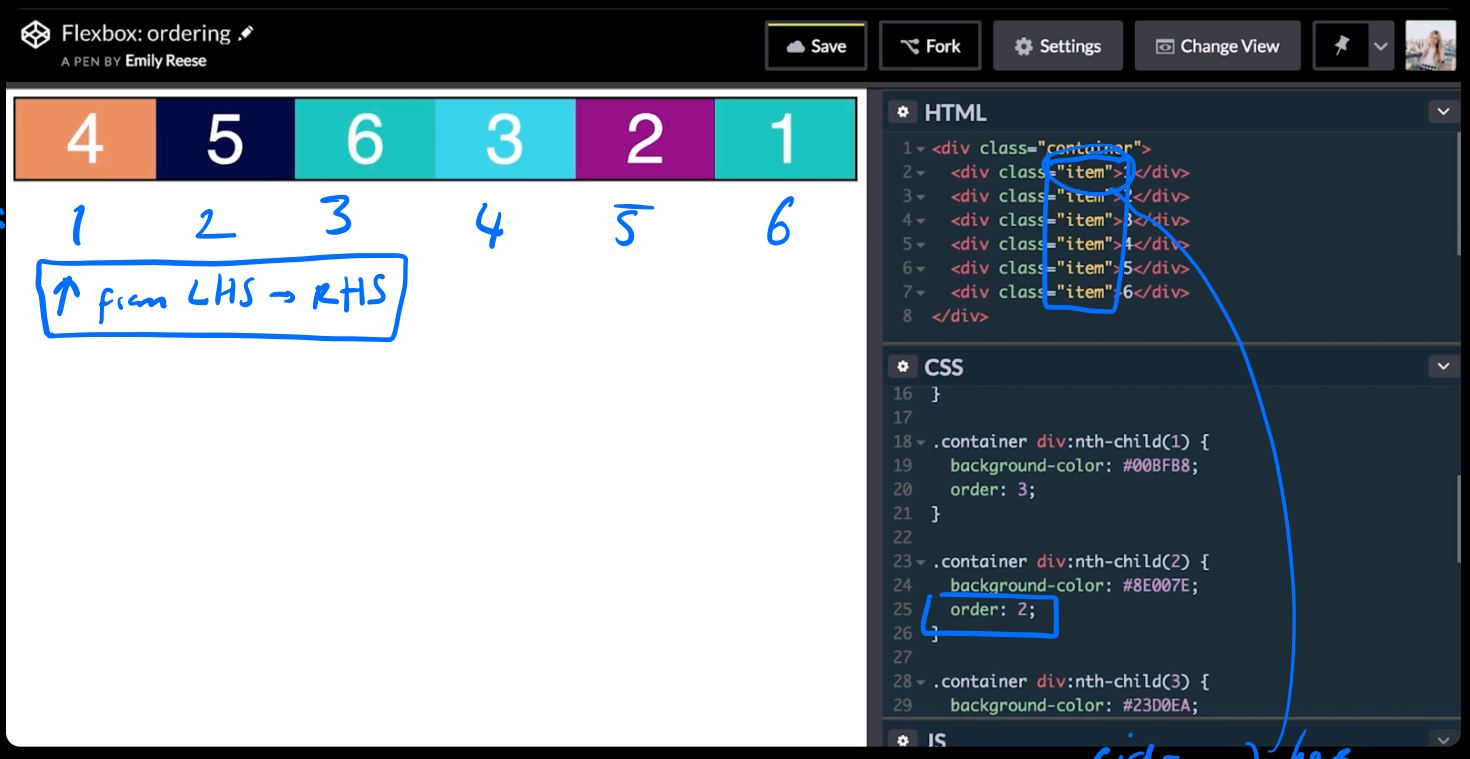
the default width of the ↓box

→ you can do this on 3 different levels of CSS (or in shorthand on one)

