LAYOUT THE WEB WITH CSS GRID

and the rest of team layout

#TeamWeb

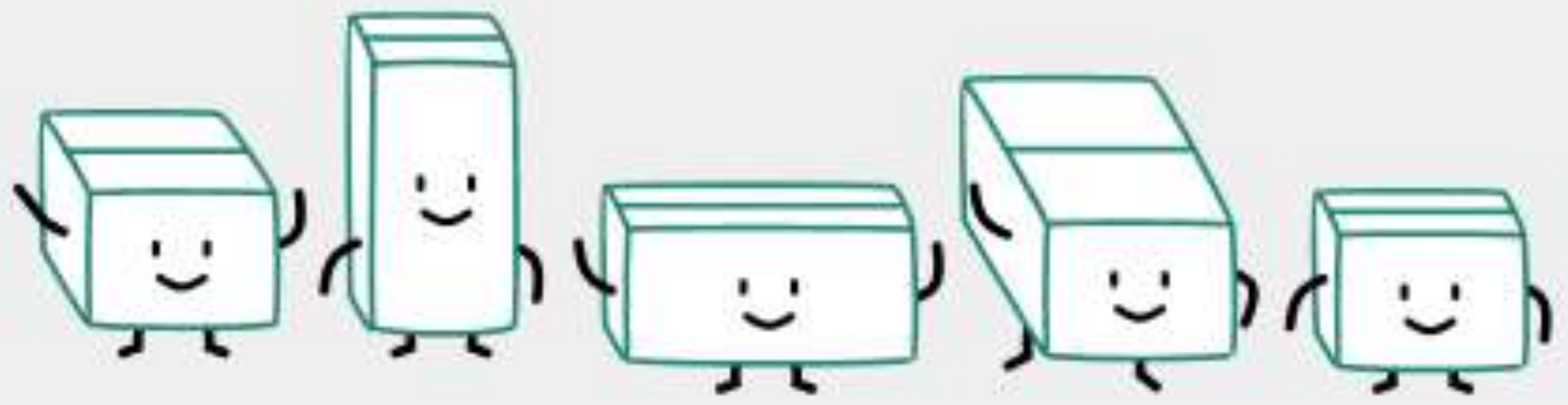
This workshop has been inadvertently brought to you by Philippine Airlines.

