



LAYOUT THE WEB WITH CSS GRID

and the rest of team layout



Original image by the amazing [Lin Clark](#)





This workshop has been inadvertently brought to you by Philippine Airlines.

Surname

First name

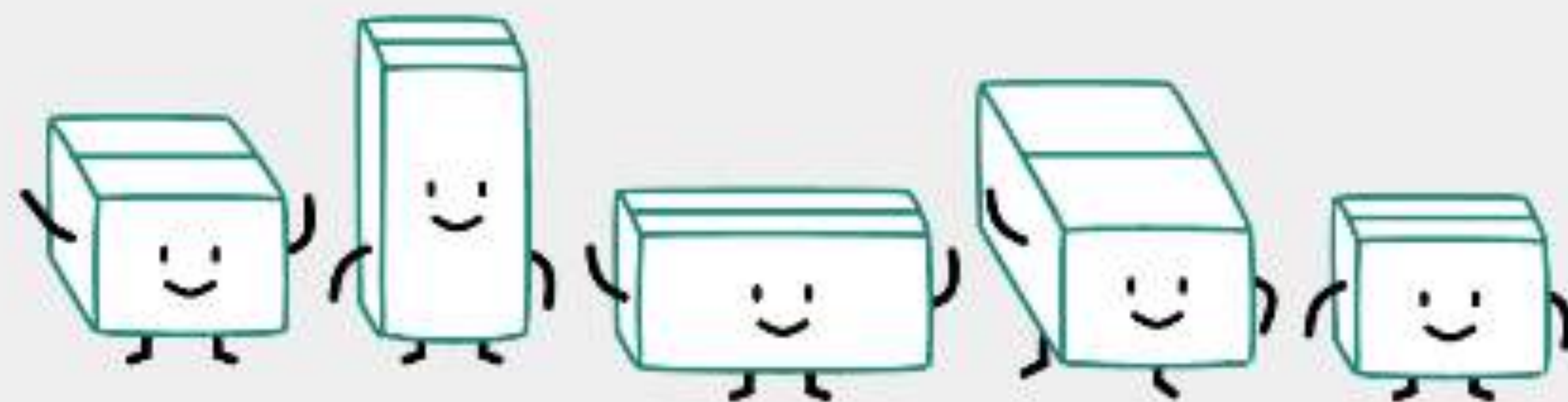
陈	慧	晶
Chen	Hui	Jing

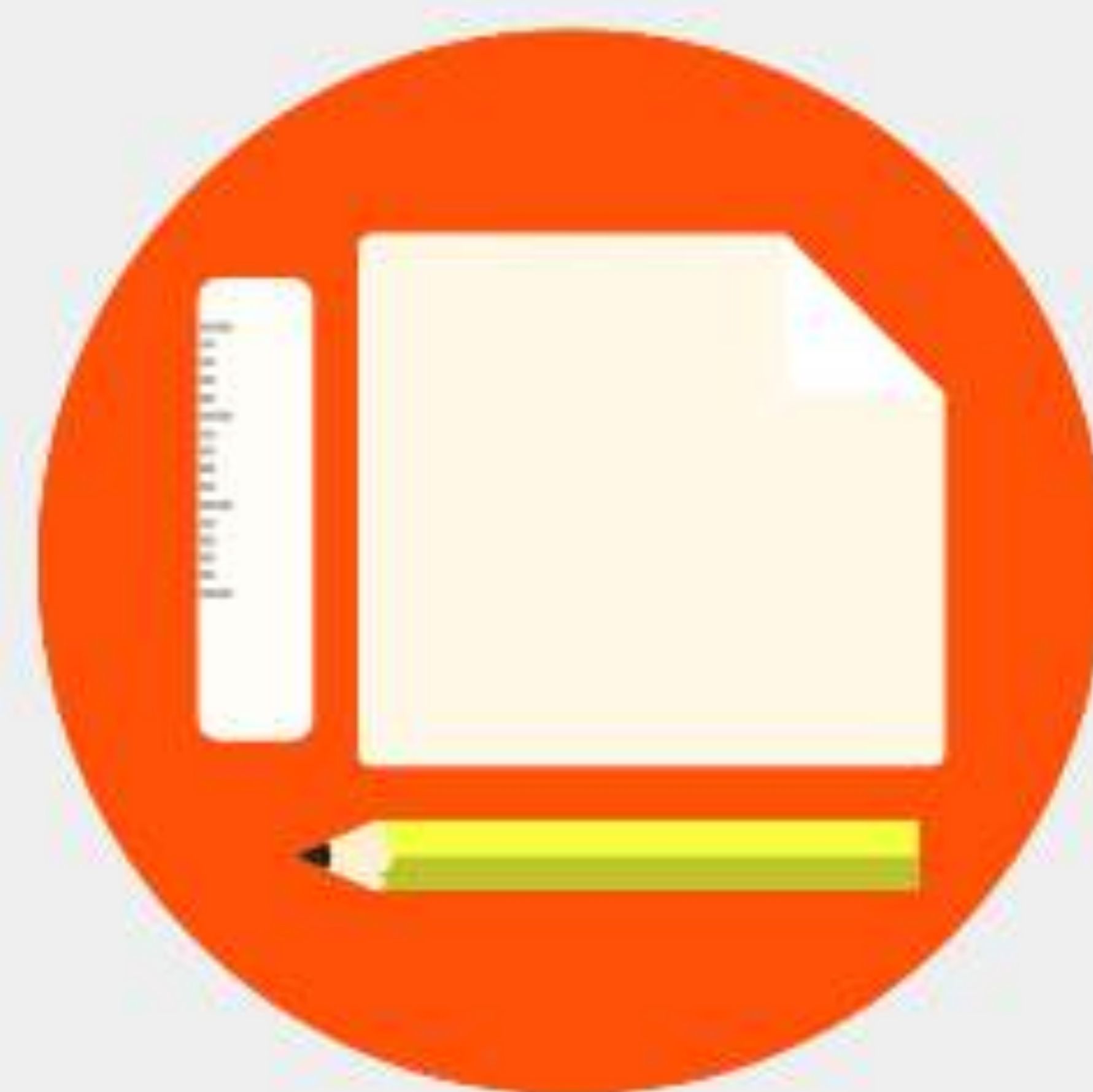
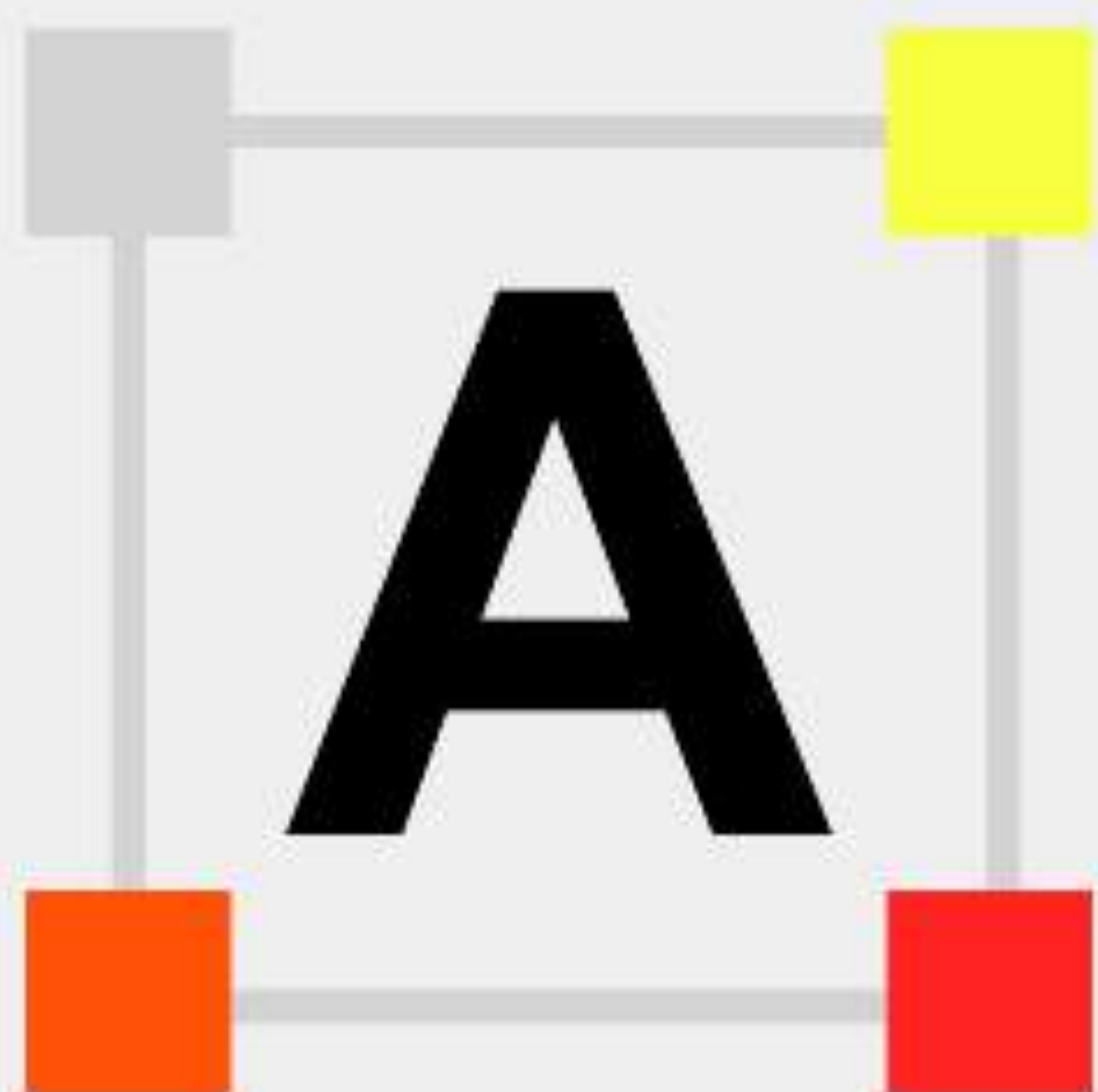


@hj_chen

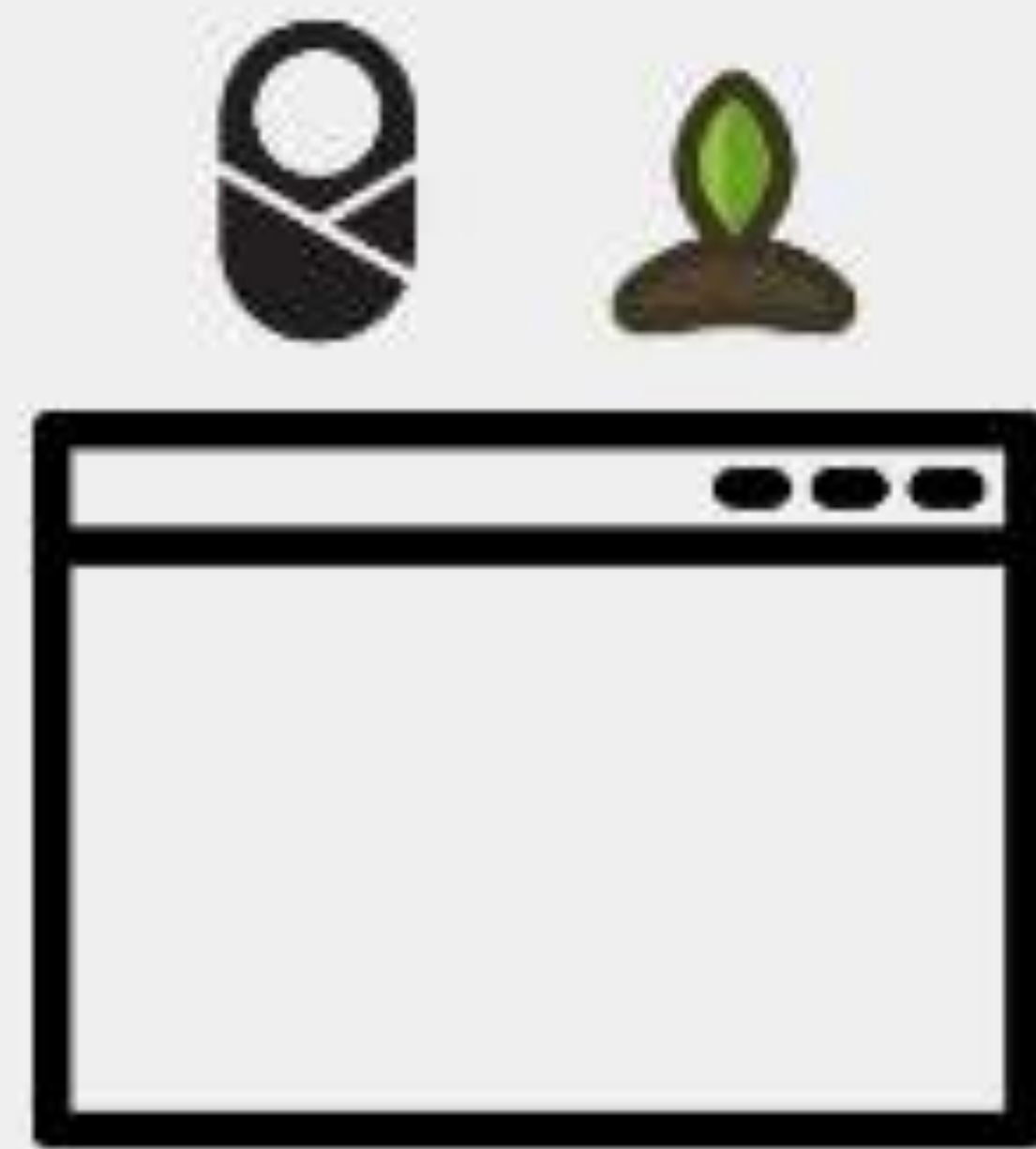
Starter files

<http://bit.ly/ffc9-layout>





Evolution of browsers



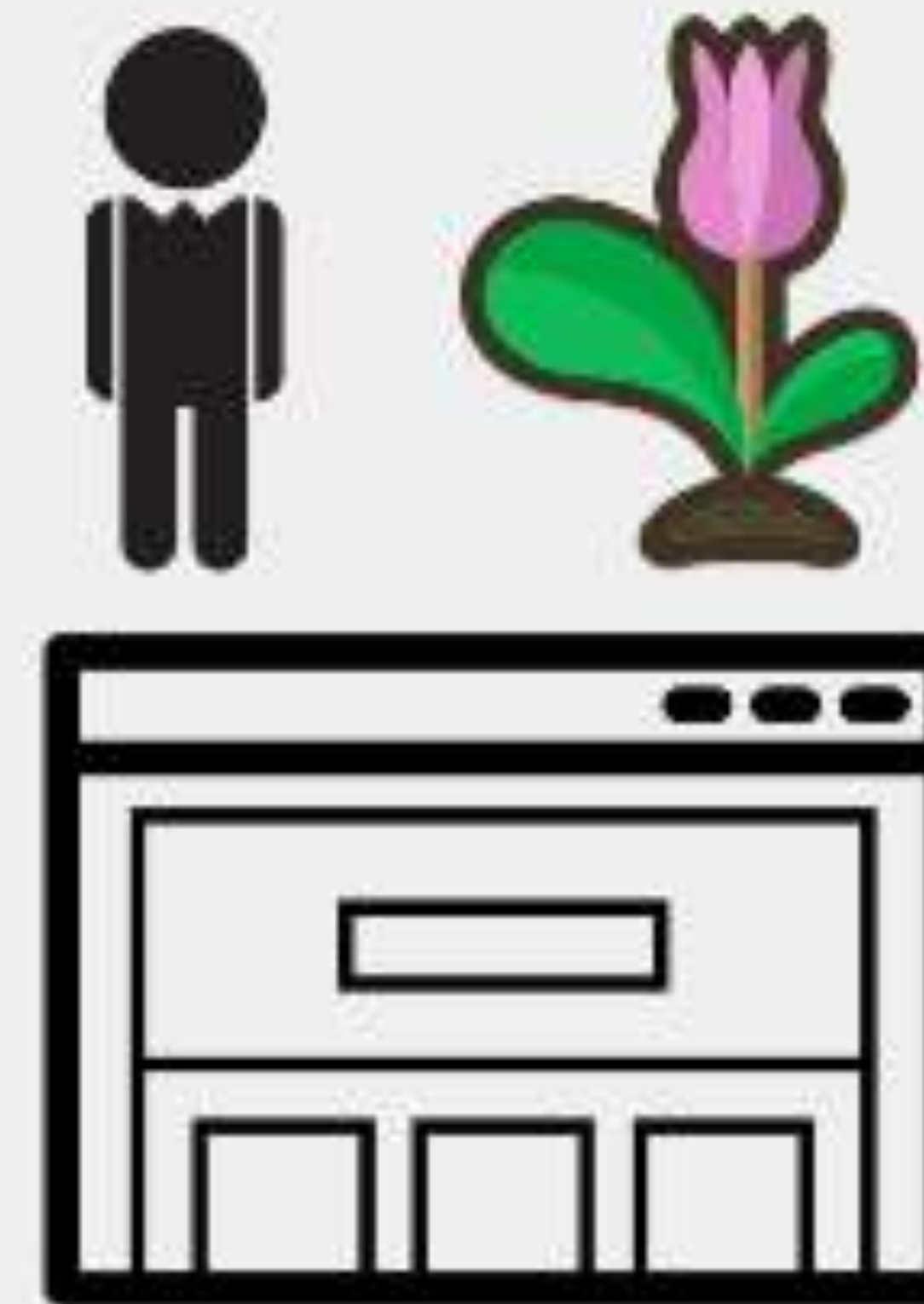
No layout



HTML Tables



CSS Floats

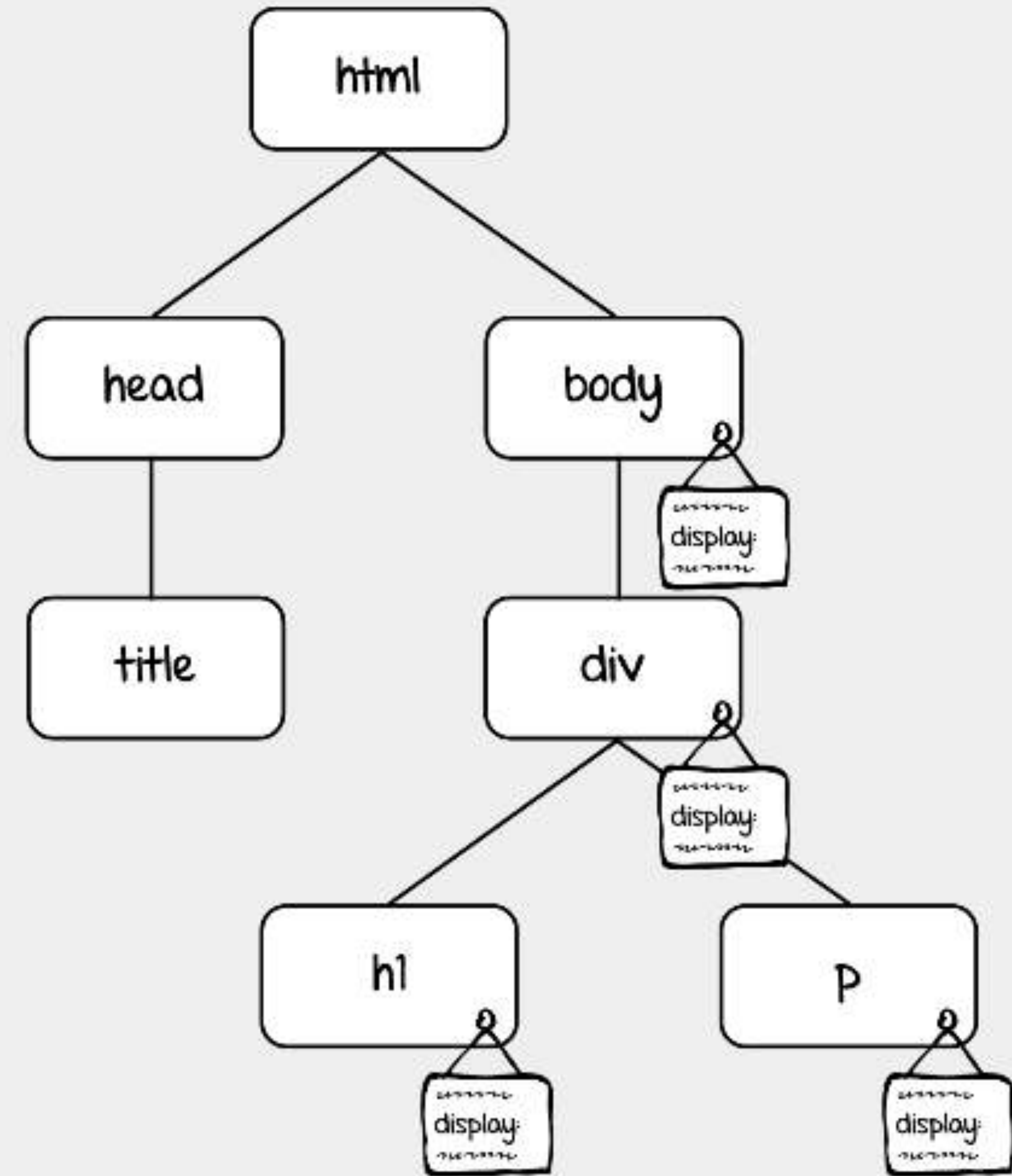


Frameworks



Grid and beyond



```
<!doctype html>
<html lang="en">
  <head>
    <title>CSS rocks</title>
  </head>
  <body>
    <div>
      <h1>Title</h1>
      <p>Flying rabbits, whoa</p>
    </div>
    ...
    ...
  </body>
</html>
```



 Box dimensions and type

 Positioning scheme
(normal flow / float / absolute positioning)

Layout of boxes

 Relationships between
elements in the document tree

 External information
(e.g. viewport size, intrinsic dimensions of images etc.)