6. Reorder elements

→ to Δ the order of the ↓[]·s in CSS (e.g., if ≠ had ch'n over the HTML / was tedious)

TO Δ THE ORDER OF THE ↓BOXES IN THE ↑ BOX IN CSS



→ you can id your ↓boxes in HTML - (id=....) here

→ add id's → then under #'s { } for them in CSS → order: ...;
that ↓ box

→ then all of the ↓ boxes will order according to that in the ↑ box

Quiz point: use wrap-reverse to allow elements to take up multiple lines and display them in reverse

Part #4 - Explore legacy layouts

1. Manually adjust elements' positions
2. Float elements
3. Clear floated elements
4. Stack elements in an order
5. Discover third-party solutions

Quiz: Understand legacy layouts

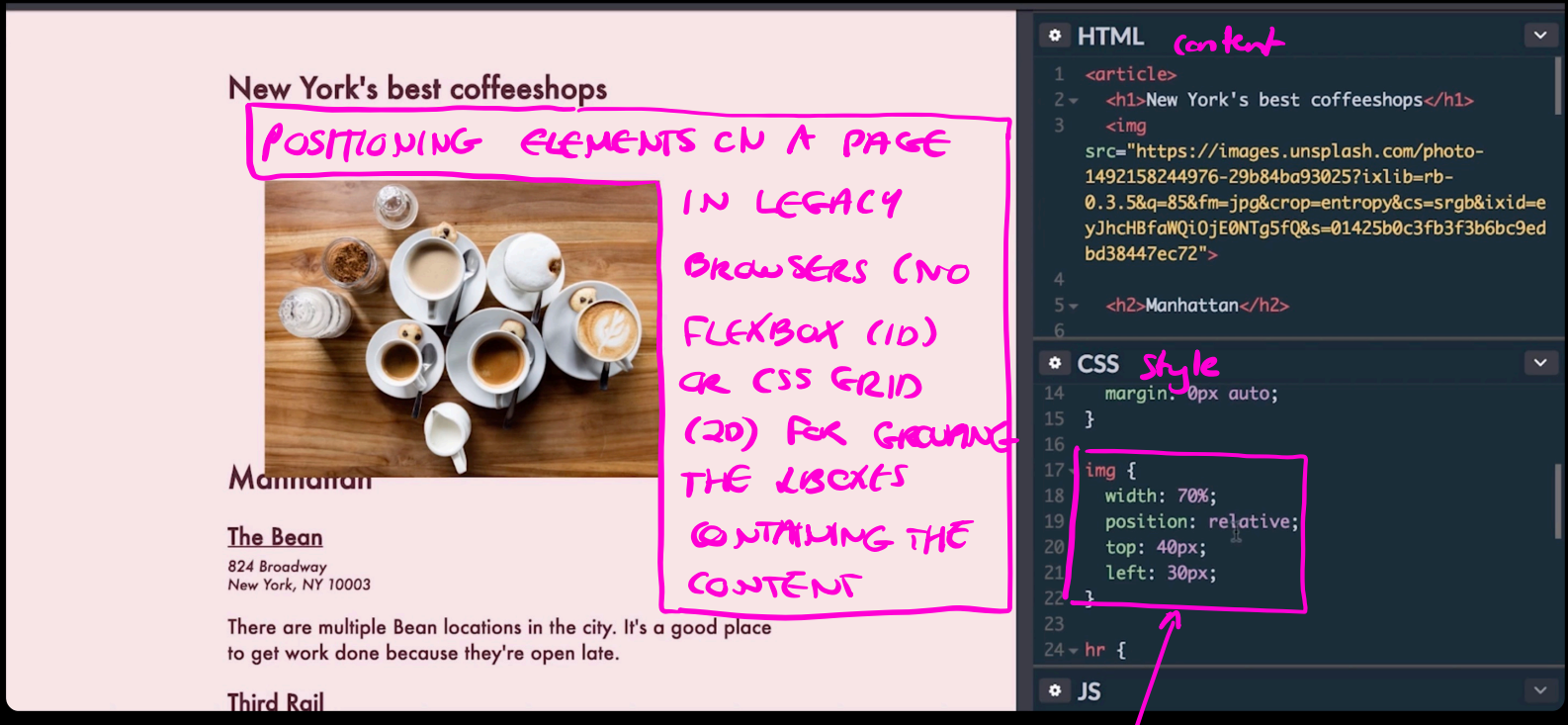
1. Manually adjust elements' positions

- CSS grid / flex box <- 2D / 1D layouts
  - -> **these are new**
  - -> **older code passes used position**
- this is how it worked before flex box
  - -> coffee shops website

→ legacy browsers use cascade of the old :)

Positioning
A PEN BY Emily Reese

Save | Fork | Settings | Change View



METHOD 1:
"RELATIVE TO WHERE IT NORMALLY WOULD BE"