Chen Hui Jing

@hj_chen

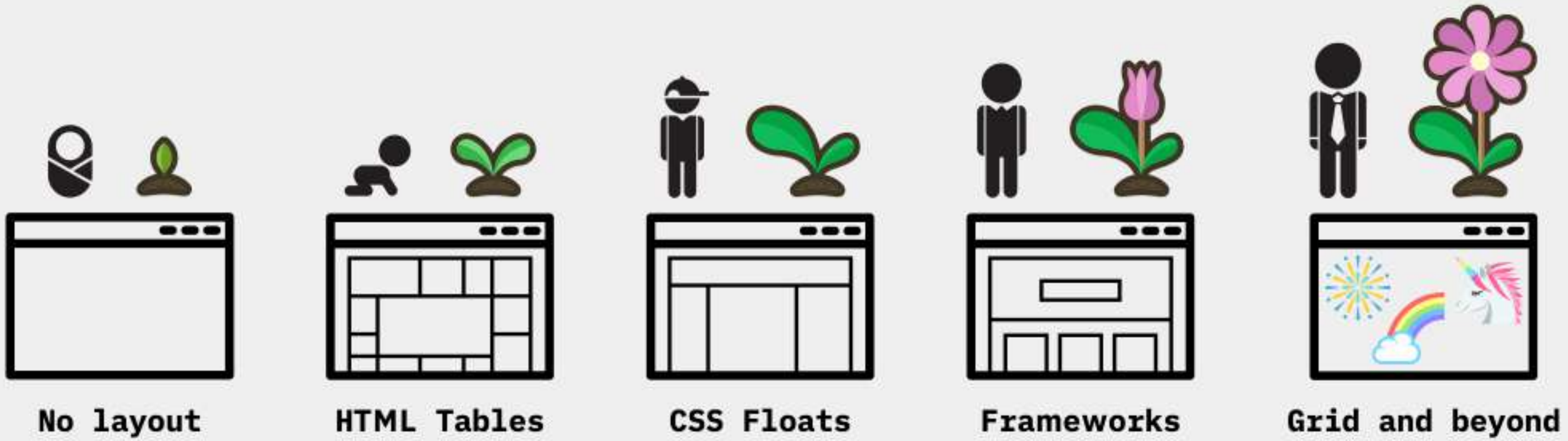
Starter files

http://bit.ly/ffc9-layout