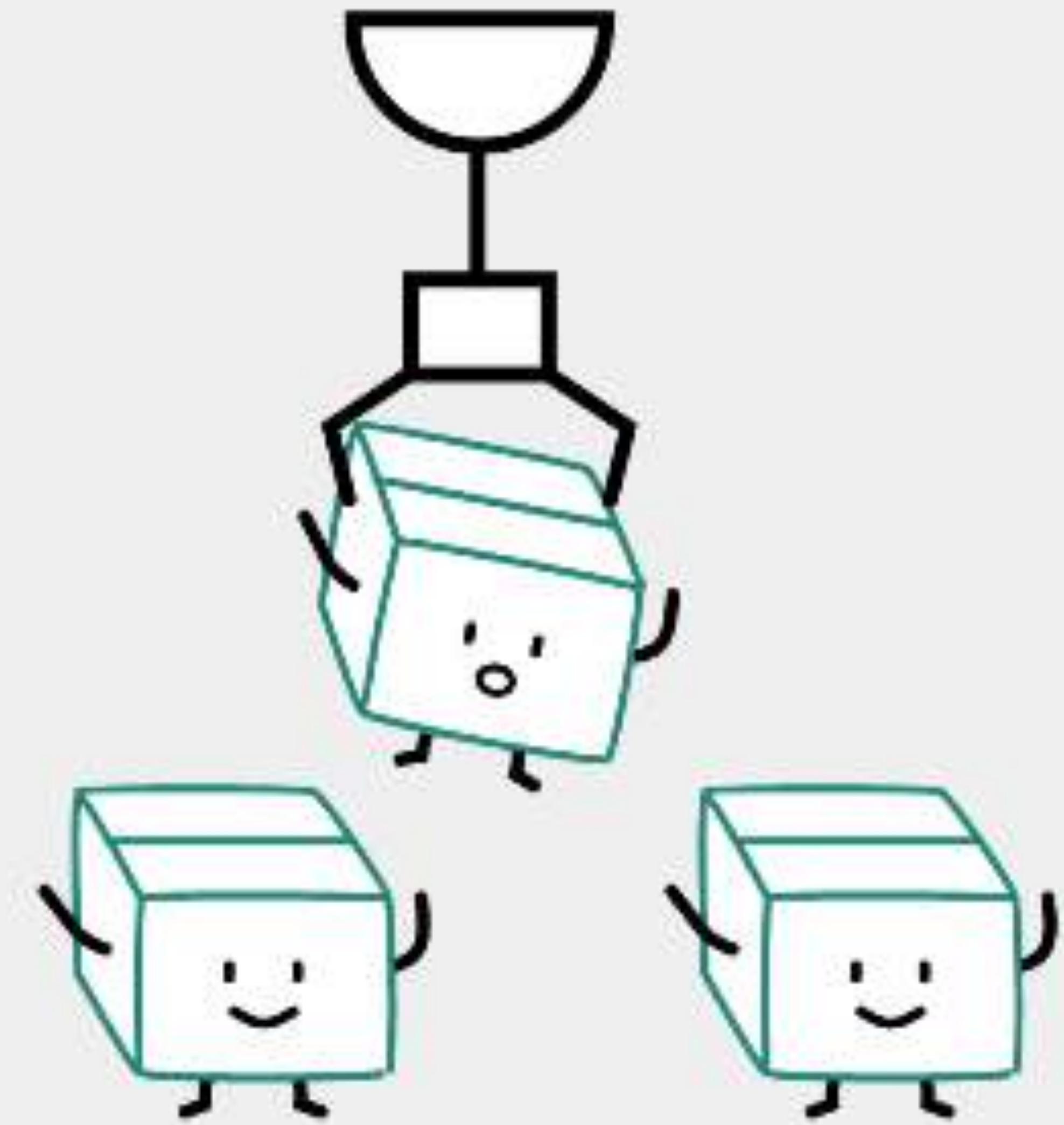
Positioning schemes



Normal flow



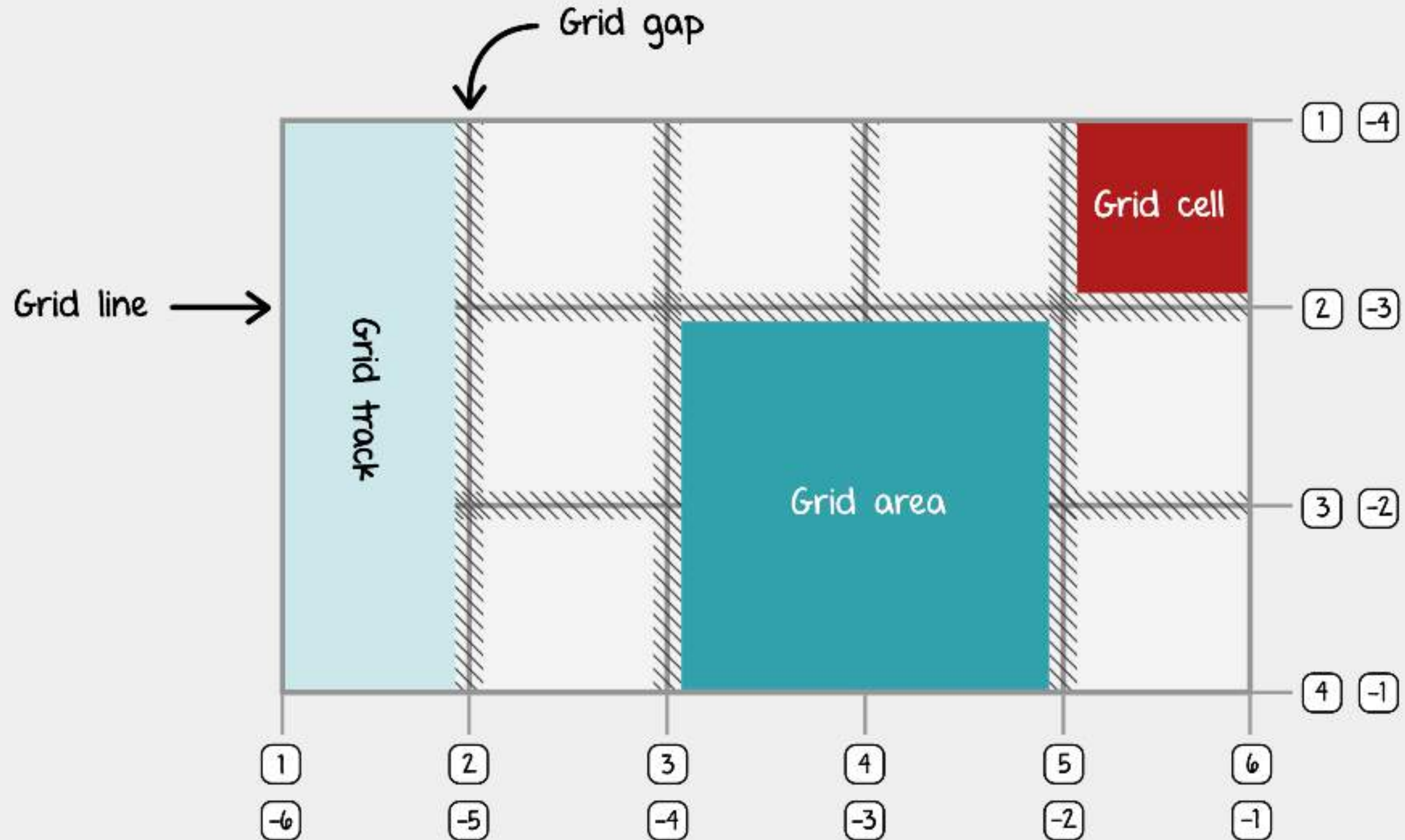
Floats



Absolute positioning

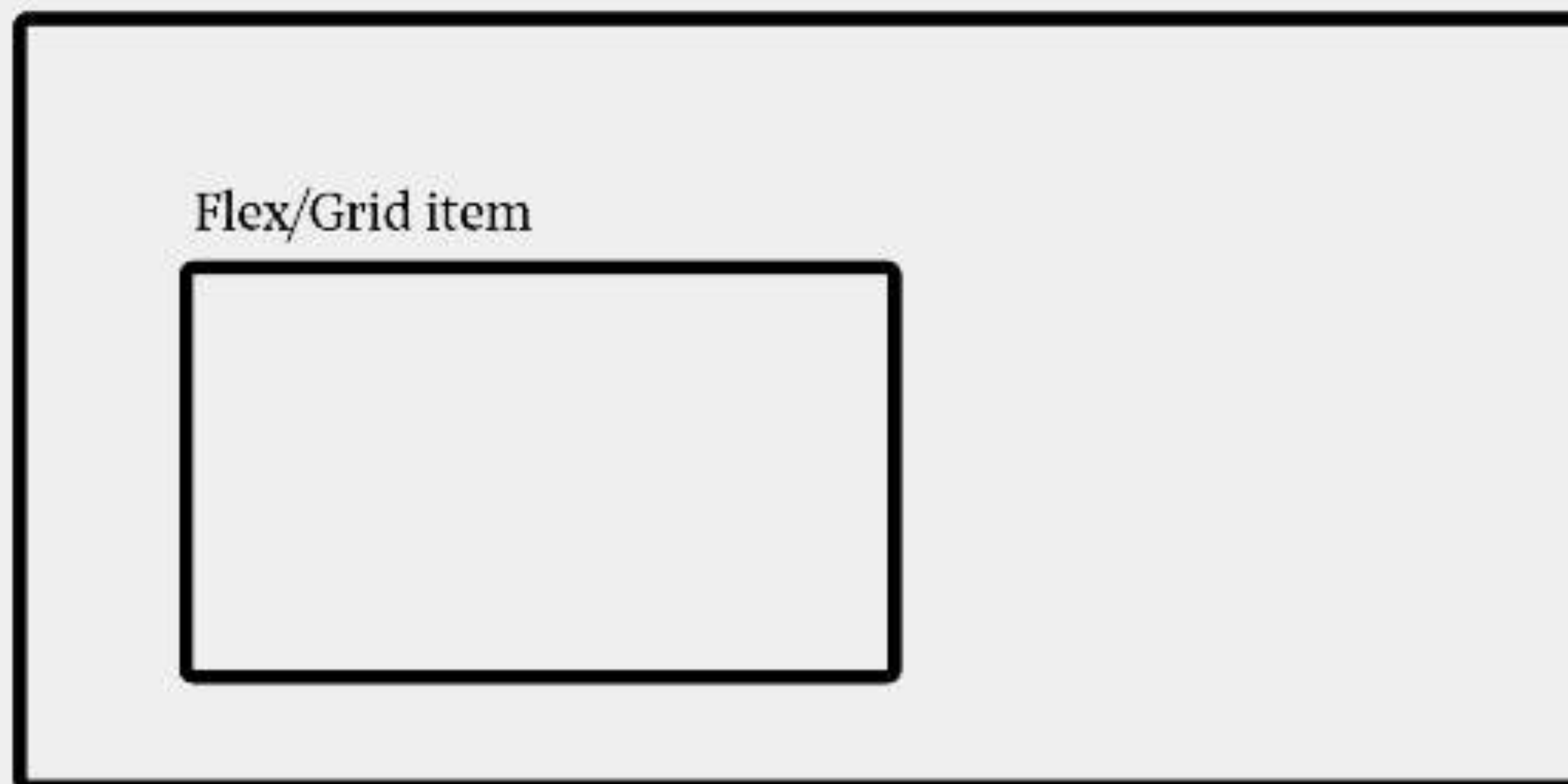


Basic terminology



The container-child relationship

Flex/Grid container



*“Grid works from the **container** in, other layout methods start with the **item**”*

—Rachel Andrew



Layout technique: `inline-block`

Item A	Item B	Item C
Item D	Item E	Item F

```
.inlineblock__item {  
  width: calc(100% / 3);  
  display: inline-block;  
}
```



Layout technique: float

Item A	Item B	Item C
Item D	Item E	Item F

```
.float__item {  
  width: calc(100% / 3);  
  float: left;  
}
```



Layout technique: flex

Item A	Item B	Item C
Item D	Item E	Item F

```
.flexbox {  
  display: flex;  
  flex-wrap: wrap;  
}  
  
.flexbox__item {  
  flex: calc(100% / 3);  
}
```



*“Grid is the only layout technique that establishes a **relationship** between rows and columns of grid items.”*



Firefox Grid Inspector



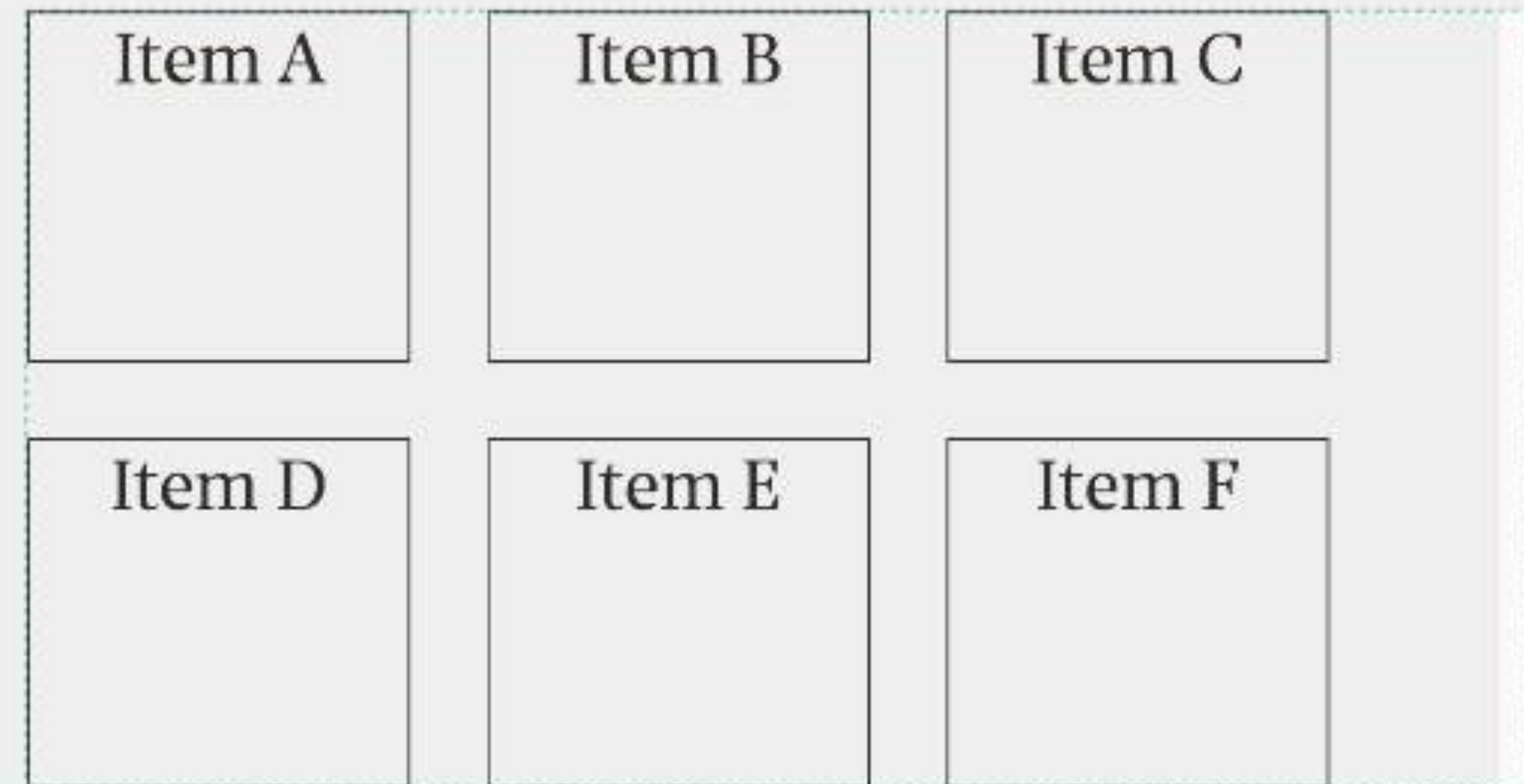
THE MOST BASIC GRID



Defining a grid

Using grid-template-rows and grid-template-columns

```
<div class="grid1">
  <div class="grid1__item">
    <p>Item A</p>
  </div>
  <div class="grid1__item">
    <p>Item B</p>
  </div>
  <div class="grid1__item">
    <p>Item C</p>
  </div>
  <div class="grid1__item">
    <p>Item D</p>
  </div>
  <div class="grid1__item">
    <p>Item E</p>
  </div>
  <div class="grid1__item">
    <p>Item F</p>
  </div>
</div>
```



```
.grid1 {
  display: grid;
  grid-template-columns: 150px 150px 150px;
  grid-gap: 1em;
}
```



AUTO-PLACEMENT OF GRID ITEMS

(and the implicit grid)

The repeat() function

To specify a large number of columns or rows that follow a similar pattern

Item	Item	Item	Item	Item	Item	Item	Item
------	------	------	------	------	------	------	------

```
.grid2 {  
  display: grid;  
  grid-template-columns:  
  repeat(4, 75px 120px);  
}
```



auto-fill versus auto-fit

Letting the browser figure out the math

A	B	C	D	E	F	

```
.keyword {  
  display: grid;  
  grid-template-columns:  
  repeat(auto-fill, minmax(100px,  
  1fr));  
}
```



grid-auto-row and grid-auto-column

Controlling the size of implicit rows/columns

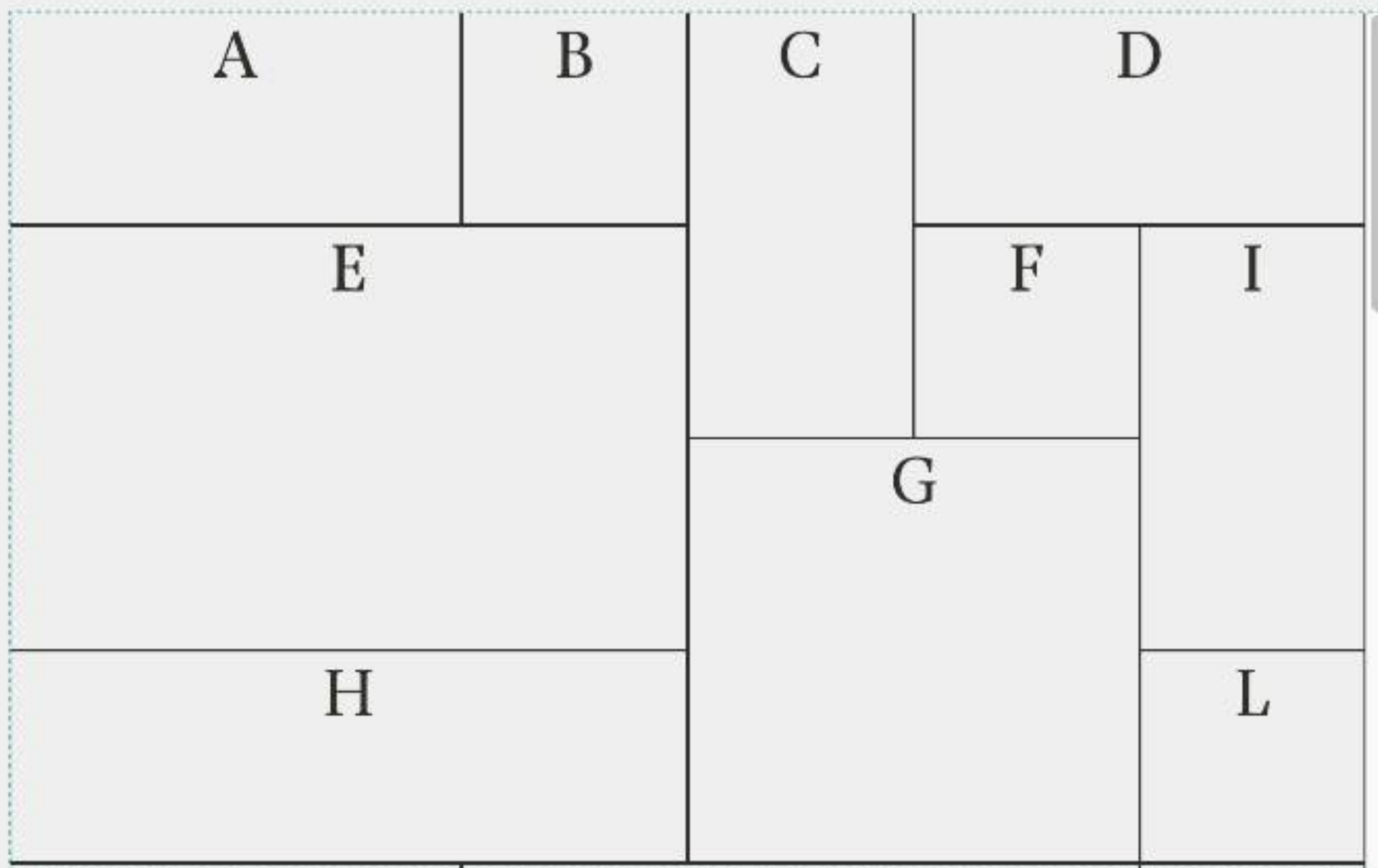
A A	B	C
D	E	F
G	H	I
J	K	L

```
.grid3 {  
  display: grid;  
  grid-template-columns: 150px  
150px 150px;  
  grid-template-rows: 150px;  
  grid-auto-rows: 120px;  
}
```



The `grid-auto-flow` property

Adjusting the direction and density of grid items



```
.grid4 {  
  display: grid;  
  grid-template-columns:  
    repeat(auto-fit, minmax(120px,  
    1fr));  
  grid-auto-rows: 120px;  
  grid-auto-flow: dense row;  
}  
  
.grid4__item:nth-child(3n+1) {  
  grid-column-end: span 2;  
}  
  
.grid4__item:nth-child(2n+3) {  
  grid-row-end: span 2;  
}  
  
.grid4__item:nth-child(3n+5) {  
  grid-column-end: span 3;  
}
```



MORE WAYS TO SIZE GRID ROWS AND COLUMNS

The `fr` unit

Represents a **fraction** of the **free space** in the grid container.