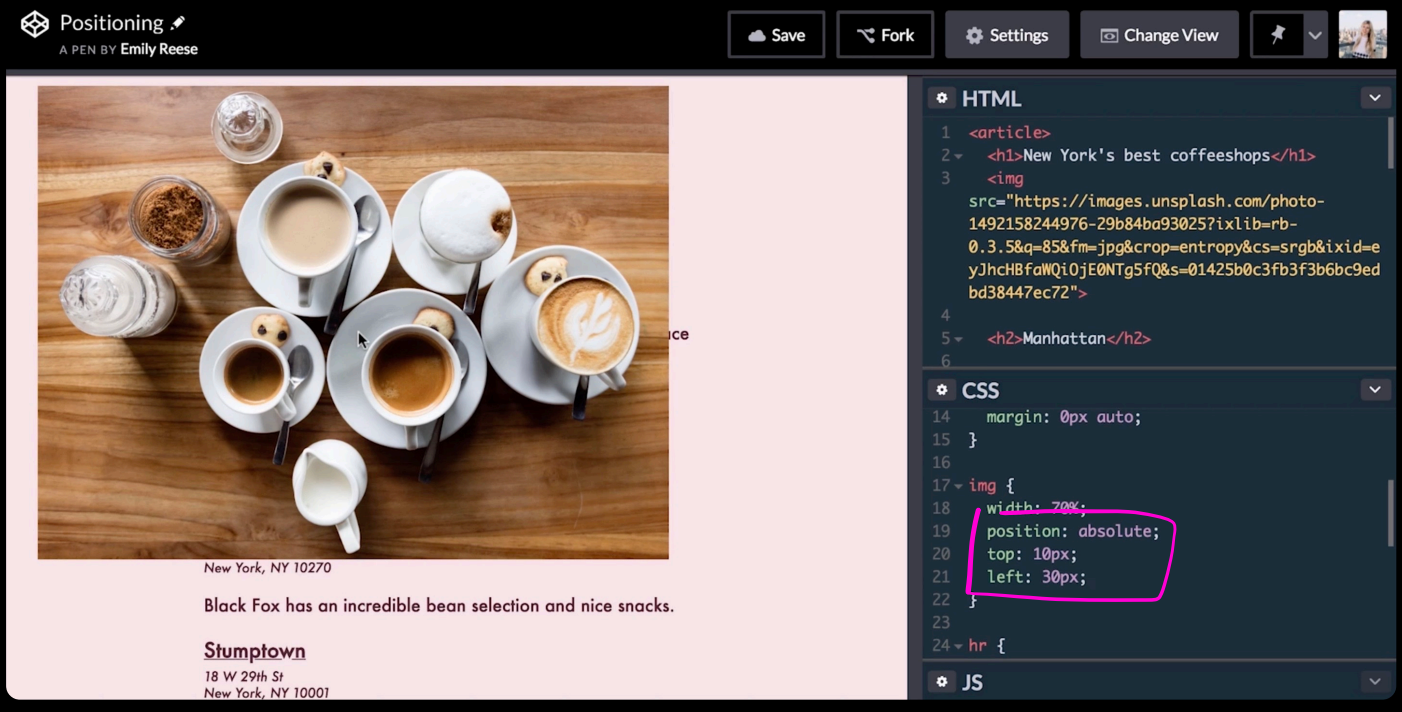
→ default pos. is static

TO TRANSFORM THE POS. OF THE IMAGE

position: relative; ← moves the image relative to where it would normally be

top: ... px;
bottom: ... px; ← it moves left or right (left: ..., right: ...)