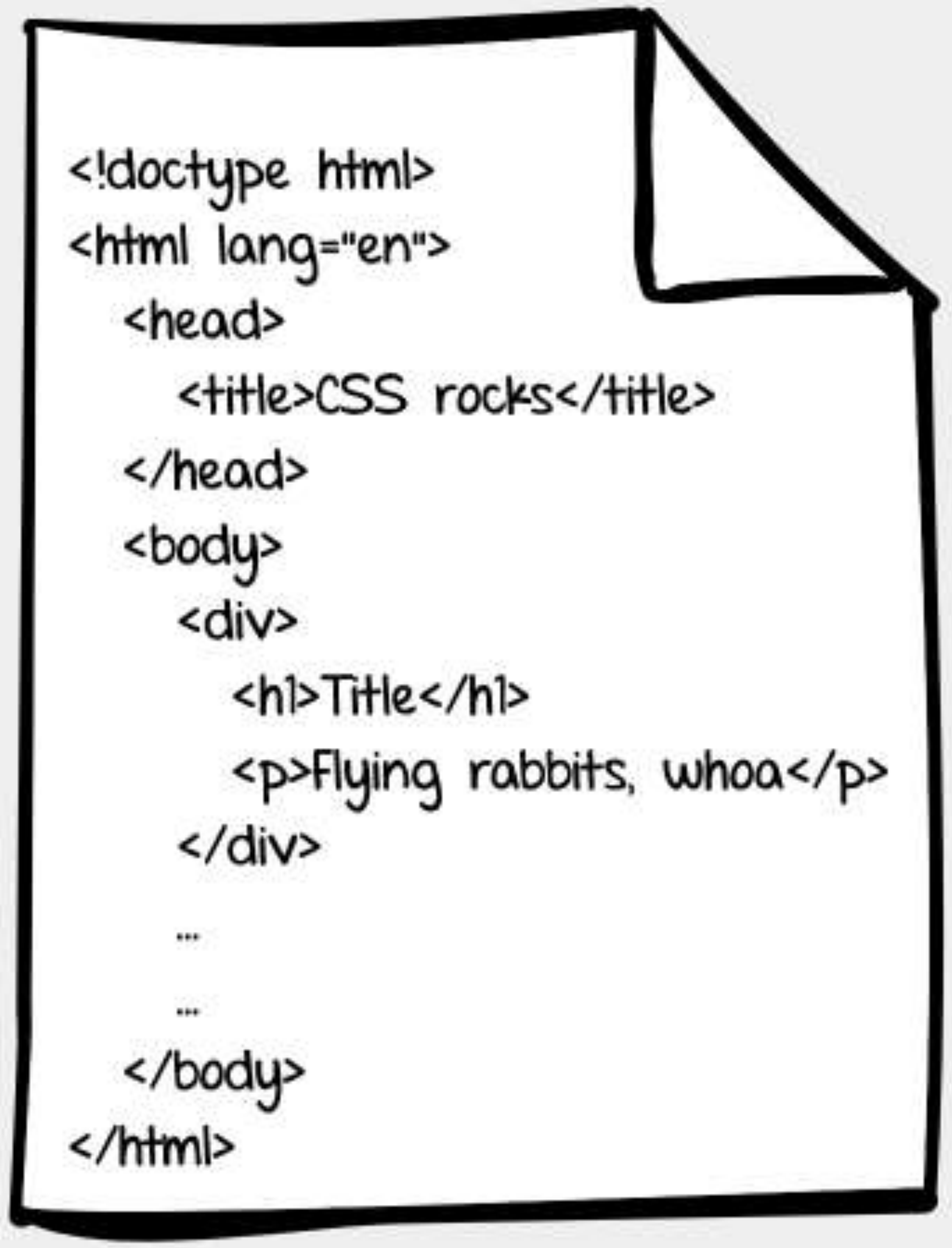
A

Evolution of browsers

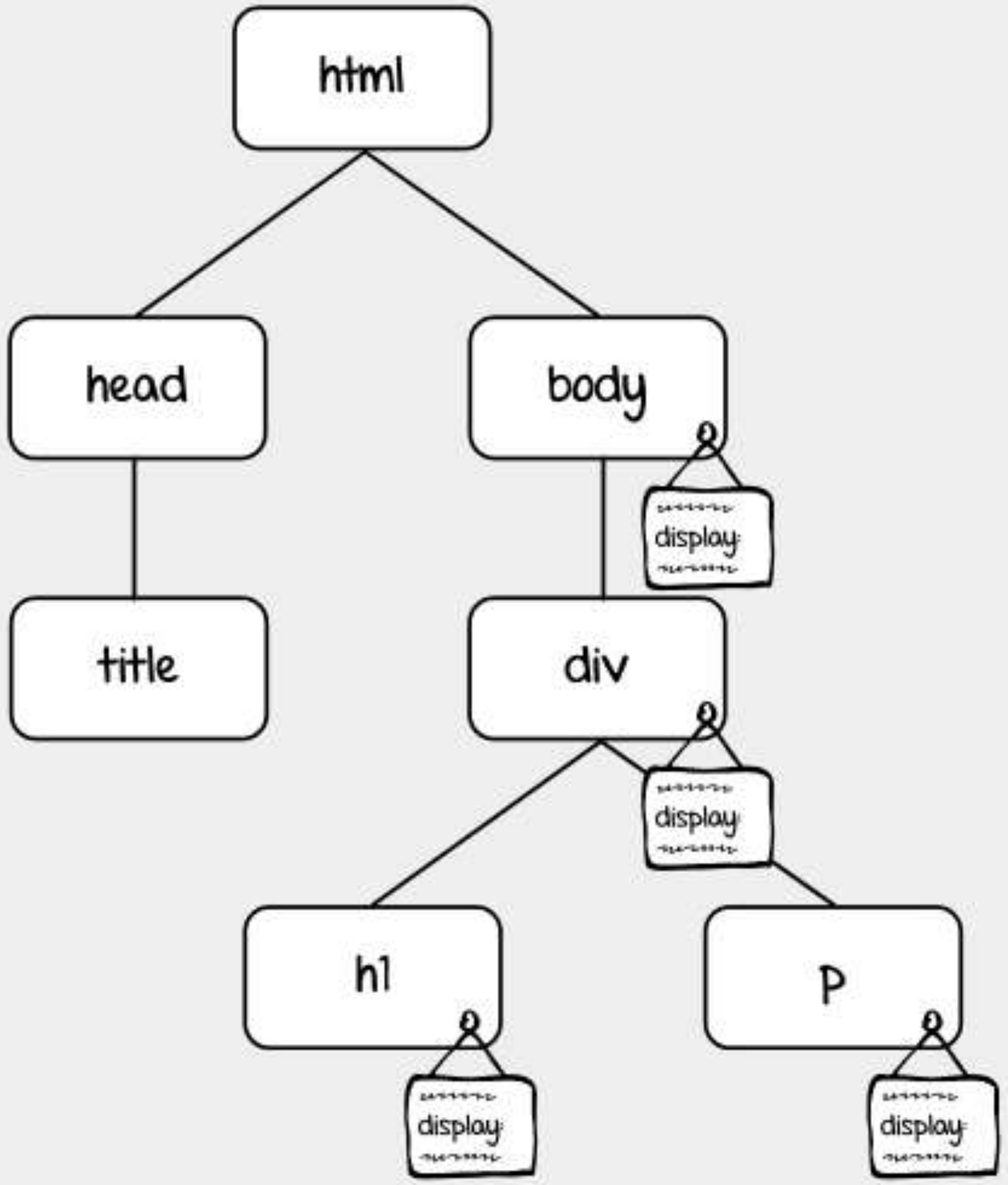


HOW BROWSERS RENDER STUFF

(and lay them out)