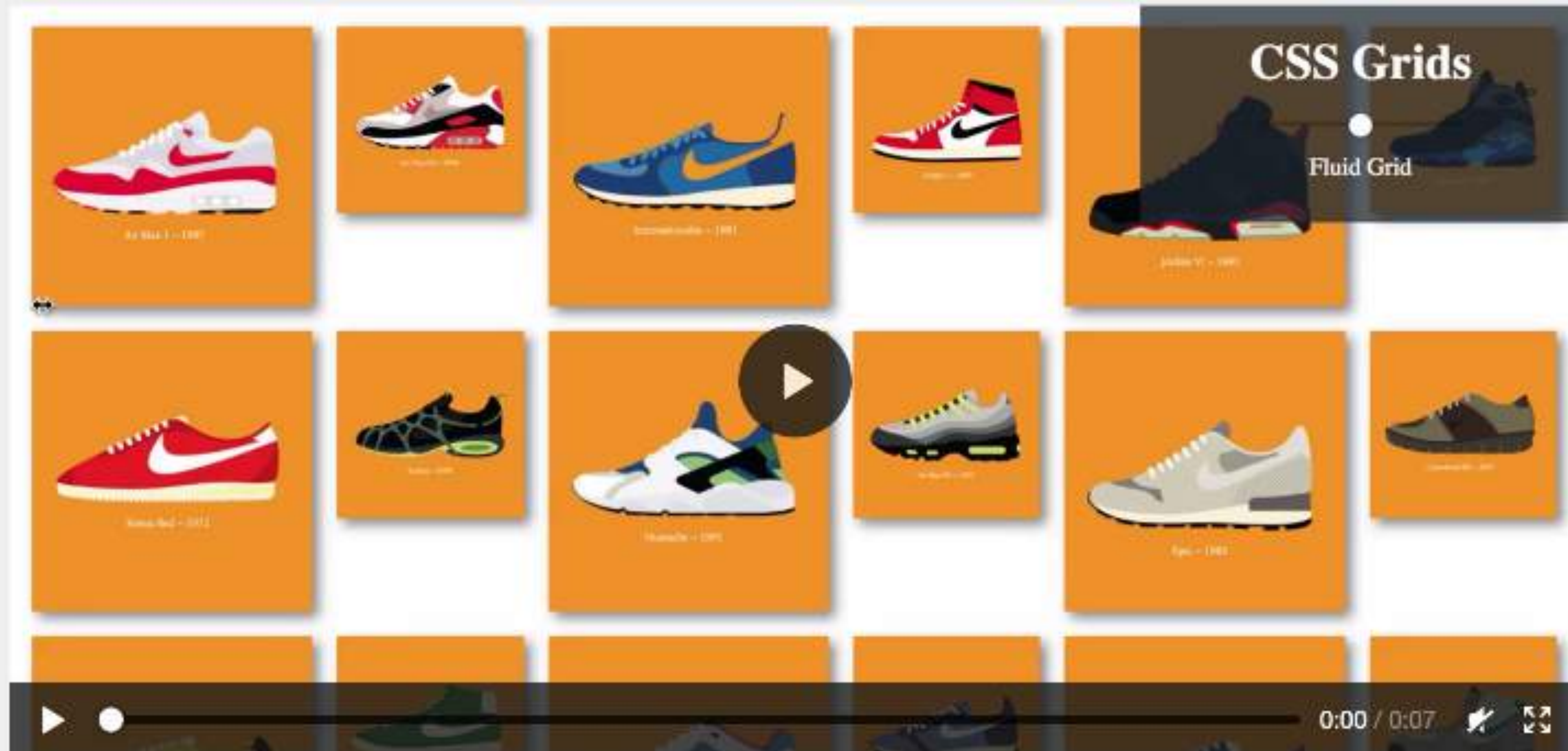
Item A	Item B	Item C
--------	--------	--------

```
.grid5 {  
  display: grid;  
  grid-template-columns: 150px  
  1fr 2fr;  
}
```



Fluid CSS grid

```
.container {  
  display: grid;  
  grid-template-columns: repeat(3, 3fr 2fr);  
}
```



The `minmax()` function

Defines a size range for columns or rows in the grid.

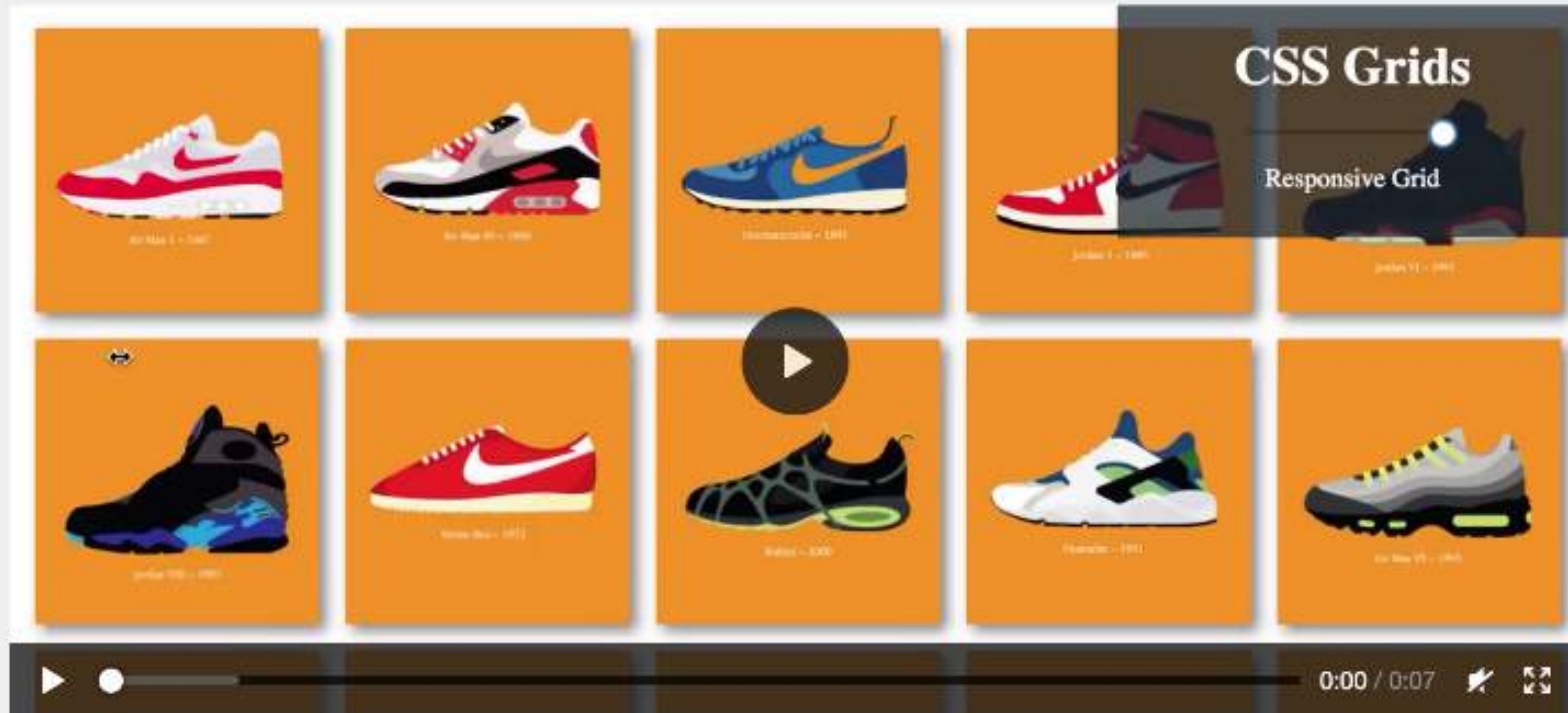
Item A	Item B	Item C
--------	--------	--------

```
.grid6 {  
  display: grid;  
  grid-template-columns:  
  minmax(200px, 1fr) 300px 300px;  
}
```



Responsive grid without media queries

```
.container {  
  display: grid;  
  grid-template-columns: repeat(auto-fill, minmax(10em, 1fr));  
}
```



Content-based sizing

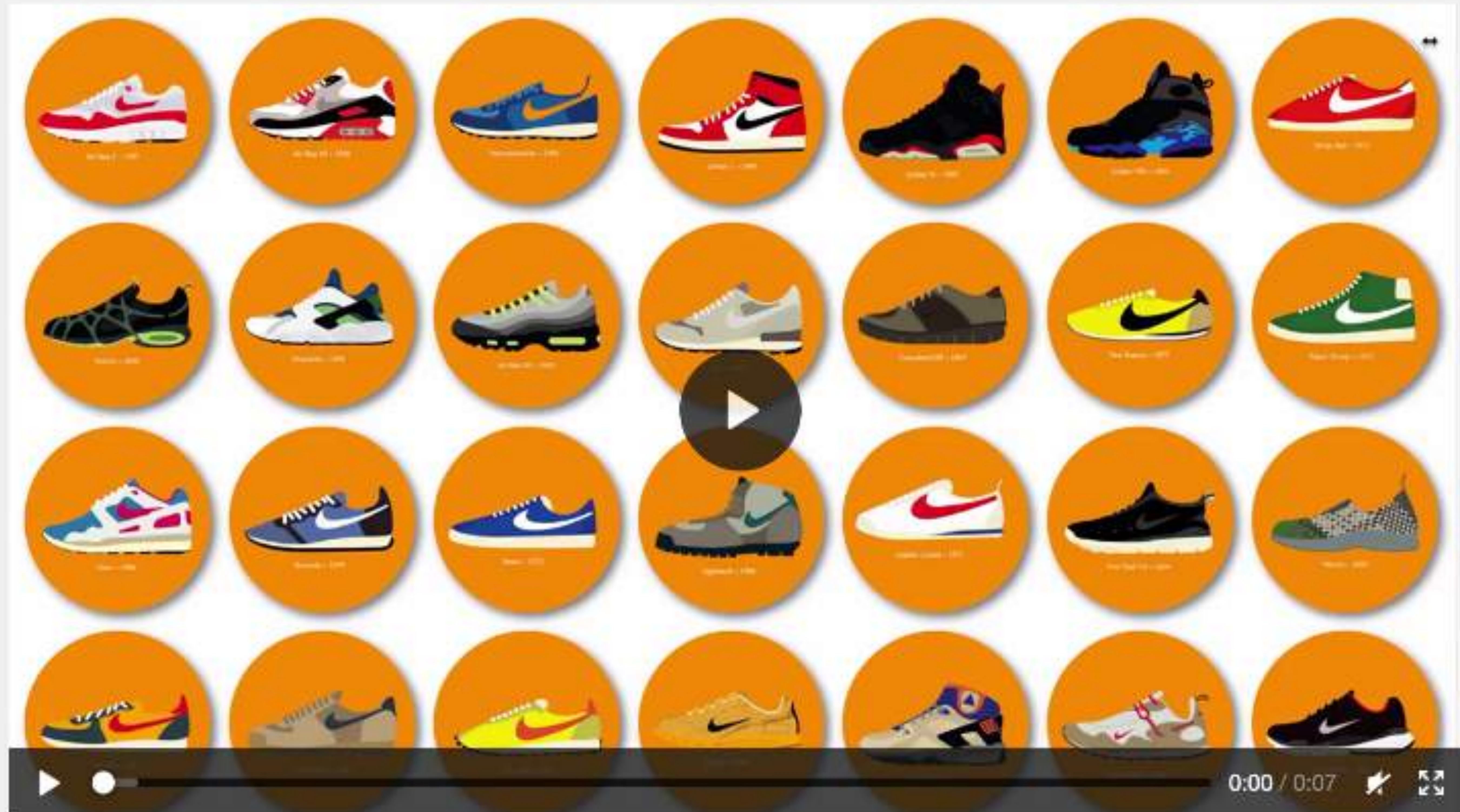
Using the `min-content`, `max-content` and `fit-content` properties

What's worse, lookin' jealous or crazy? Jealous or crazy?	Or like being walked all over lately, walked all over lately	I'd rather be crazy
---	--	---------------------

```
.grid7 {  
  display: grid;  
  grid-template-columns: min-content fit-content(25ch) max-content;  
}
```



Simple responsive grid



MANUAL PLACEMENT OF GRID ITEMS

(and the explicit grid)

Line-based placement

Using `grid-row-start`, `grid-row-end` and `grid-column-start`, `grid-column-end`



```
.grid8 {  
  display: grid;  
  grid-template-columns: 33%  
  33% 33%;  
  grid-template-rows: 20vh 20vh  
  20vh;  
}  
  
.grid8__item {  
  grid-column-start: 2;  
  grid-column-end: 3;  
  grid-row-start: 2;  
  grid-row-end: 3;  
}
```



SOME GRID SHORTHANDS

(because brevity is a virtue...maybe)

¬_(ツ)_/

Using the `grid-row` and `grid-column` shorthands

By default, grid items will take up the space of 1 grid cell



```
.grid9 {  
  display: grid;  
  grid-template-columns: 2em  
2em 2em 2em 2em 2em;  
  grid-template-rows: 2em 2em  
2em 2em 2em 2em;  
}  
  
.grid9__item:nth-child(1) {  
  grid-row: 2;  
  grid-column: 5;  
}  
  
.grid9__item:nth-child(2) {  
  grid-row: 4;  
  grid-column: 5;  
}  
  
.grid9__item:nth-child(3) {
```



The span keyword

Content can span multiple grid cells



```
.grid10 {  
  display: grid;  
  grid-template-columns: 20%  
20% 20% 20% 20%;  
  grid-template-rows: 12vh 12vh  
12vh 12vh 12vh;  
}  
  
.grid10__item {  
  grid-row: 1 / span 3;  
  grid-column: 2 / span 2;  
}
```



Using the `grid-area` shorthand

```
grid-area: <grid-row-start> / <grid-column-start> / <grid-row-end> / <grid-column-end>
```



```
.grid11 {  
  display: grid;  
  grid-template-columns: 20%  
  20% 20% 20%;  
  grid-template-rows: 20vh 20vh  
  20vh;  
}  
  
.grid11__item {  
  grid-area: 2 / 2 / 4 / 6;  
}
```



ASSIGNING NAMES TO GRID THINGS

Naming grid lines

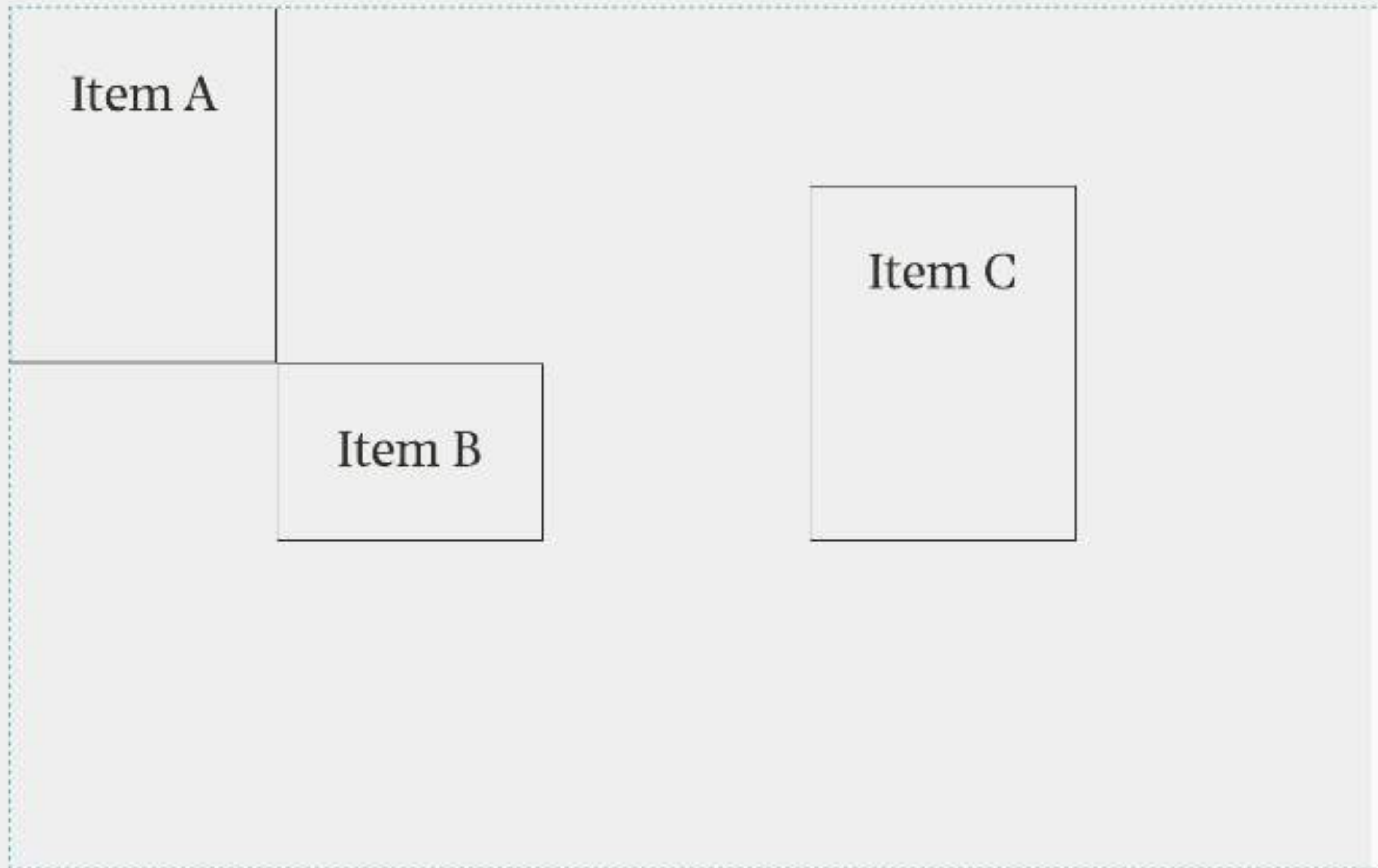
Item A	Item B	
Item C		Item D
Item E	Item F	

```
.grid12 {  
  display: grid;  
  grid-template-columns:  
    [alpha-start] 150px [alpha-end  
    beta-start] 150px [beta-end  
    gamma-start] 150px [gamma-end];  
}  
  
.grid12 .c {  
  grid-column: alpha-start /  
    beta-end;  
}
```