METHOD 2:
YOU CAN GIVE IT AN "ABSOLUTE" POSITION ON THE ENTIRE WEBPAGE



→ absolute is relative to the pos of nearest pos'd el on page

POSITION: FIXED; ← TO FIX THE IMAGE AS SOMEONE SCROLLS THROUGH THE WEBPAGE

→ pos. rel to where it would be
→ screen
→ webpage

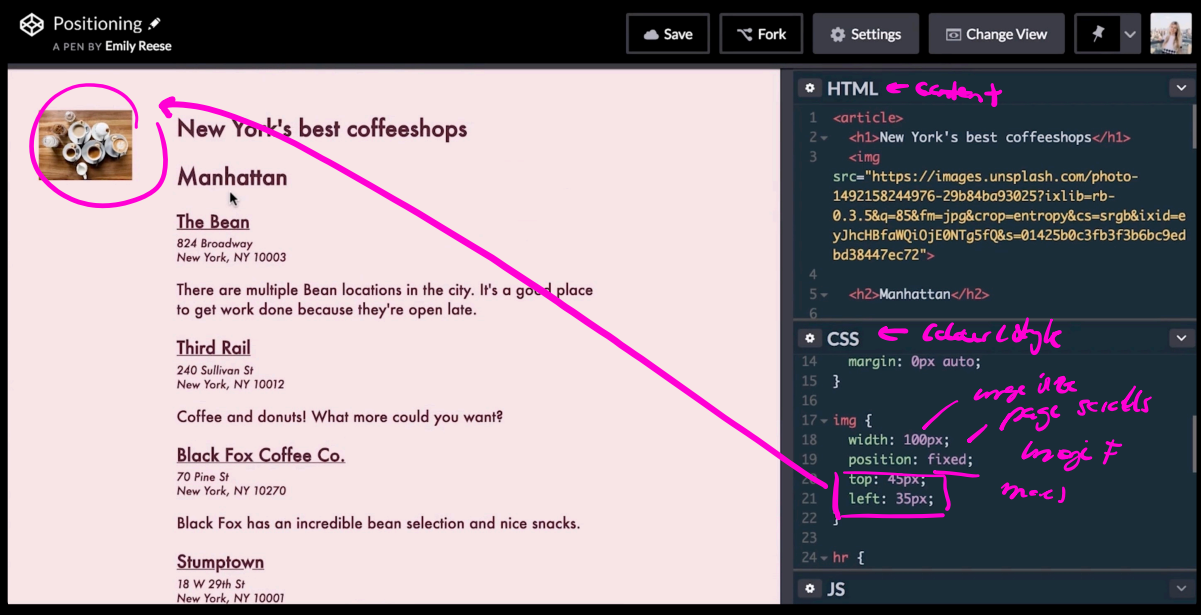


image like page scrolls image ≠ moves

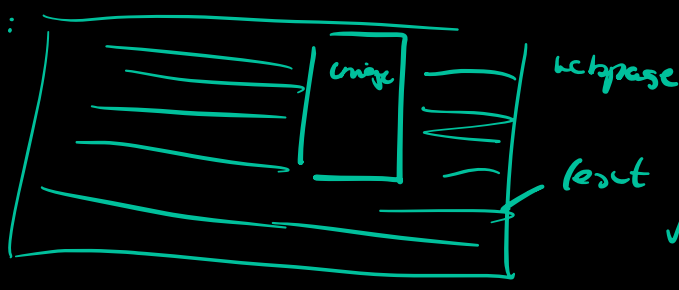
2. Float elements

→ floating text around an image

→ you can use float to float an entire layout

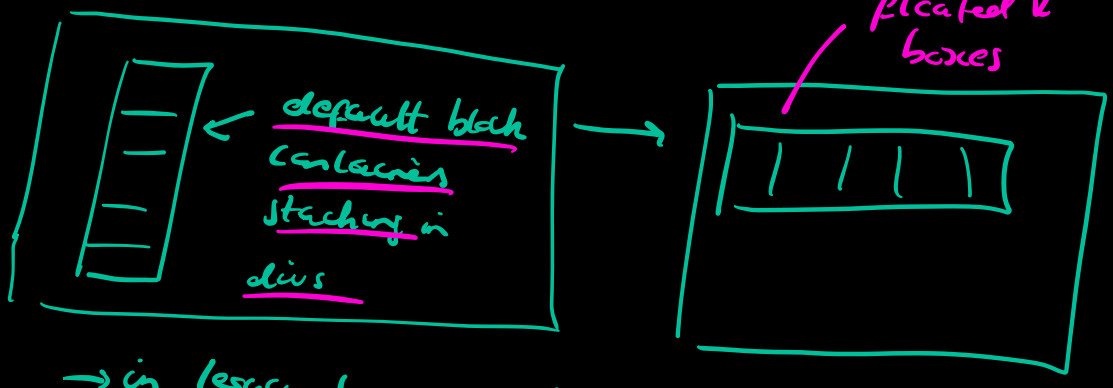
→ divs containing els are block els and normally stack → this is how can float them

EITHER:



floated image & text around it

OR:

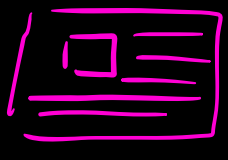


→ in legacy browsers before 10 flexbox stacking wasn't a thing

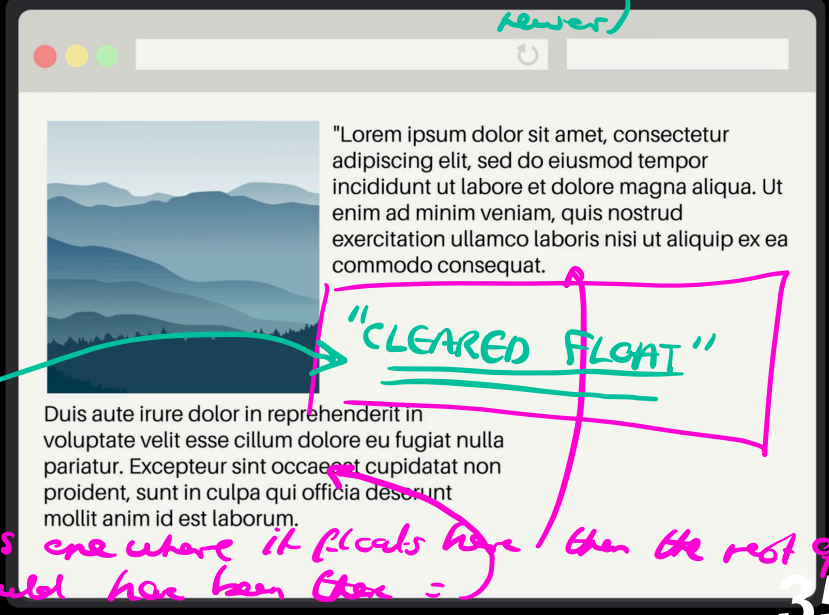
3. Clear floated elements

(for legacy browsers (older → flexbox / CSS grids = newer)

a floated image is one where text



goes around it



→ a cleared float is one where it floats lower than the rest of the text that would have been there =

Clearing floats

A PEN BY Emily Reese

BEFORE // AFTER

Save | Fork | Settings | Change View

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Et netus et malesuada fames ac turpis egestas integer. Duis ut diam quam nulla porttitor massa. Lectus arcu bibendum at varius vel pharetra vel turpis.