```
<!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

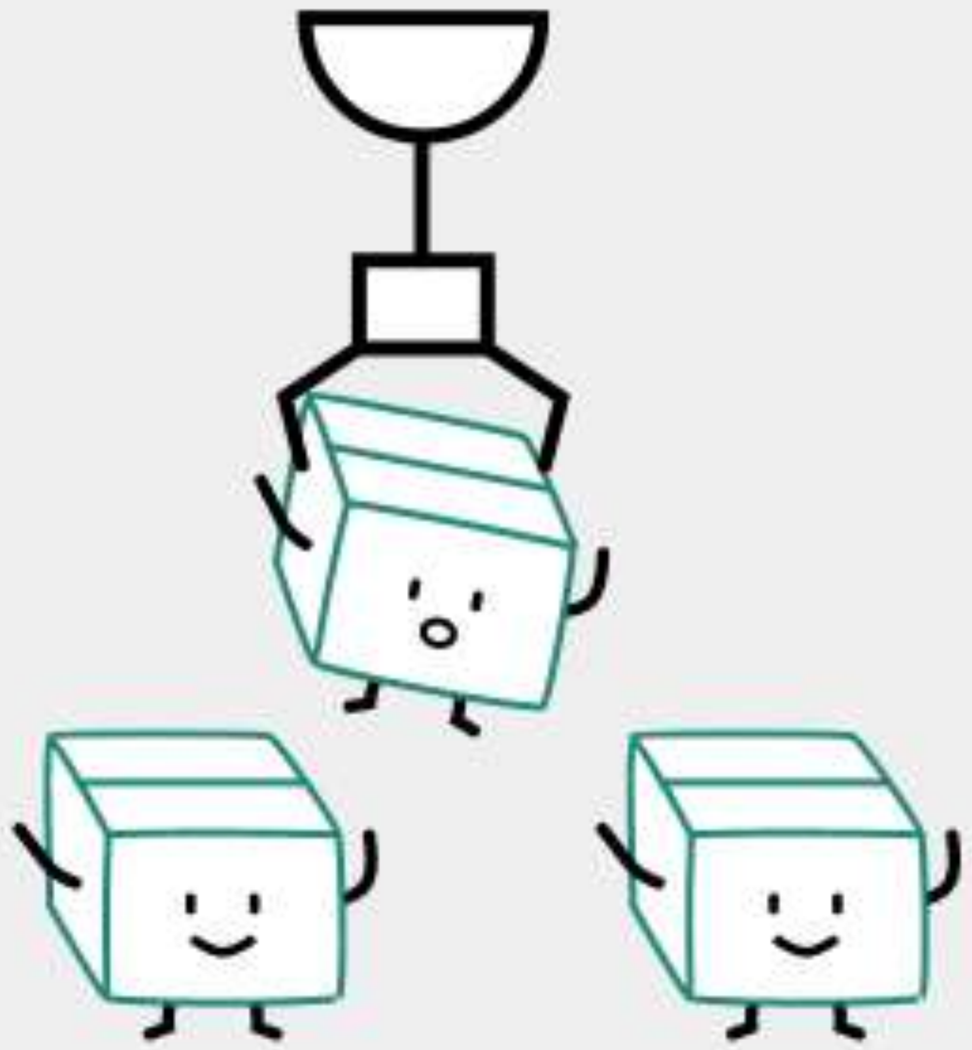