Naming grid areas

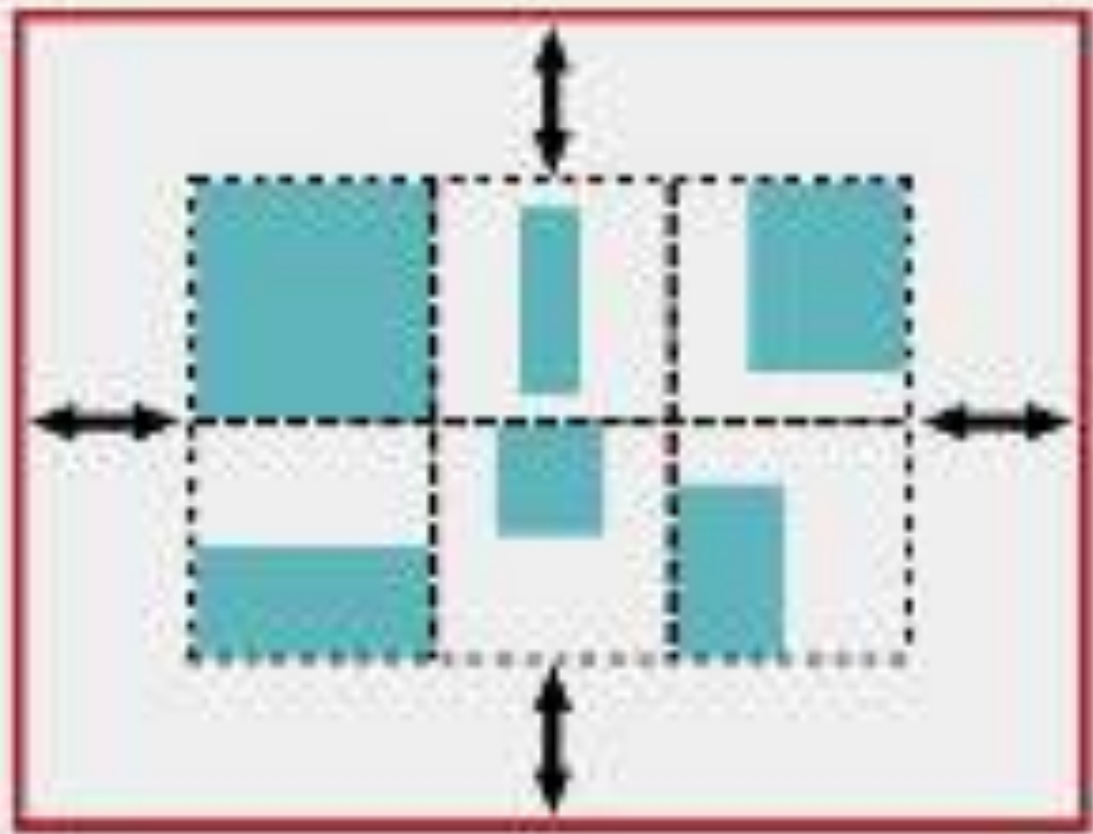
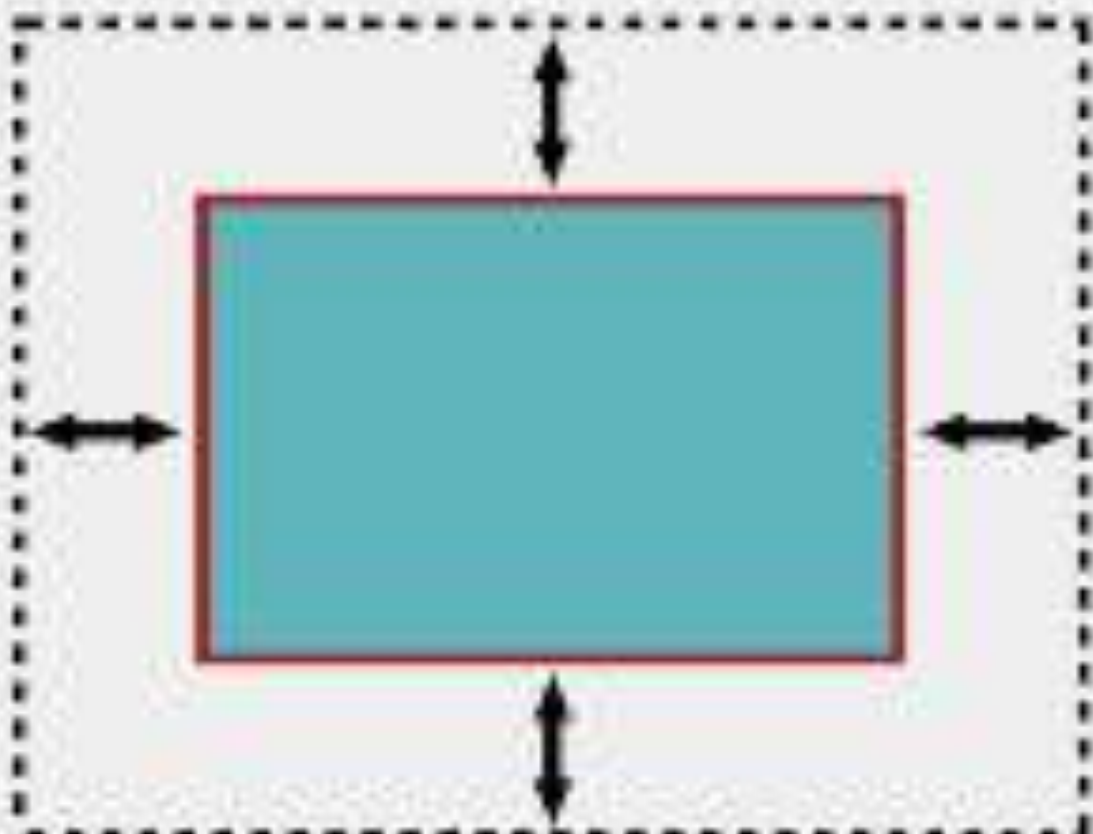
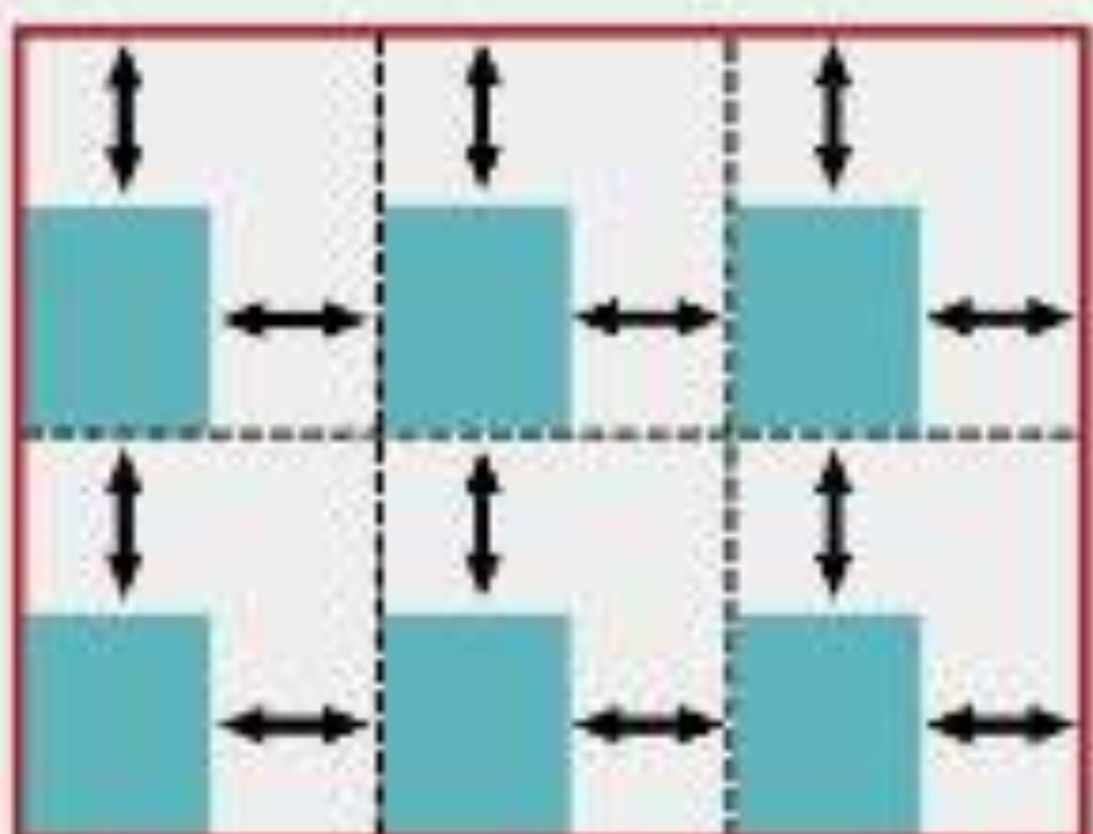
Using `grid-template-areas` and `grid-area`



```
.grid13 {  
  display: grid;  
  grid-template-columns: 150px  
150px 150px 150px;  
  grid-template-rows: 100px  
100px 100px;  
  grid-template-areas: 'a . .  
.'  
                        'a . .  
c'  
                        '. b .  
c';  
}  
  
.grid13 .a { grid-area: a; }  
.grid13 .b { grid-area: b; }  
.grid13 .c { grid-area: c; }
```



MAKING SENSE OF BOX ALIGNMENT

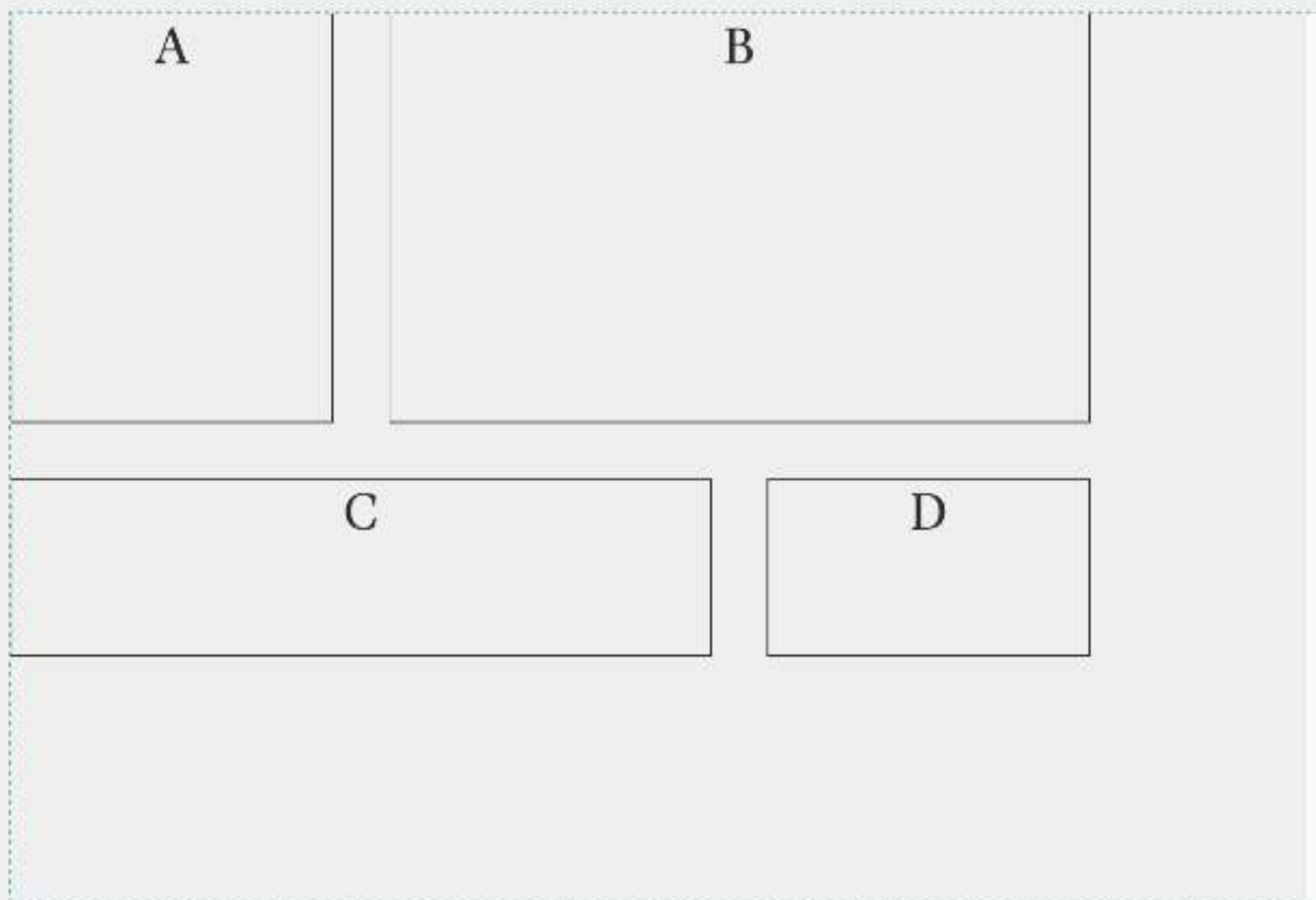
Property	Axis	Aligns		Applies to
<code>justify-content</code>	main/inline	content within element (effectively adjusts padding)		block containers, flex containers and grid containers
<code>align-content</code>	cross/block			
<code>justify-self</code>	inline	element within parent (effectively adjusts margins)		block-level boxes, absolutely-positioned boxes and grid items
<code>align-self</code>	cross/block			absolutely-positioned boxes, flex items and grid items
<code>justify-items</code>	inline	items inside box (controls child items)		block containers and grid containers
<code>align-items</code>	cross/block			flex-containers and grid-containers

Source: [CSS Box Alignment Module Level 3](#)




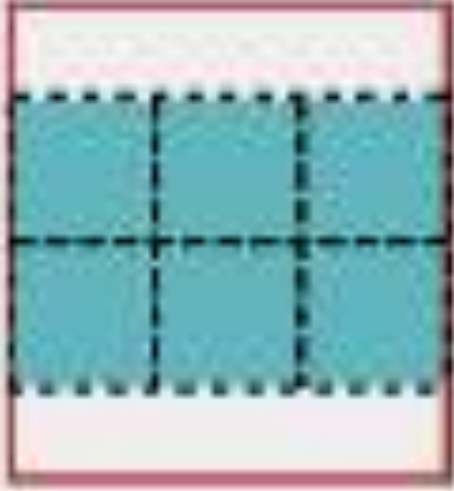
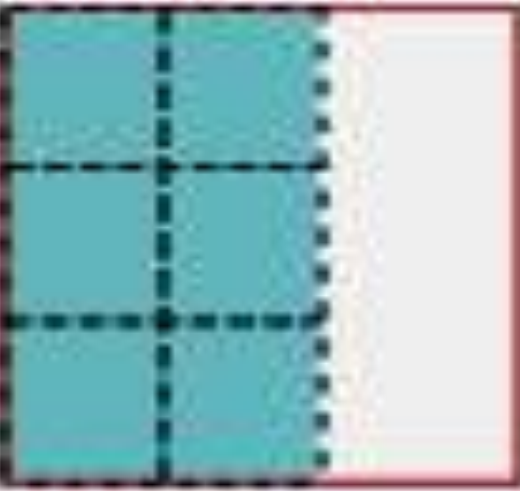

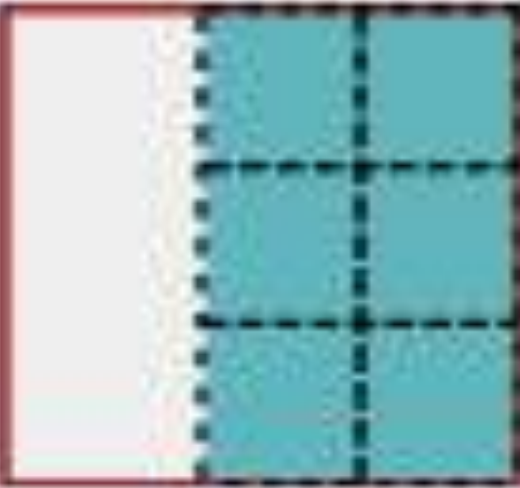
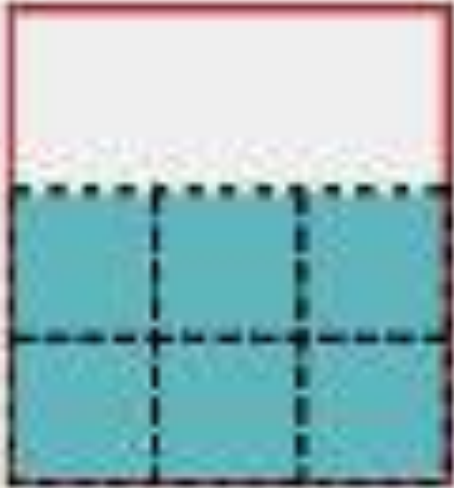
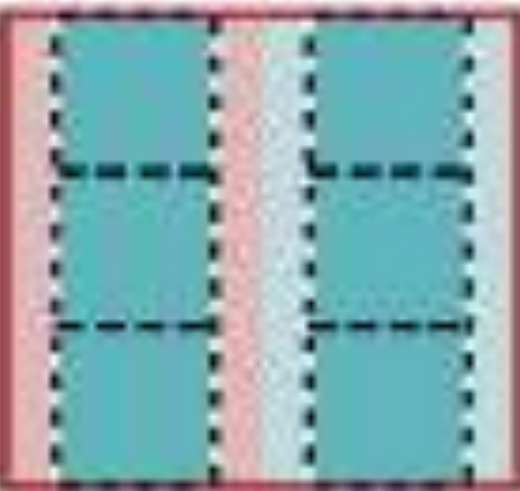
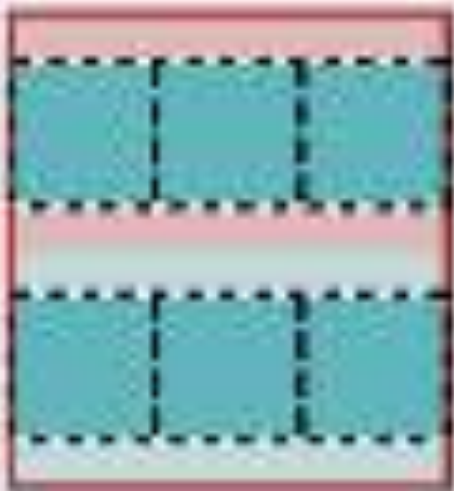
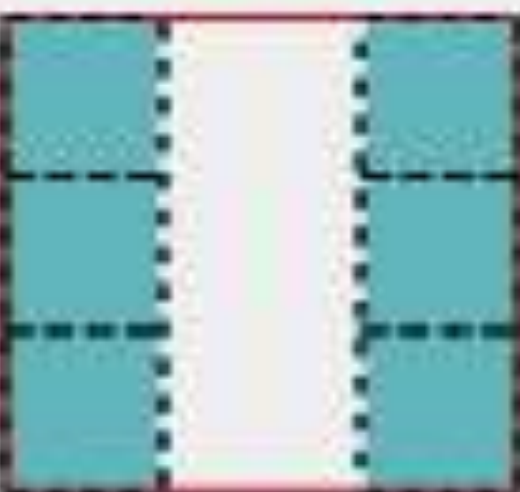
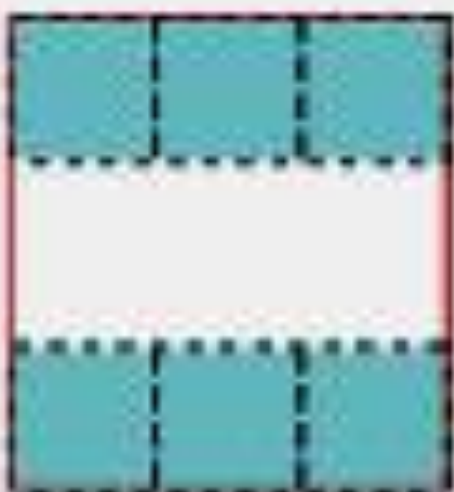
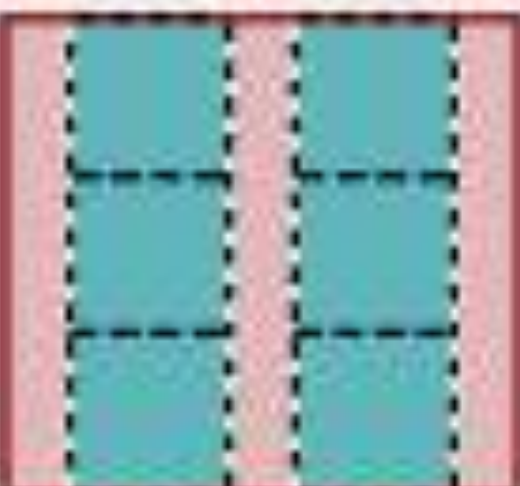
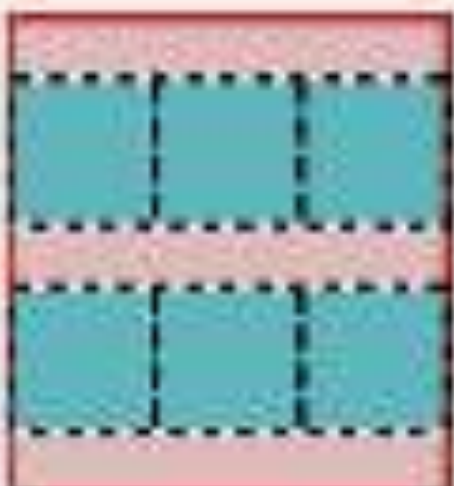
justify/align-content

content-distribution properties



```
.content {  
  justify-content: normal;  
  align-content: normal;  
  
  display: grid;  
  grid-template-columns: repeat(3,  
25%);  
  grid-template-rows: repeat(3,  
20%);  
  grid-gap: 1em;  
  grid-template-areas:  
    "a b b"  
    "a b b"  
    "c c d";  
}  
  
.content_itemA {  
  grid-area: a;  
}
```



Values	<code>justify-content</code>	<code>align-content</code>
<code>center</code>		
<code>start</code>		
<code>end</code>		
<code>space-around</code>		
<code>space-between</code>		
<code>space-evenly</code>		