Elementum facilisis leo vel fringilla est ullamcorper. Lorem ipsum dolor sit amet consectetur adipiscing elit ut. Leo duis ut diam quam. Ultrices tincidunt arcu non sodales neque. Tortor condimentum lacinia quis vel eros donec ac. Rhoncus urna neque viverra justo. Feugiat in fermentum posuere urna nec tincidunt praesent semper. Urna et pharetra pharetra massa massa ultricies. Sed egestas egestas fringilla phasellus faucibus scelerisque eleifend. Sit amet dictum sit amet justo donec enim diam vulputate. Convallis tellus id interdum velit laoreet id donec. Mi eget mauris pharetra et ultrices. Mattis pellentesque id nibh tortor id. Ac placerat vestibulum lectus mauris ultrices eros. Egestas tellus rutrum tellus pellentesque eu tincidunt tortor aliquam. Nam libero justo laoreet sit amet cursus sit amet dictum.

image

text

HTML CONTENT

```
<article>
  <img
src="https://images.unsplash.com/photo-1470770841072-f978cf4d019e?ixlib=rb-0.3.5&q=85&fm=jpg&crop=entropy&cs=srgb&ixid=eyJhcHBfaWQiOjE0NTg5fQ&s=7196fdeaccc275a9853bf35261c7878e">
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Et netus et
```

CSS STYLE

```
article {
    background-color: #c1e0f4;
    padding: 10px;
}

img {
    width: 200px;
    float: right;
    margin-right: 10px;
    margin-bottom: 10px;
}
```

JS

Clearing floats

A PEN BY Emily Reese

Save | Fork | Settings | Change View

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Et netus et malesuada fames ac turpis egestas integer. Duis ut diam quam nulla porttitor massa. Lectus arcu bibendum at varius vel pharetra vel turpis.

Elementum facilisis leo vel fringilla est ullamcorper. Lorem ipsum dolor sit amet consectetur adipiscing elit ut. Leo duis ut diam quam. Ultrices tincidunt arcu non sodales neque. Tortor condimentum lacinia quis vel eros donec ac. Rhoncus urna neque viverra justo. Feugiat in fermentum posuere urna nec tincidunt praesent semper. Urna et pharetra pharetra massa massa ultricies. Sed egestas egestas fringilla phasellus faucibus scelerisque eleifend. Sit amet dictum sit amet justo donec enim diam vulputate. Convallis tellus id interdum velit laoreet id donec. Mi eget mauris pharetra et ultrices. Mattis pellentesque id nibh tortor id. Ac placerat vestibulum lectus mauris ultrices eros. Egestas tellus rutrum tellus pellentesque eu tincidunt tortor aliquam. Nam libero justo laoreet sit amet cursus sit amet dictum.

HTML (content)

```
porttitor massa. Lectus arcu bibendum at varius vel pharetra vel turpis.</p>
<p id="clear">Elementum facilisis leo vel fringilla est ullamcorper. Lorem ipsum dolor sit amet consectetur adipiscing elit ut. Leo duis ut diam quam. Ultrices tincidunt arcu non sodales neque. Tortor condimentum lacinia quis vel eros donec ac. Rhoncus urna neque viverra justo. Feugiat in fermentum posuere urna nec tincidunt
```

CSS

```
    float: right;
    margin-right: 10px;
    margin-bottom: 20px;
}

#clear {
  clear: both;
}
```

JS

(1) → in HTML (content) mai kaal the para want to clear w/ an id (only for that para)

image

(2) CSS → for styling, coloring etc.

→ clearing...