📦 Relationships between elements in the document tree

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

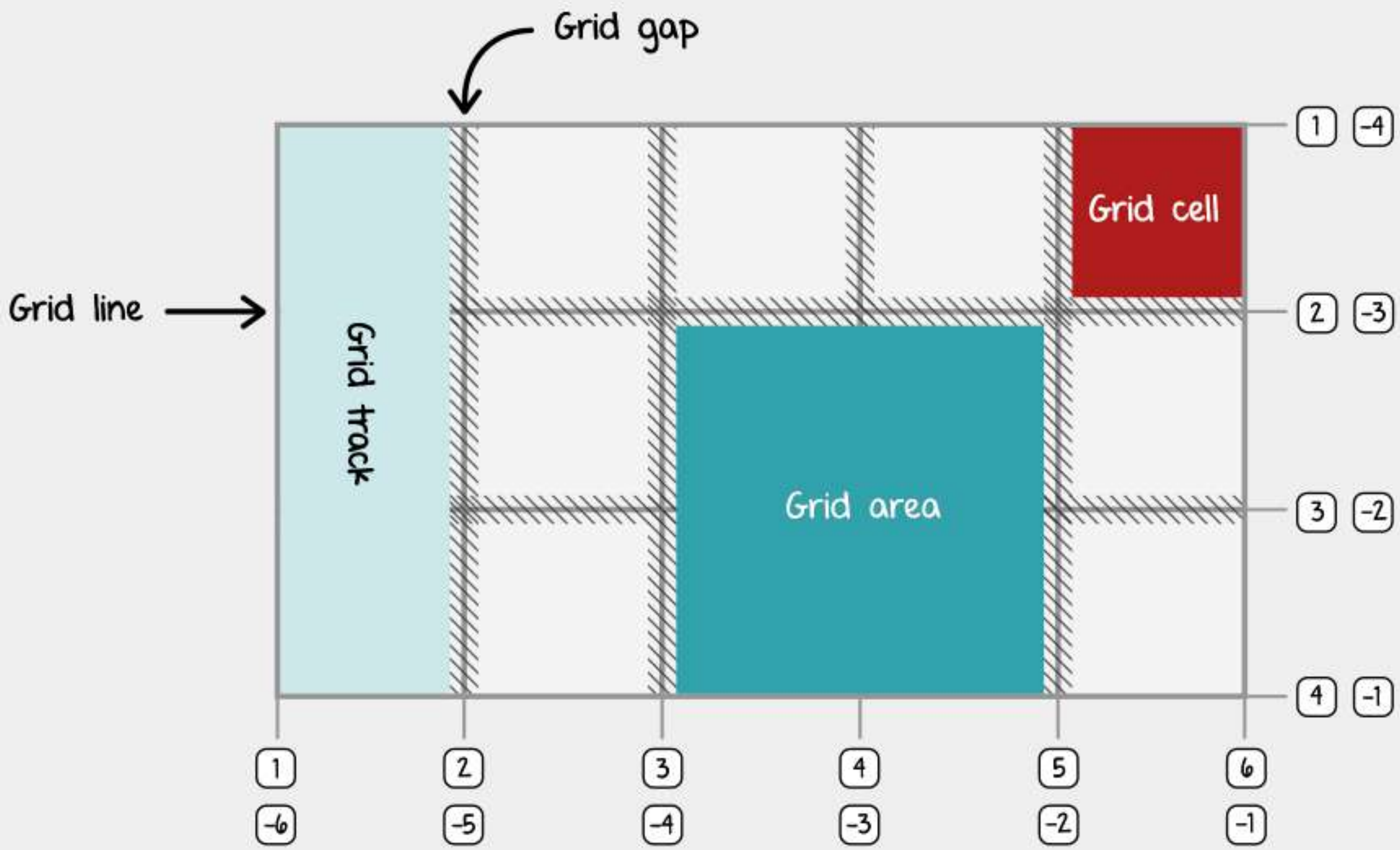