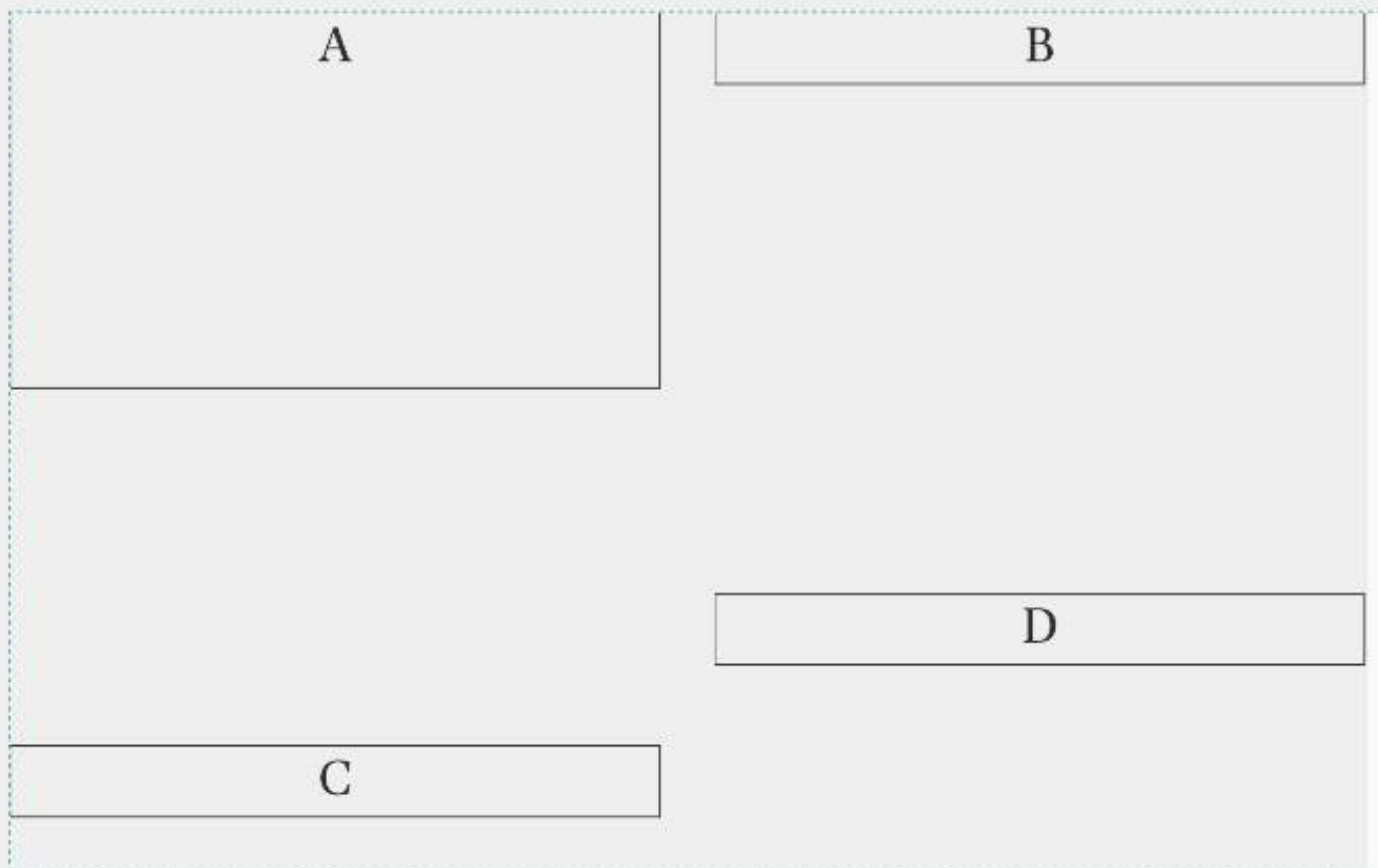


Values	justify-content	align-content
center		
start		
end		
space-between		
space-around		
space-between		
space-around		



justify/align-self

self-alignment properties

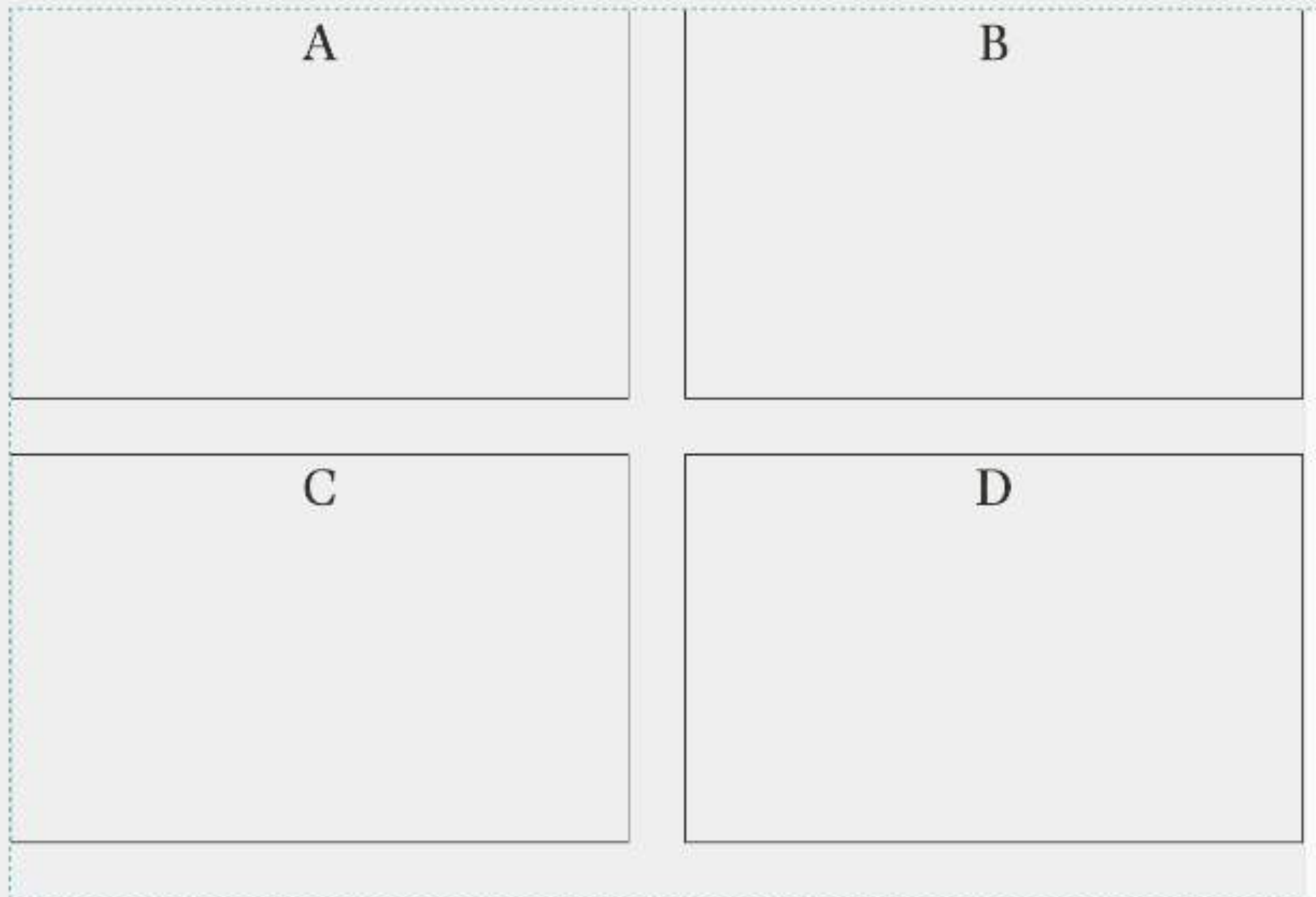


```
.self {  
  display: grid;  
  grid-template-columns:  
repeat(4, 1fr);  
  grid-gap: 1em;  
  grid-auto-rows: calc(25% -  
1em);  
  grid-template-areas:  
    "a a b b"  
    "a a b b"  
    "c c d d"  
    "c c d d";  
}  
  
.self_itemA {  
  grid-area: a;  
}  
  
.self_itemB {
```



justify/align-items

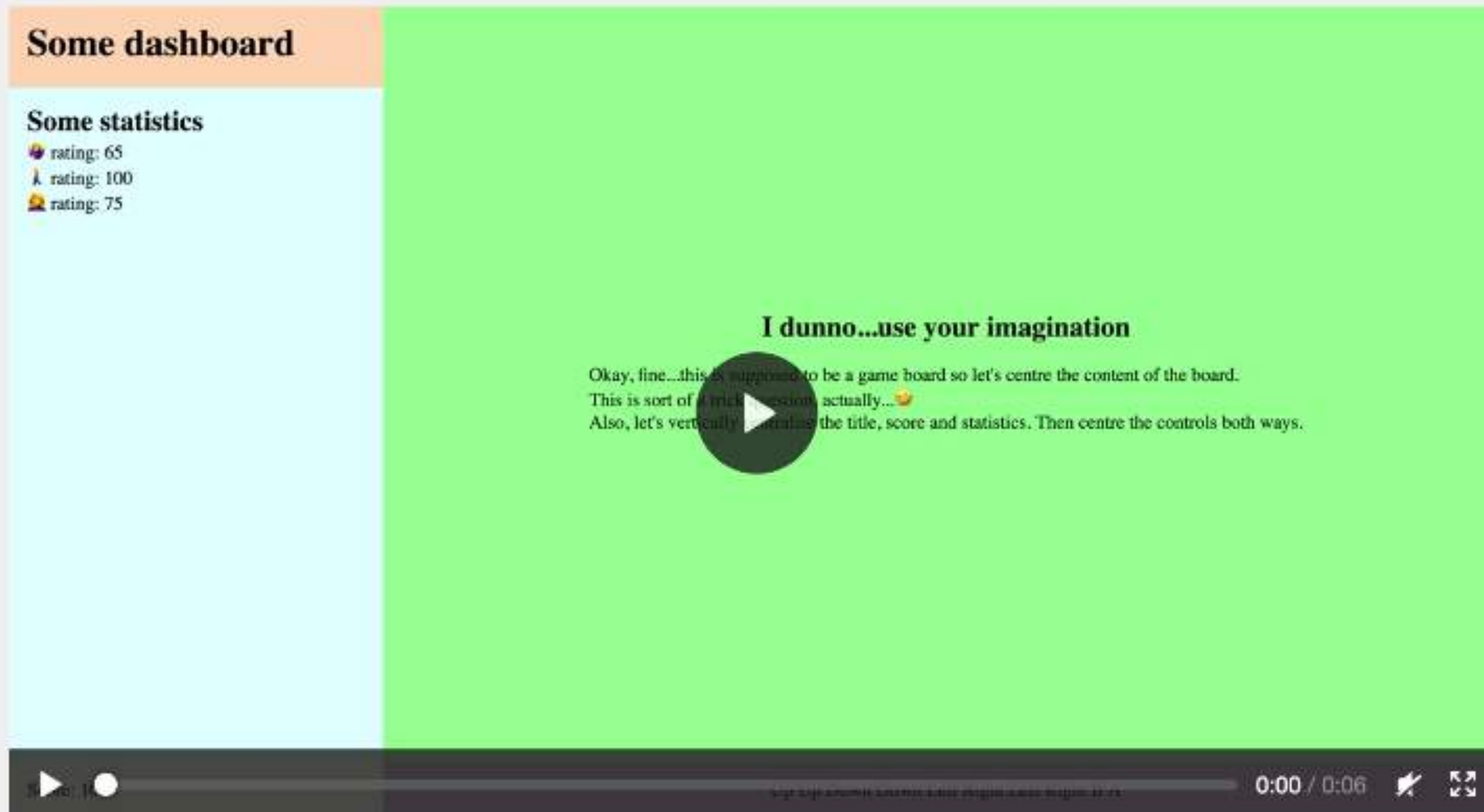
defaults for justify/align-self



```
.items {  
  justify-items: normal;  
  align-items: normal;  
  
  display: grid;  
  grid-template-columns: repeat(4,  
1fr);  
  grid-gap: 1em;  
  grid-auto-rows: calc(25% - 1em);  
  grid-template-areas:  
    "a a b b"  
    "a a b b"  
    "c c d d"  
    "c c d d";  
}  
  
.items_itemA {  
  grid-area: a;  
}
```



Simple responsive dashboard





GRID VERSUS FLEXBOX?



GRID VERSUS FLEXBOX?

WRONG QUESTION

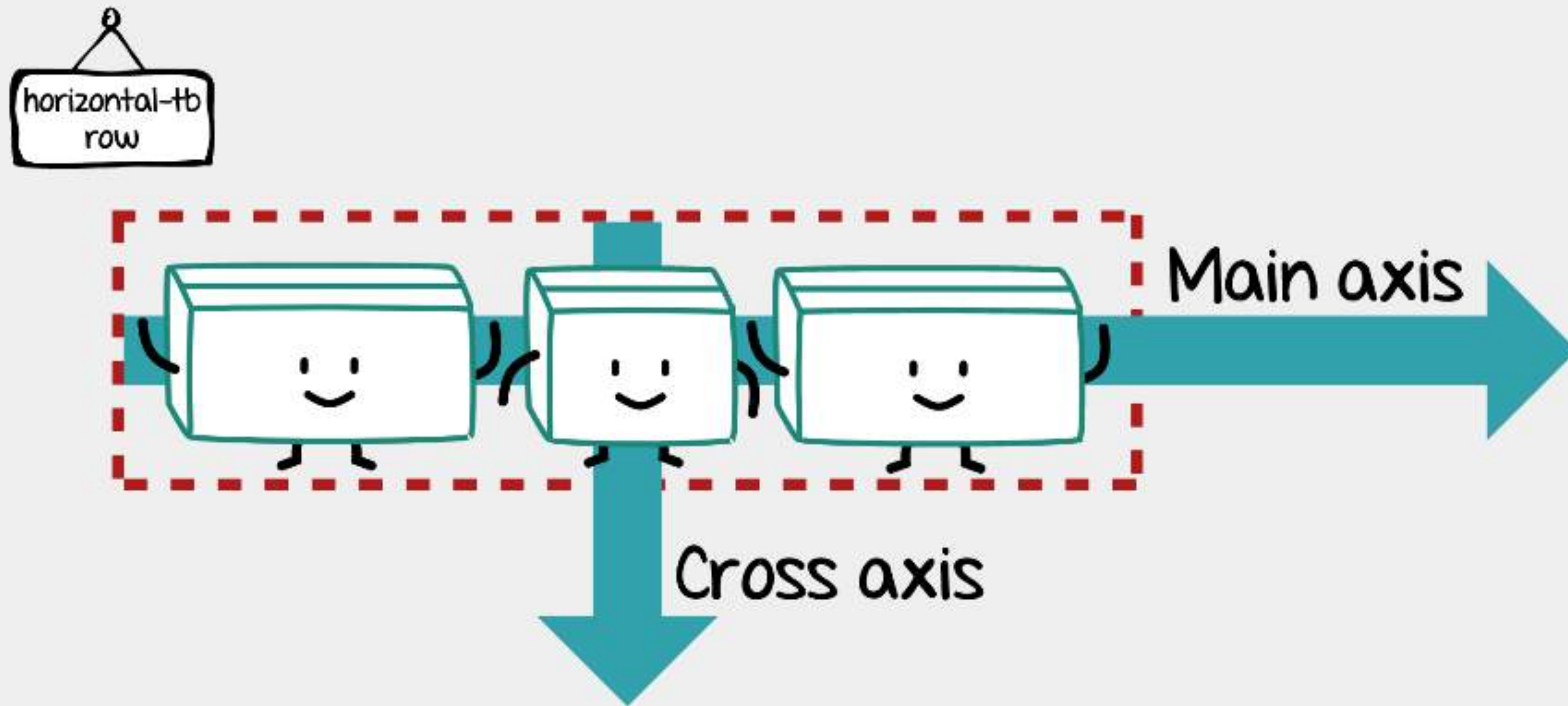


Grid AND Flexbox

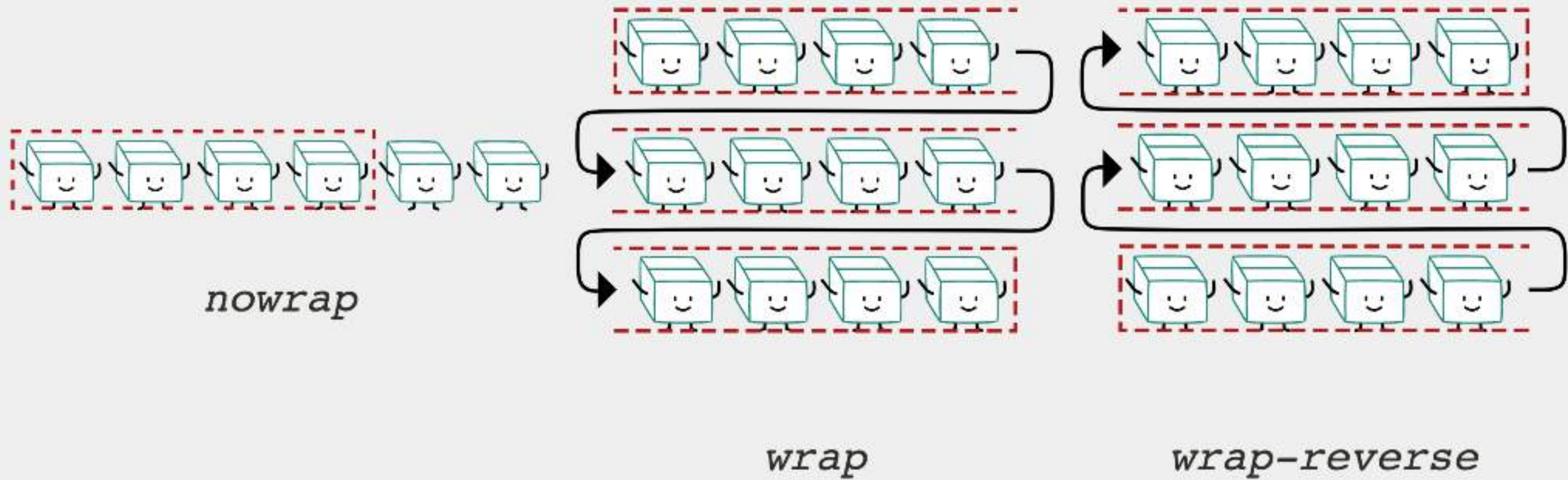


FIGURING OUT FLEXBOX

Flex axes



Flex lines



Flex directions

1一	2二	3三	4四	5五	6六	7七
8八	9九	10十	11十一	12十二	13十三	14十四
15十五	16十六	17十七	18十八	19十九	20二十	

```
.directions .wrapper {  
  display: flex;  
  flex-wrap: wrap;  
  writing-mode: horizontal-tb;  
  flex-direction: row;  
}  
  
.directions .box {  
  height: 6em;  
  width: 6em;  
  border: 1px solid;  
}
```





**WHAT IS THE MOST COMMON MISTAKE
DEVELOPERS MAKE WHEN USING FLEXBOX?**



Not using the `flex` shorthand

“Authors are encouraged to control flexibility using the `flex shorthand` rather than with its longhand properties directly, as the shorthand `correctly resets` any unspecified components to accommodate common uses.”

—CSS Flexible Box Layout Module Level 1



Basic flex keyword values

<code>initial</code>	<code>0 1 auto</code>	cannot grow but can shrink when there isn't enough space
<code>auto</code>	<code>1 1 auto</code>	can grow and shrink to fit available space
<code>none</code>	<code>0 0 auto</code>	cannot grow or shrink, AKA inflexible
<code><positive- number></code>	<code><positive- number> 1 0</code>	can grow and shrink, extent of growth depends on flex factor



“When a box is a flex item, flex is consulted instead of the main size property to determine the main size of the box.”