→ moves the para which is next to it in HTML (clears it of the image)

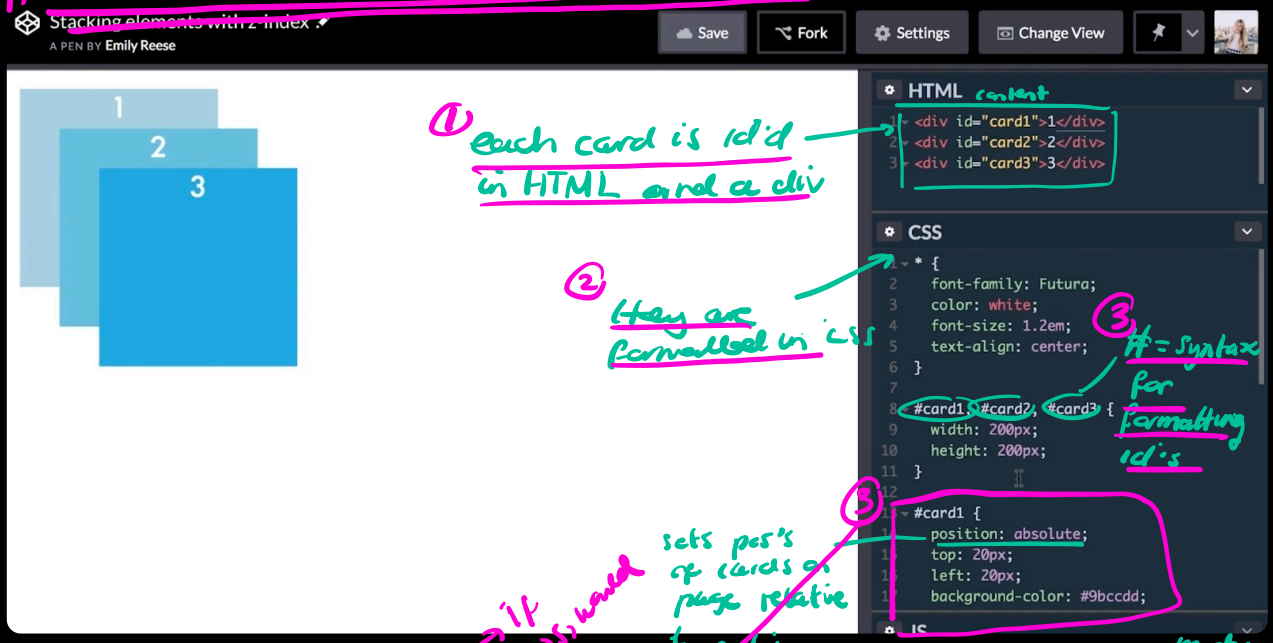
the image is floated, meaning text normally goes b/t it

clear: both ← if image is on RHS/centre/LHS, text moves b/t

: right ← " " " " RHS " " "

4. Stack elements in an order

→ move image to front but in CSS ← z index



① each card is id'd in HTML and as div

② they are formatted in CSS

③ # = syntax for formatting id's

sets pos's of cards on page relative to entire webpage

if it was a class, would be .....

each card has one of these in CSS

each #card1 { .... }

are left'ing that spec. el (if sc=id, for each card)

is formatted in CSS

→ z-index: 3;

to top of pile = ↑ er index

legacy

vs

modern

CSS grid

z = stacking order

x

y

z

5. Discover third-party solutions

→ arranging el's on pages → box mod.

→ height, widths, borders, padding, margins

→ CSS grid / flexbox (= 2D vs 1D)

→ you can use Bootstrap / foundation ← UI or web dev.