—CSS Flexible Box Layout Module Level 1



The flex syntax

brackets are for grouping

[**<'flex-grow'>** **<'flex-shrink'>?** || **<'flex-basis'>**]

preceding type/
word/group is
optional

seperates 2 options,
one or more must occur,
order doesn't matter




```
/* One value, unitless number: flex-grow */
flex: 3;

/* One value, width/height: flex-basis */
flex: 200px;
flex: 45em;

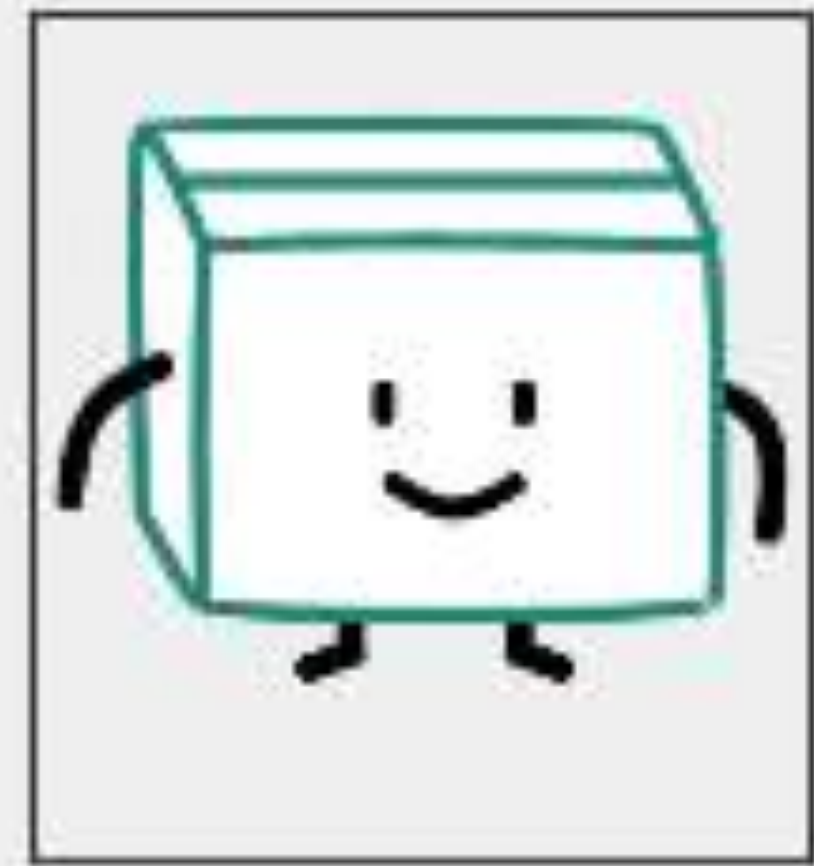
/* Two values: flex-grow | flex-basis */
flex: 1 25ch;

/* Two values: flex-grow | flex-shrink */
flex: 2 1;

/* Three values: flex-grow | flex-shrink | flex-basis */
flex: 2 3 30%;
```



Aligning with auto margins



```
.automargin {  
  display: flex;  
}  
  
.automargin div {  
  border: 1px solid;  
  margin: auto;  
}
```



Defining “auto” by Erika Etemad (AKA fantasai)



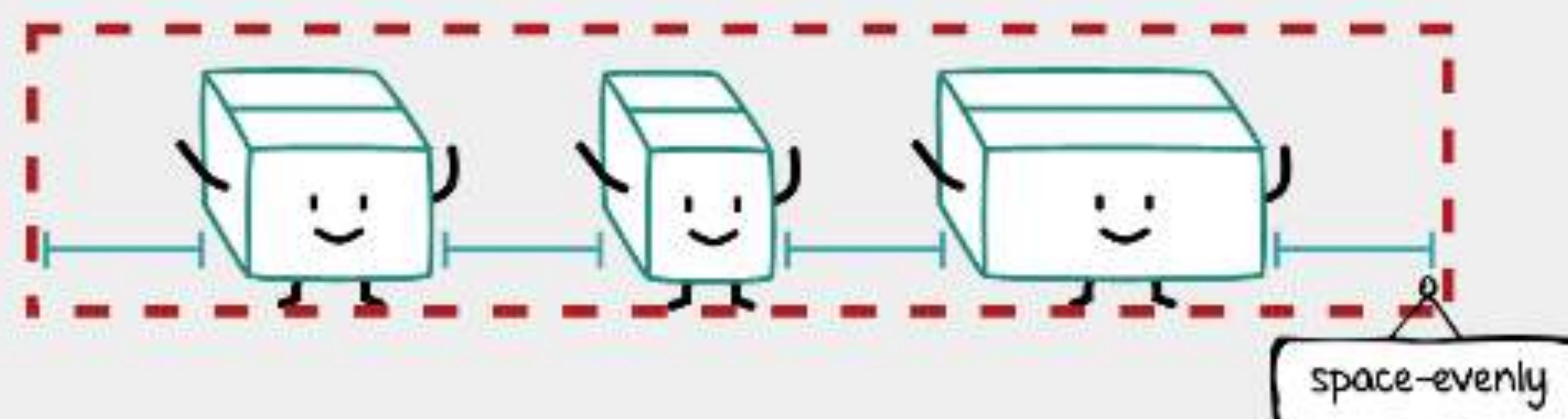
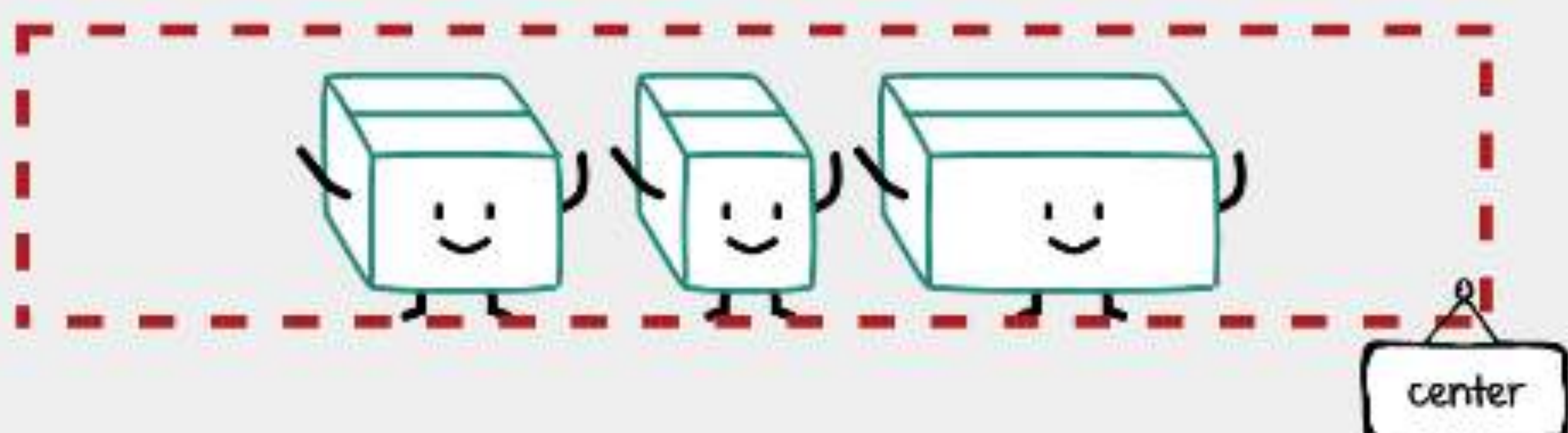
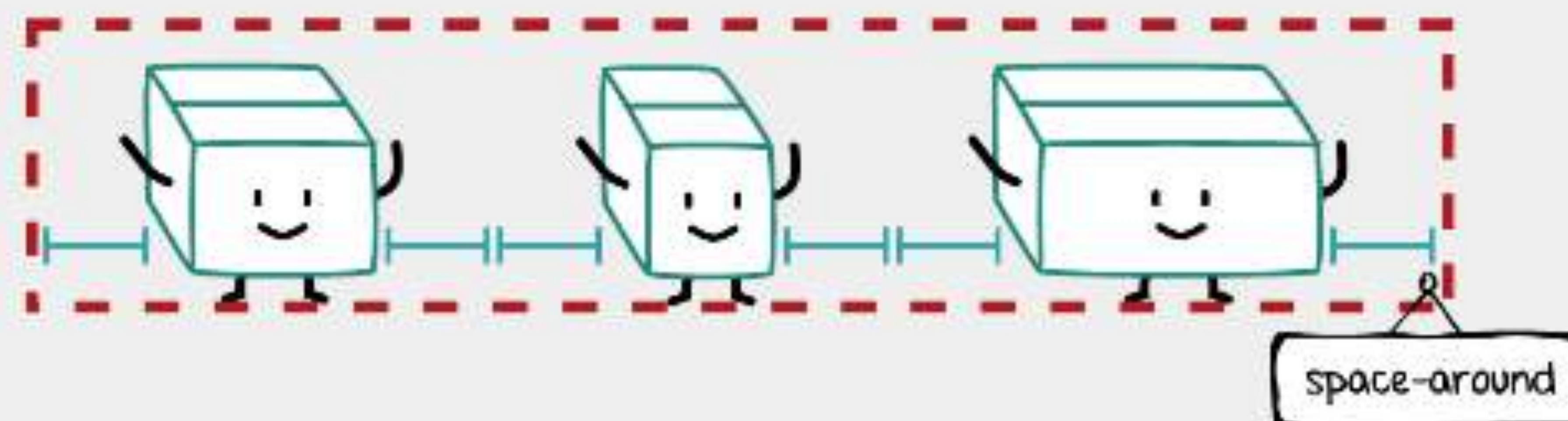
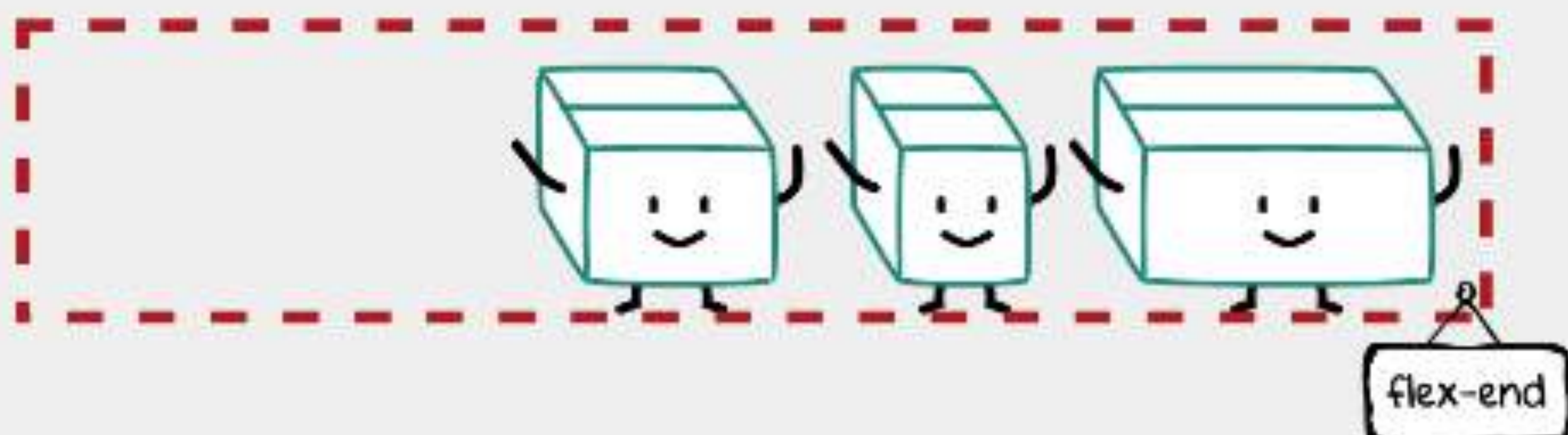
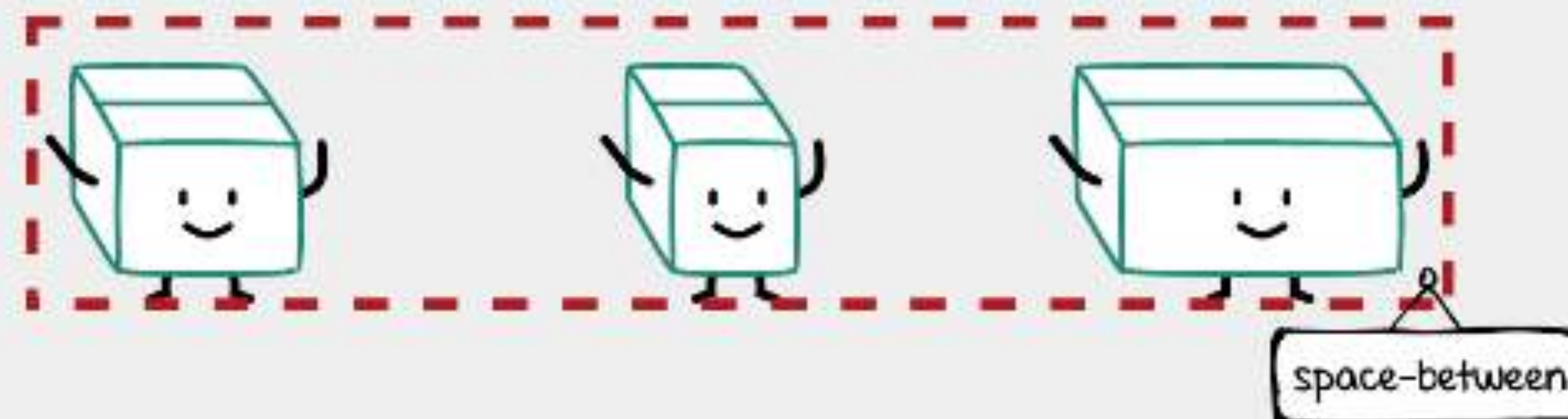
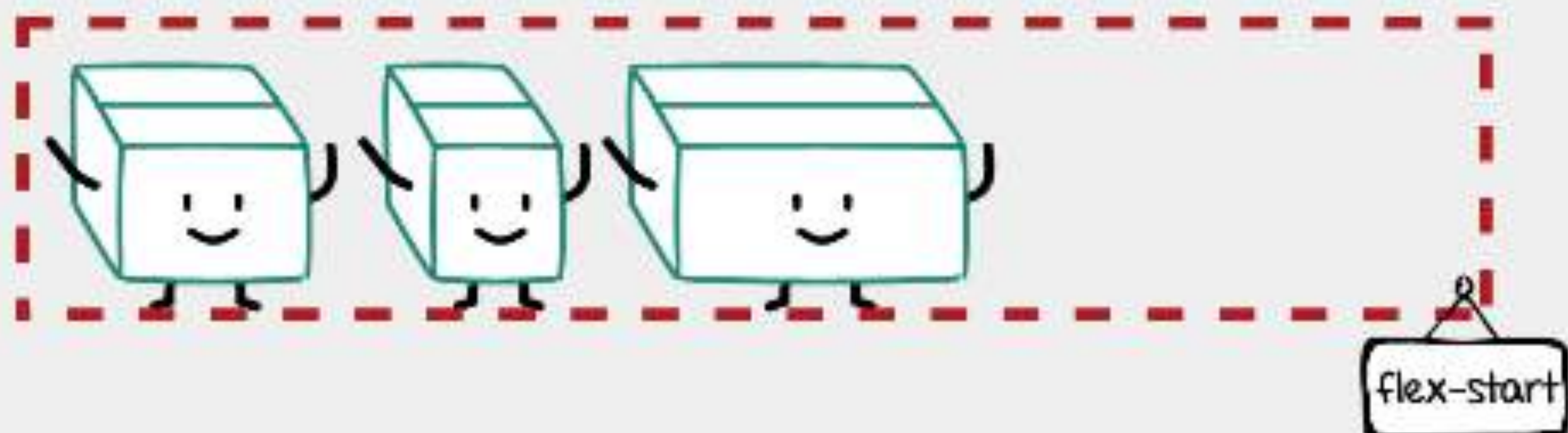
Aligning along the main axis

`justify-content` helps distribute extra free space left over **after** flexible lengths and auto margins are resolved.

1一	2二	3三	4四	5五	6六	7七	8八	9九
10十	11十一	12十二	13十三	14十四	15十五	16十六	17十七	18十八
							19十四	20二十

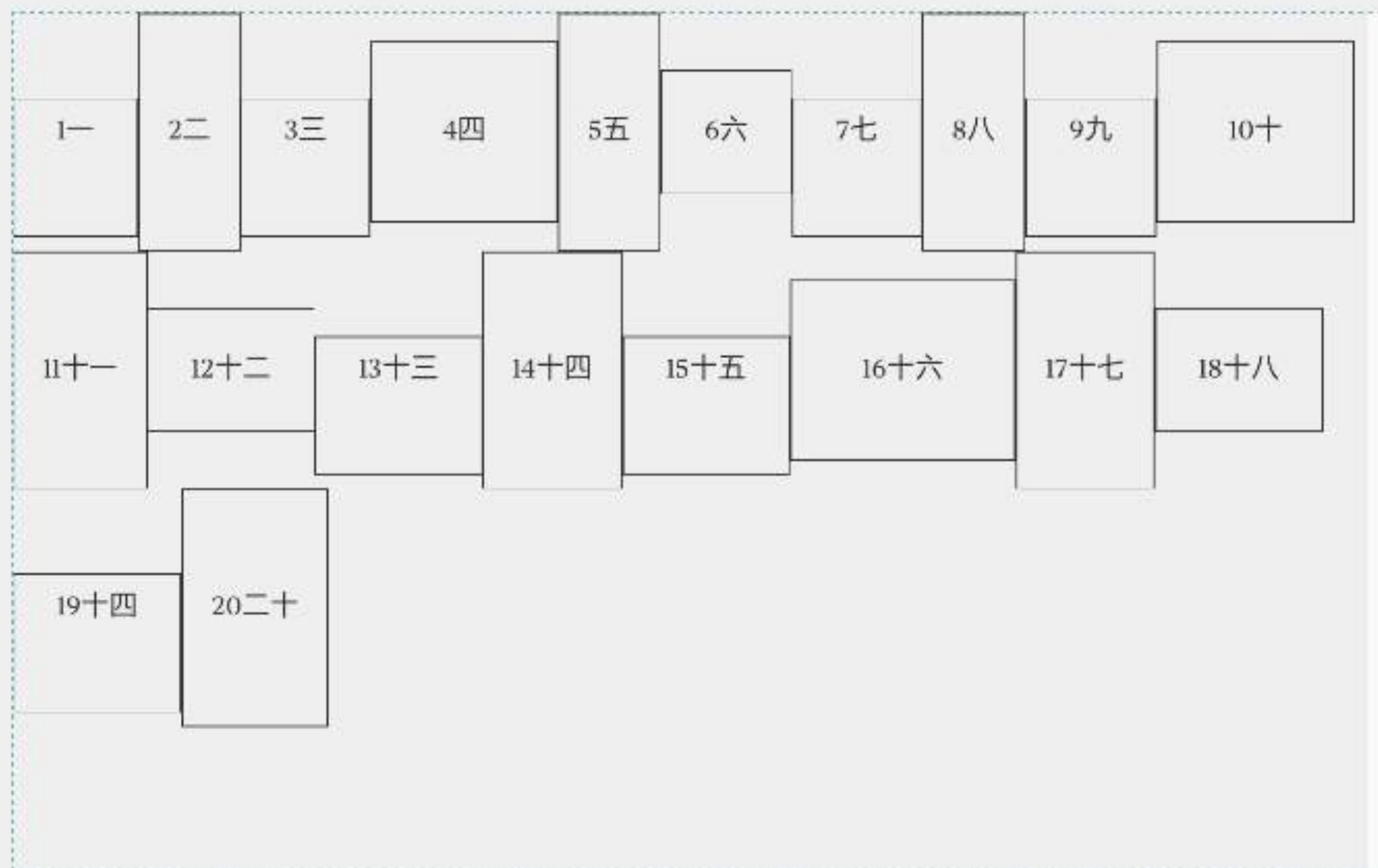
```
.mainaxis .wrapper {  
  display: flex;  
  flex-wrap: wrap;  
  justify-content: flex-end;  
}  
  
.mainaxis .box {  
  height: 5em;  
  width: 5em;  
  border: 1px solid;  
}
```





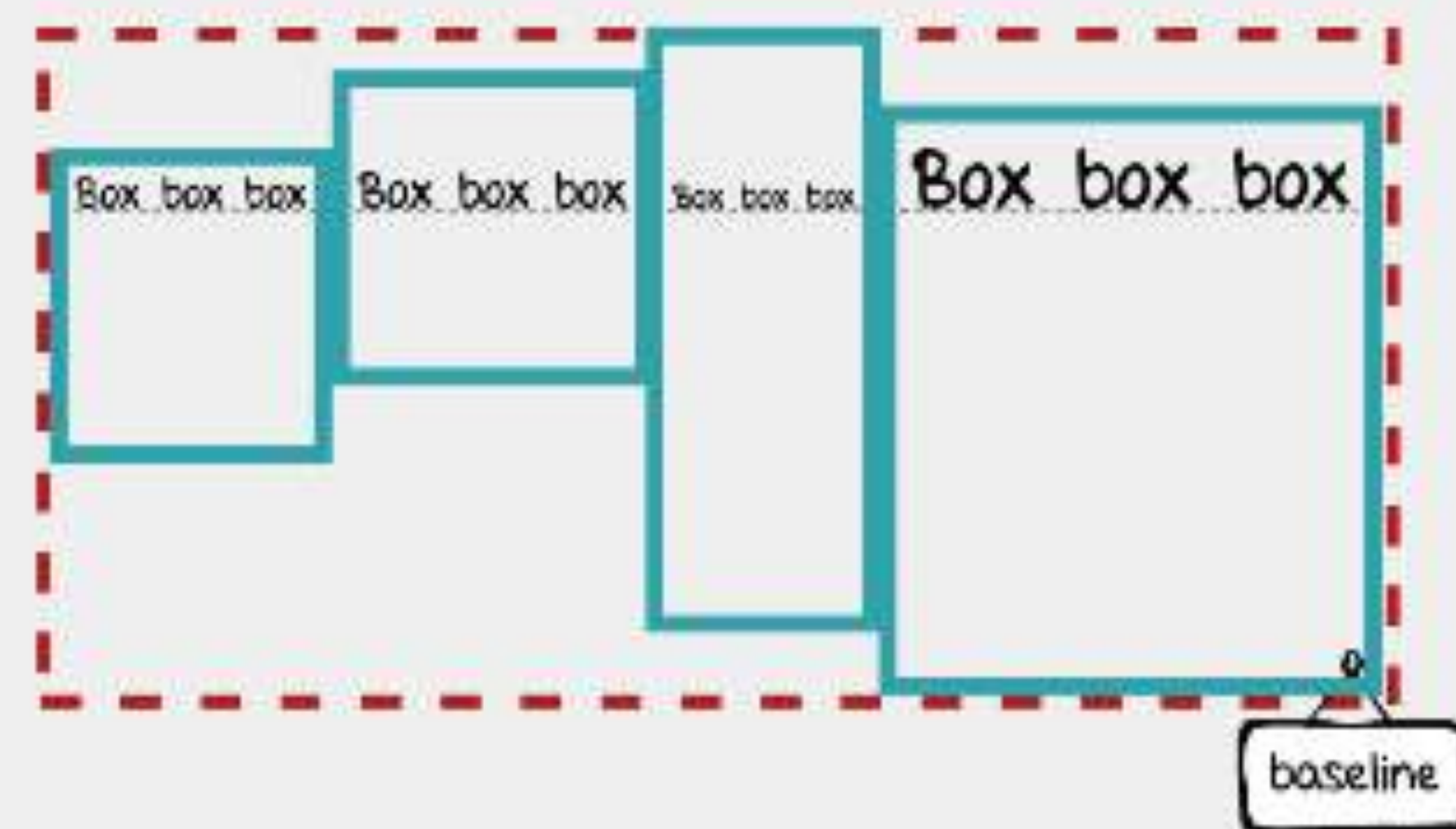
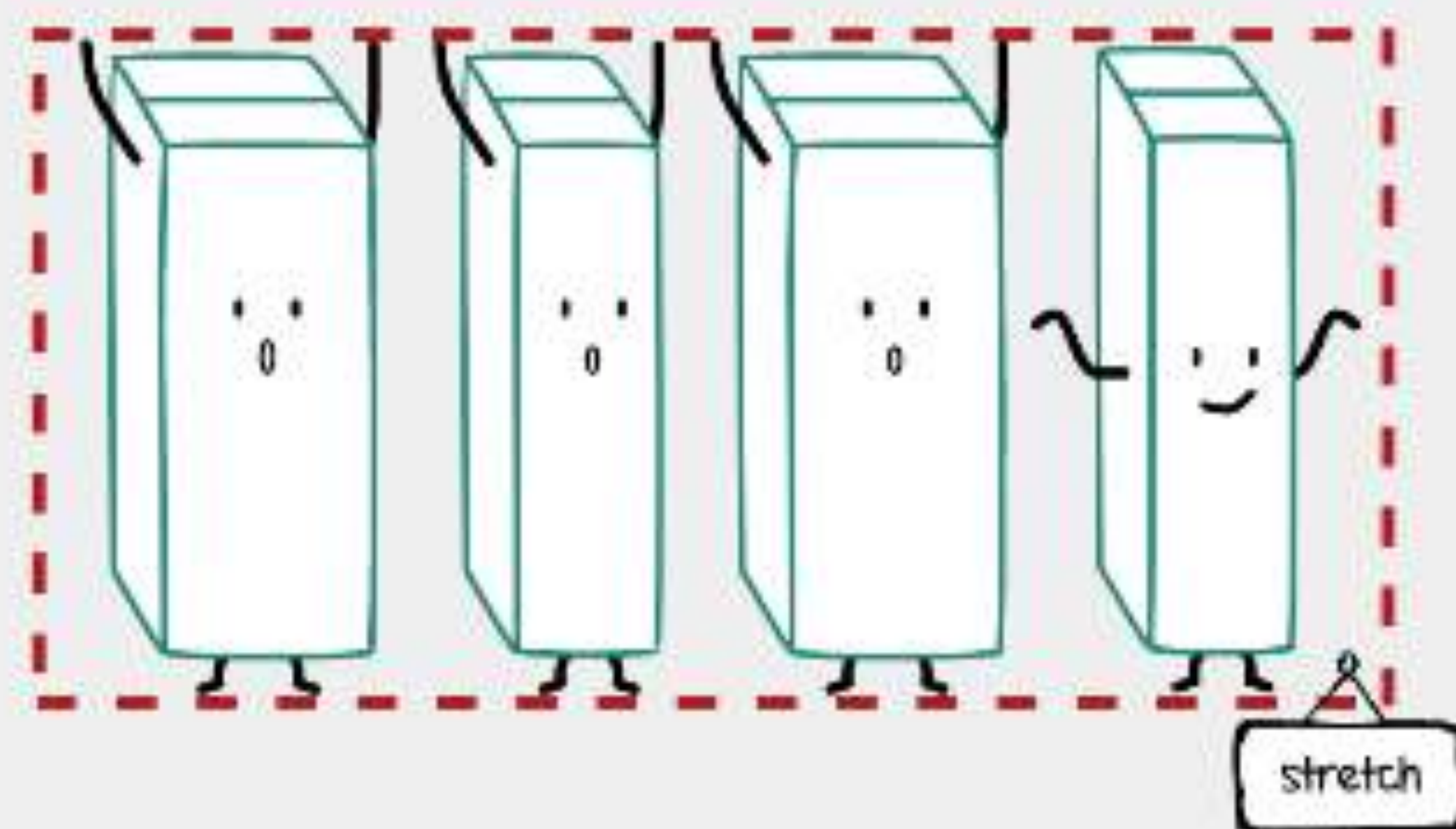
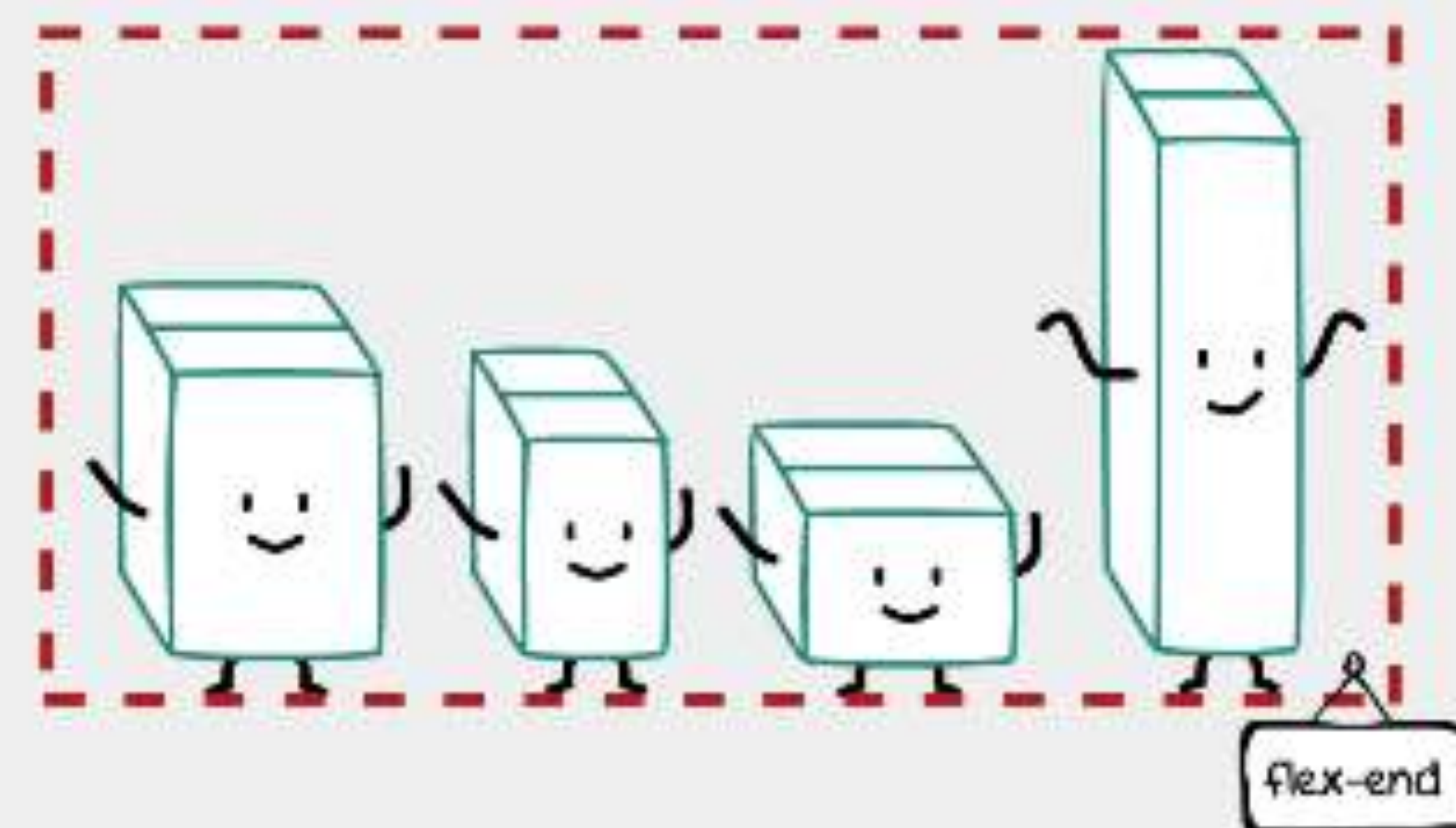
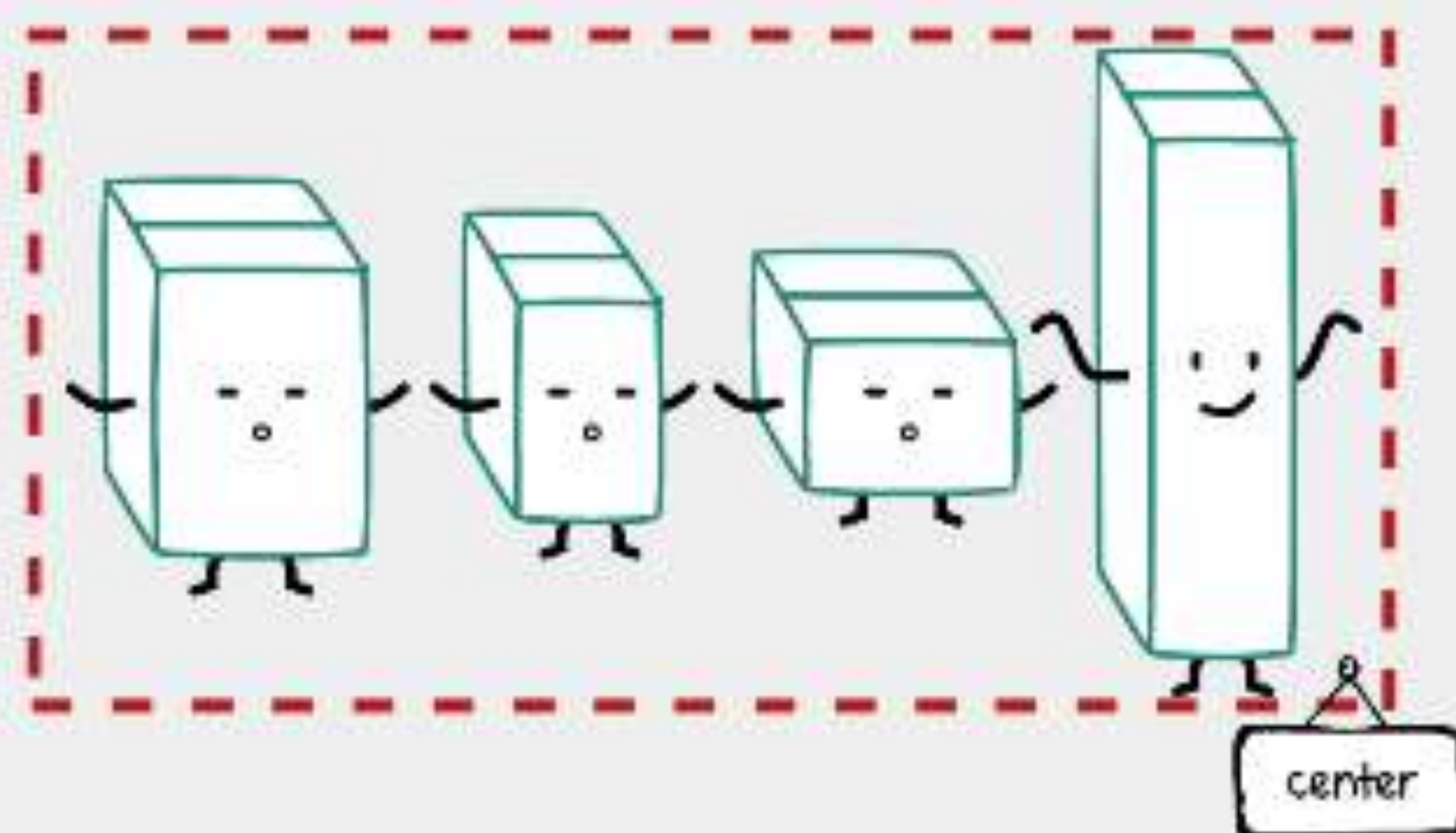
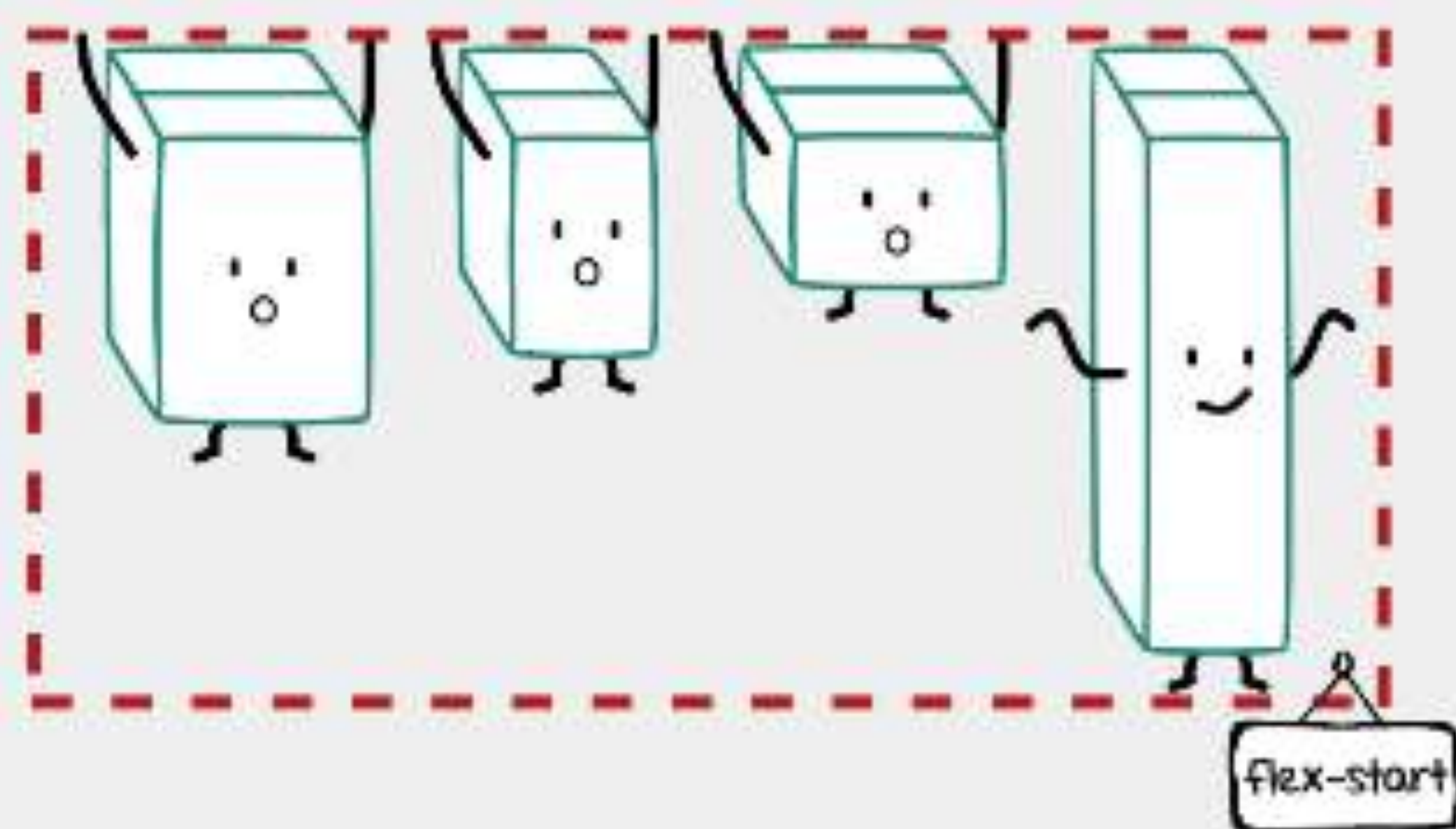
Aligning along the cross axis

`align-items` sets the default alignment for all flex items along the cross axis of the flex line. Over-ridable by `align-self`.



```
.crossaxis .wrapper {  
  display: flex;  
  flex-wrap: wrap;  
  align-items: baseline;  
}  
  
.crossaxis .box {  
  border: 1px solid  
}  
  
.crossaxis .box:nth-child(2n) {  
  padding: 2.5em;  
}  
  
.crossaxis .box:nth-child(3n) {  
  padding: 1.5em;  
}  
  
.crossaxis .box:nth-child(2n+1)
```





Packing flex lines

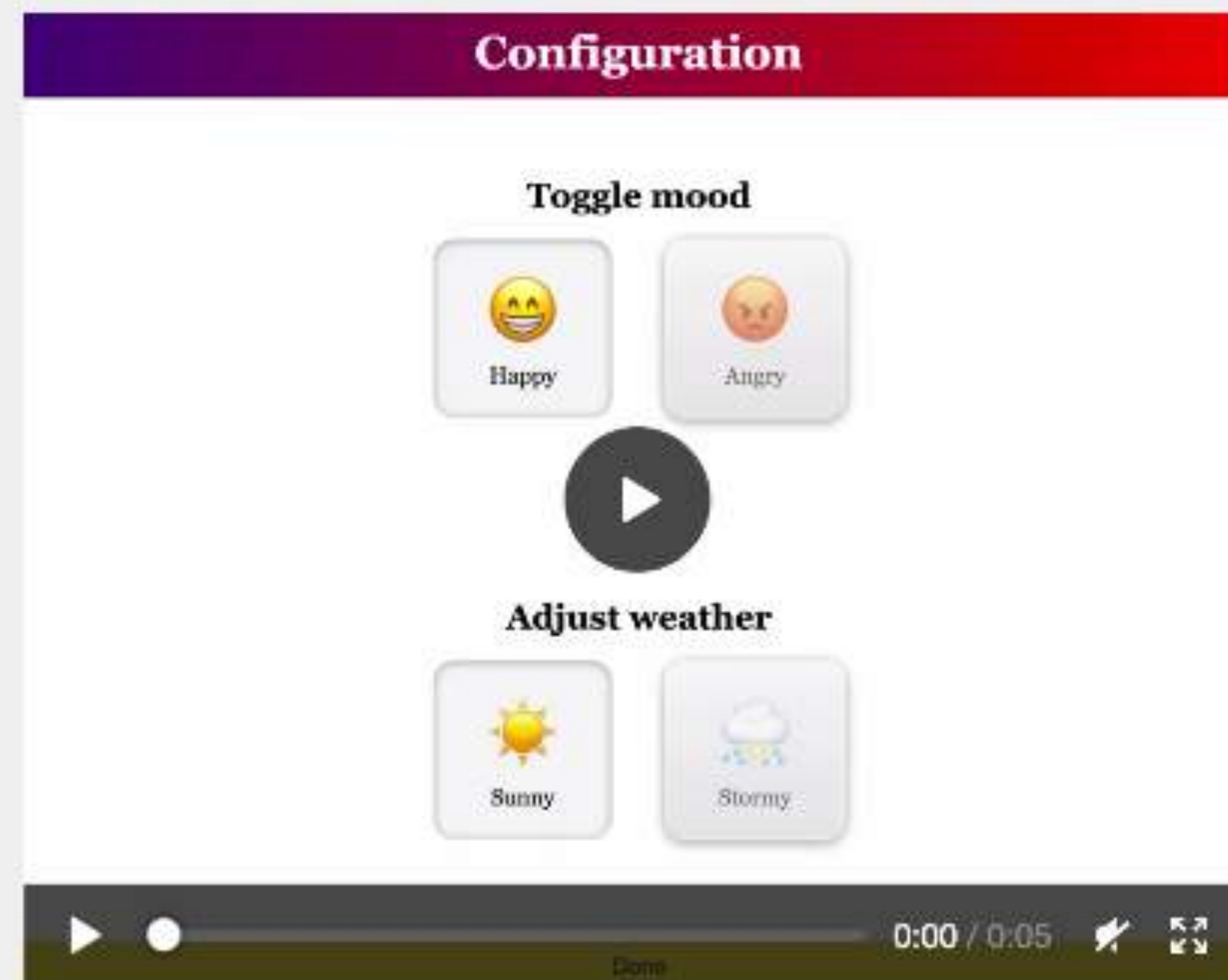
`align-content` aligns flex lines within the flex container if there is extra space along the **cross-axis**.

1一	2二	3三	4四	5五	6六	7七	8八	9九
10十	11十一	12十二	13十三	14十四	15十五	16十六	17十七	18十八
19十九	20二十							

```
.packaxis .wrapper {  
  display: flex;  
  flex-wrap: wrap;  
  align-content: center;  
}  
  
.packaxis .box {  
  height: 5em;  
  width: 5em;  
  border: 1px solid;  
}
```



Responsive configuration page





IS IT SAFE TO USE CSS GRID IN PRODUCTION?