- write native CSS to future-proof it → run evolves

Quiz points: Create Web Page Layouts With CSS

```
p {
    position: relative;
    top: 20px;
}
```

- **relative positioning moves an item on a page (a div'd item on HTML is surrounded by a box / container formatted in CSS)**
    - -> on **legacy browsers, this is formatted by position: ....**
    - -> that position can be **relative (to where the box would have otherwise been)**, **absolute (relative to the entire page as a whole (the webpage itself as an entire coordinate system,** rather than a more modern 1D flex box container or 2D CSS grid), or **fixed (the position of the image stays the same on the page as someone scrolls through it**)
        - -> and **when moving items to the front of the webpage -> this is considered to be the z axis (z-indexing)**
            - **imagine the z axis is coming out of the page -> the elements with the biggest z axis index in CSS will be moved to the front**
            - **-> and those elements are div'd and id'd uniquely in HTML, then formatted with the z-index parameter in CSS, using #...**
    - -> the default position of the box containing the element is **static (in CSS)**
- floated elements -> these are removed from the normal flow of the page, with content moving around them
    - **floats are for moving text around images**
    - **-> not for - layering them or bringing them to the front of webpages**
- static and z-indexes
    - -> static positioning -> we're looking at positioning, which is a legacy method of arranging HTML elements which have been labelled / put into div containers, then their positioning is being altered in CSS
    - **-> static positioning is default**
    - **-> in which case -> the boxes won't overlap -> and will stack next to each other (block elements vs inline elements, they will stack like boxes, in line is when they take the line and block is when they stack - the containers containing the content, the not big flex box containers containing the little containers containing the element swhich contain the content)**
    - **-> so when you have non-static positioning (which you set -> relative, absolute etc) -> then we have the possibility of the boxes overlapping**
    - **-> which means -> then we need z-indexing to control how they stack relative to each other on the page**
    - **-> z-indexing isn't used for static elements because**
        - **z indexing is to control which picture / box for instance is moved to the front of the page when we have multiple of them overlapping**
        - -> but static (default) containers don't overlap because they are block level elements (so there is no need to control which goes in front of the other -> using z-indexing, because they are all on the same plane)
        - **-> static elements are fixed and don't overlap -> so there is no need to z-index them to figure out / set which goes in front of the other (they are all on the same plane)**

Question 5

Take the following code:

text

```
h1 {
    width: 500px;
    padding: 5px;
    border: 2px dotted blue;
}
```

Which line of code would make the element's padding and borders be included in its specified width, so that the total width remains 500px?

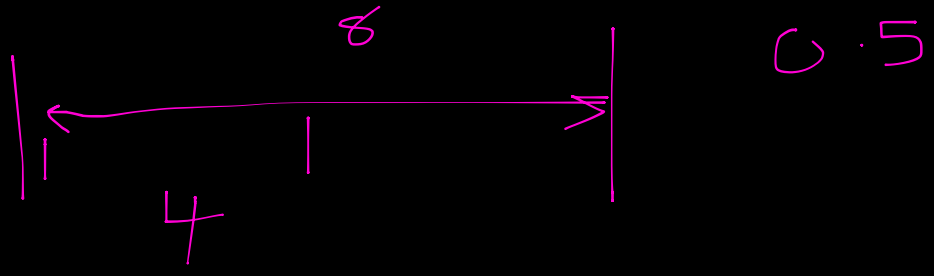
- ○ `box-sizing: border-box;`
- ○ `box-model: box-sizing;`
- ○ `box-border: content-body;`
- ○ `box-sizing: content-box;`

In other words, fit the content to the end of the boarder (not the beginning of the box).

Question 6

If you want an element's width to be 400px by default, and an element's containing block is 800px wide, how should you set its width?

- ○ `width: 10%;`
- ○ `width: 20%;`
- ○ `width: 50%;`
- ◉ `width: 200%;`



Size as a percentage of the size of the box which contains the element

Question 7

You measure an element to find that it has 16px of padding by default, and its font size is 16px. However, its padding value is set in em; not in pixels! What must its padding value be?

- ○ 0.5em
- ○ 1em 
- ○ 2em
- ○ 3em

1em is equal to the current font size