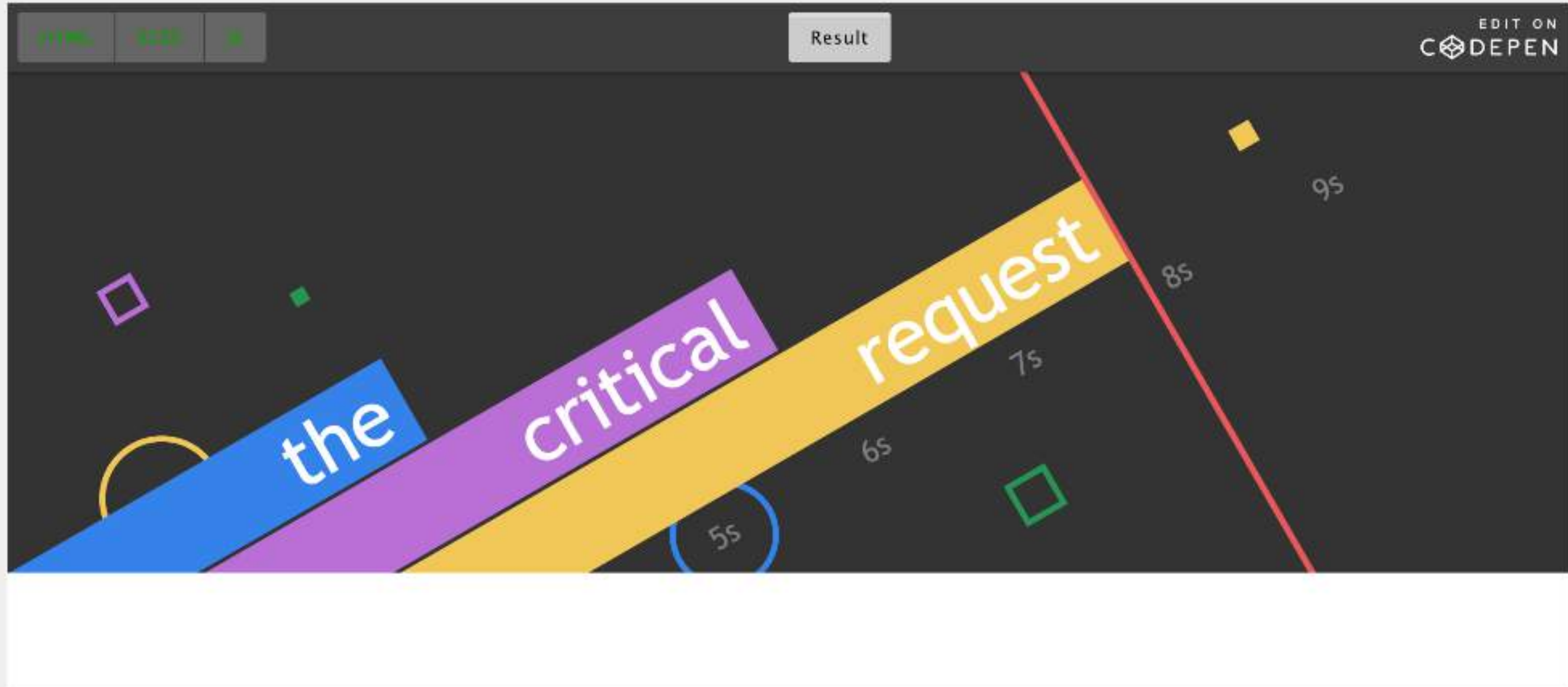
Falling back with style

Using @supports AKA feature queries

```
.selector {  
  /* Styles that are supported in old browsers */  
}  
  
@supports (property:value) {  
  .selector {  
    /* Styles for browsers that support the specified property */  
  }  
}
```



Diagonal header





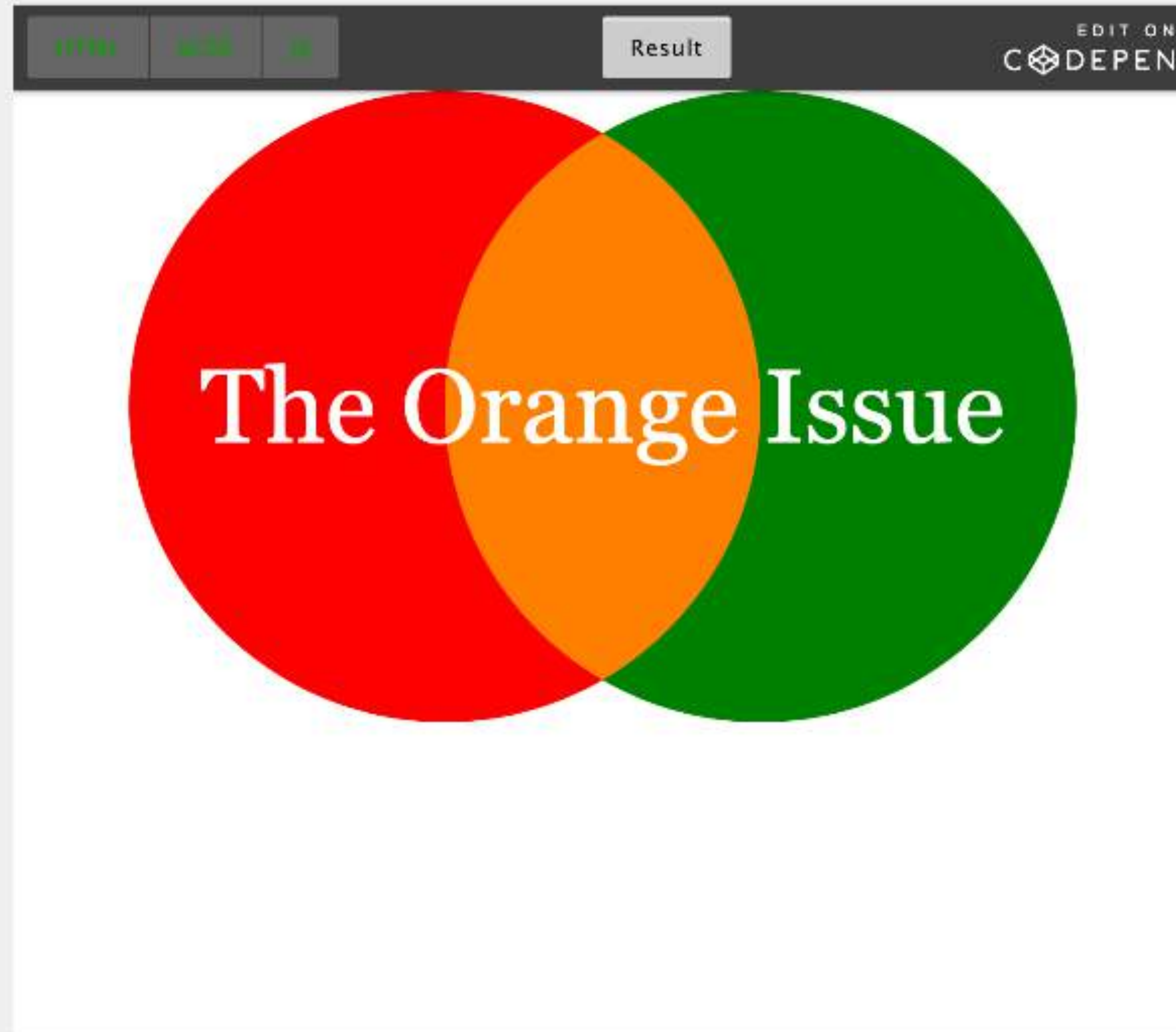
HOW WE CAN IMPROVE DESIGN USING GRIDS?

and

**WHAT ARE SOME OF THE AMAZING
TECHNIQUES YOU'VE SEEN THAT USE CSS
GRID?**

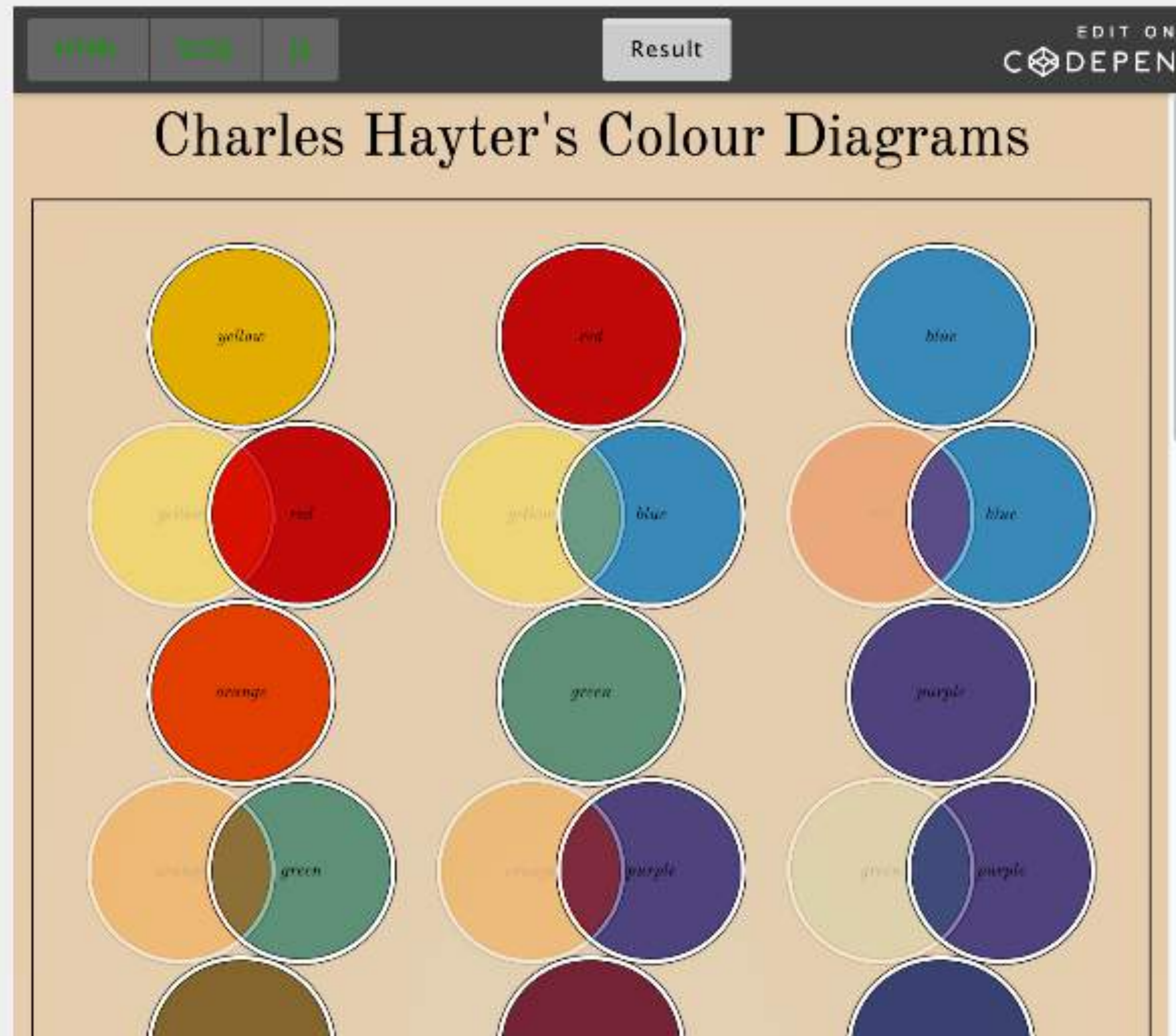


Overlap



Contents





Vertical whitespace



How to get hired

"LinkedIn is more than just a job portal, use it as an avenue for networking and reaching out, as well as a way to share knowledge."

"Nowadays people are looking to move on from their current roles for growth and development rather than just a salary bump."

"Be aware of what you want out of your career before going in for interviews, so that you won't find yourself in situations where there is obviously no synergy between yourself and the company in question."

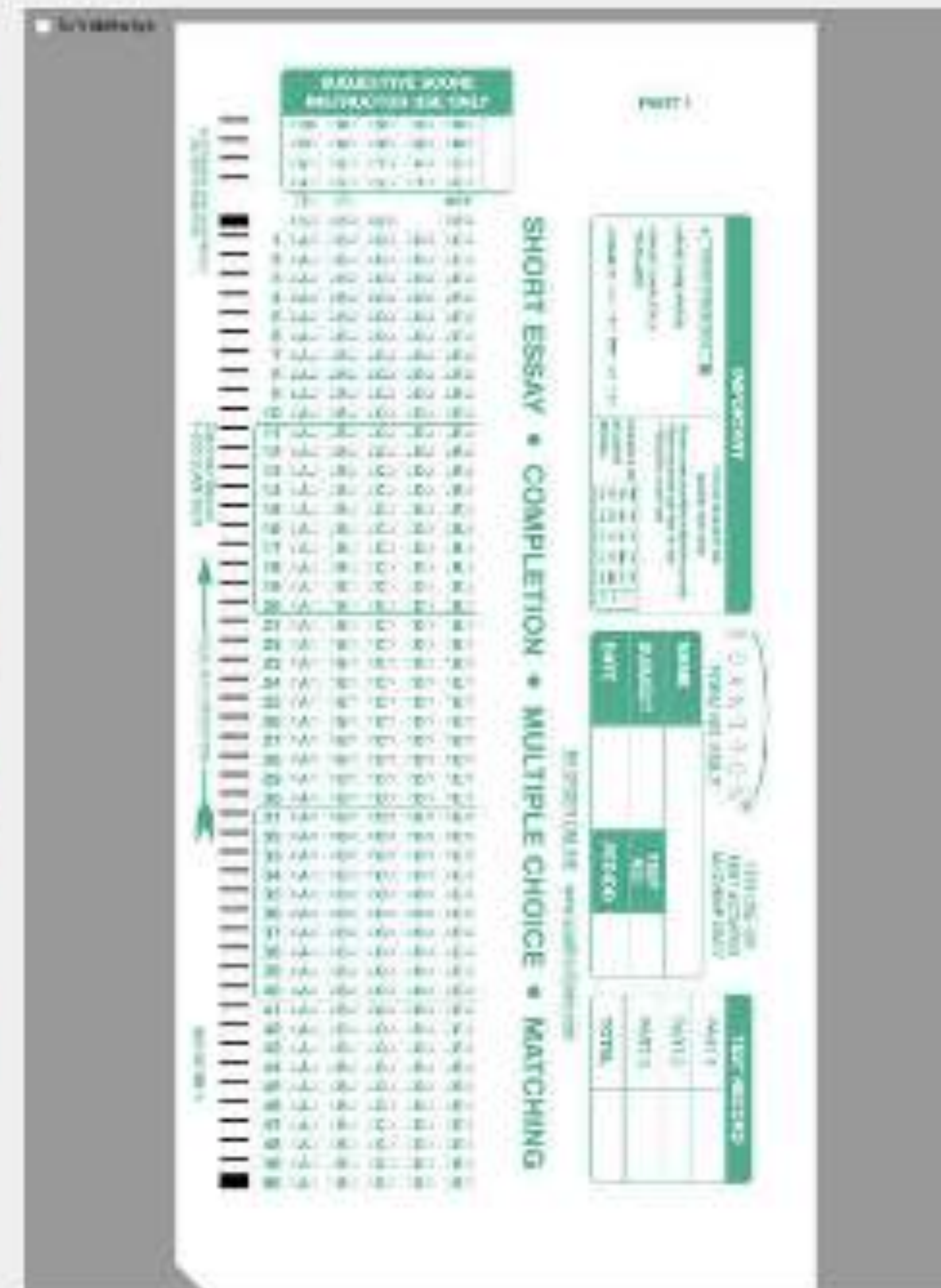
<https://singaporecss.github.io/specials/s2701.html>



CSS grid showcase



by Andy Barefoot



by Jon Kantner



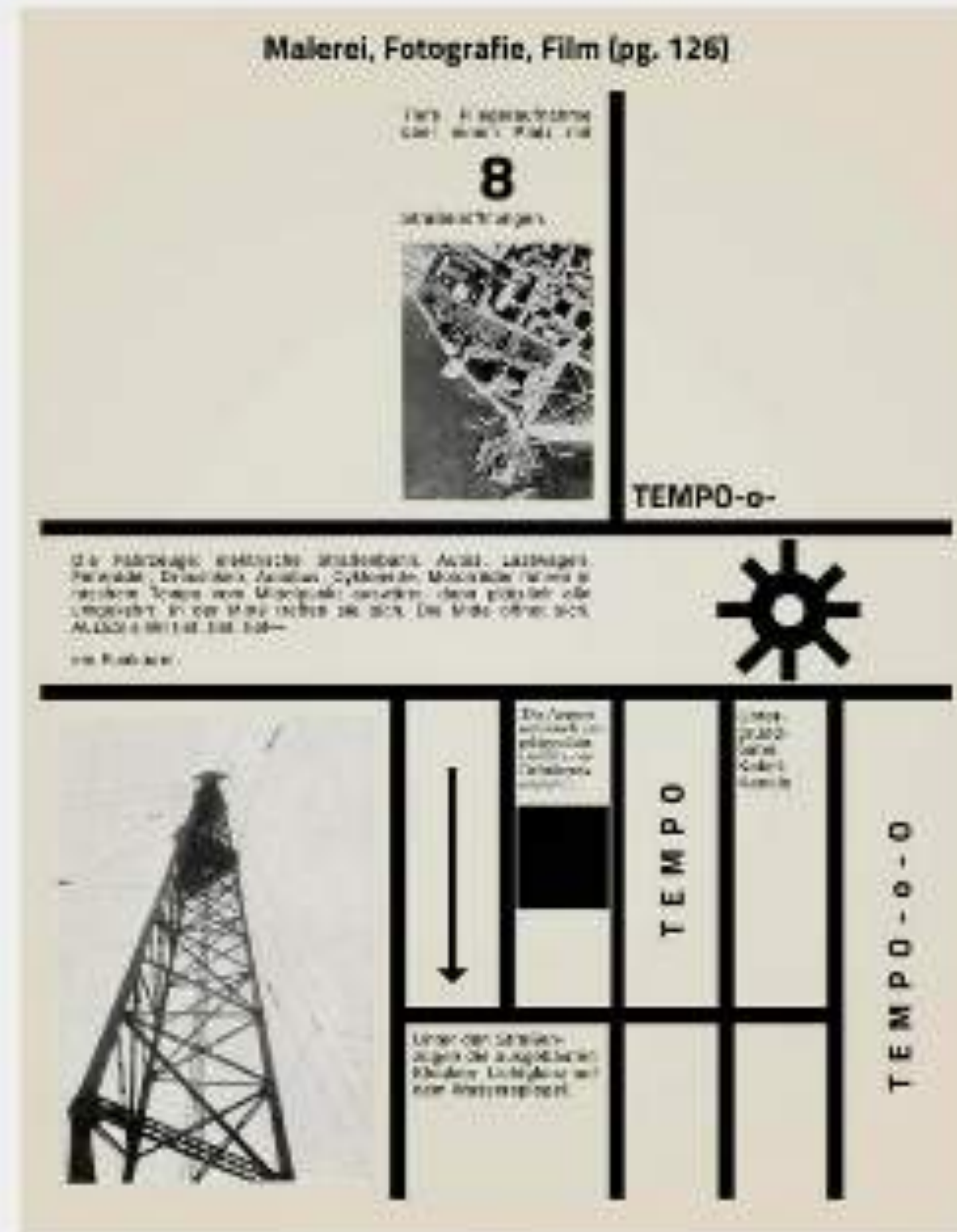
by Yuan Chuan



by Andy Barefoot



CSS grid showcase



by Chen Hui Jing



by Adrian Roworth



by Aysha
Anggraini



by Chen Hui Jing



Artist profile page



A photograph of the band Tycho performing live on stage. Two members are visible: one at a keyboard and another with a guitar. The stage is lit with blue and yellow lights, and there is smoke in the background.

Tycho

ARTIST SPOTLIGHT

10.2.2010 TORONTO
THE DRAKE HOTEL

Tycho is an American ambient music project led by Scott Hansen as primary composer, songwriter and producer. Hailing from San Francisco, California, he is known as ISO50 for his photographic and design works. His music is a combination of downtempo vintage-style synthesizers and ambient melodies.

READ MORE

0:00 / 0:09





Q&A time!

(for the rest of the questions not already covered)

- What feature would you like to add to the current spec of CSS grid?
- Did you start off as a freelancer? How difficult was it to get your first client?
- Tips for when you feel inferior about your work?
- Describe design in three words?



HJ Chen @hj_chen · Apr 24

As I continue to learn more about **CSS**, I'm ever more convinced that the best part of **CSS** is how the different properties can work together to make magic, opening up so many possibilities for web design moving forward.

CSS is a team sport!

#css #webdev #webdesign



“CSS is a team sport.”

—*Me*



Useful references

- [CSS Grid Layout Module Level 1](#)
- [Codrops CSS Grid reference](#)
- [Grid by Example](#)
- [Learn CSS Grid](#)
- [Grid Auto-Placement Is Ready](#)
- [Automatizing the Grid](#)
- [Deep Dive into Grid Layout Placement](#)
- [CSS Grid Layout and positioned items](#)
- [CSS Logical Properties and Values in Chromium and WebKit](#)
- [Changes on CSS Grid Layout in percentages and indefinite height](#)
- [The Story of CSS Grid, from Its Creators](#)
- [CSS Grid Layout is Here to Stay](#)
- [The New Layout Standard For The Web: CSS Grid, Flexbox And Box Alignment](#)
- [What Happens When You Create A Flexbox Flex Container?](#)
- [Everything You Need To Know About Alignment In Flexbox](#)
- [Grid “fallbacks” and overrides](#)

Salamat!



<https://www.chenhuijing.com>



@hj_chen



@hj_chen



@huijing

Font used is **Prospectus**, by Dave Bailey