Positioning schemes

Normal flow

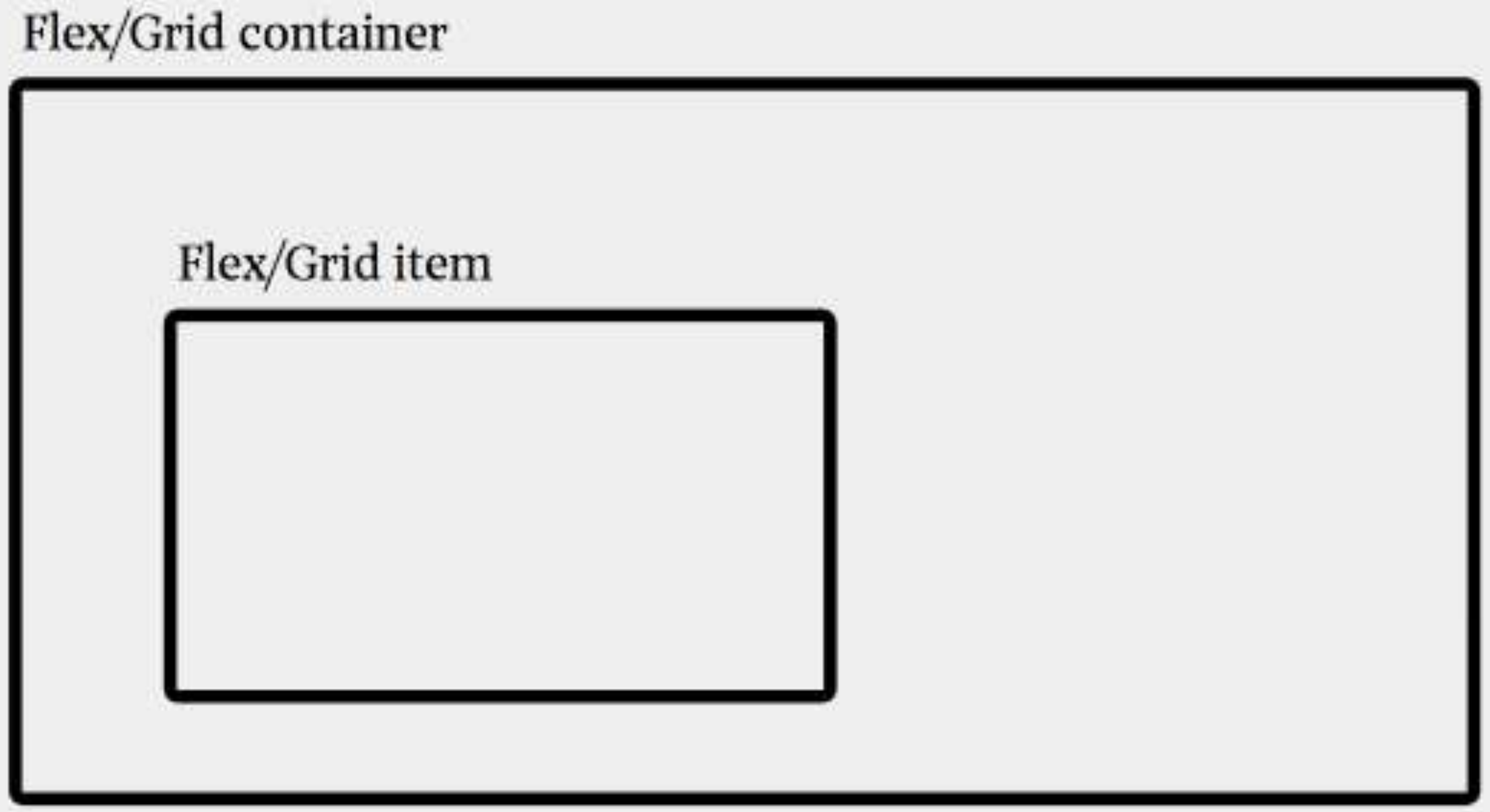
Floats


Absolute positioning

Basic terminology



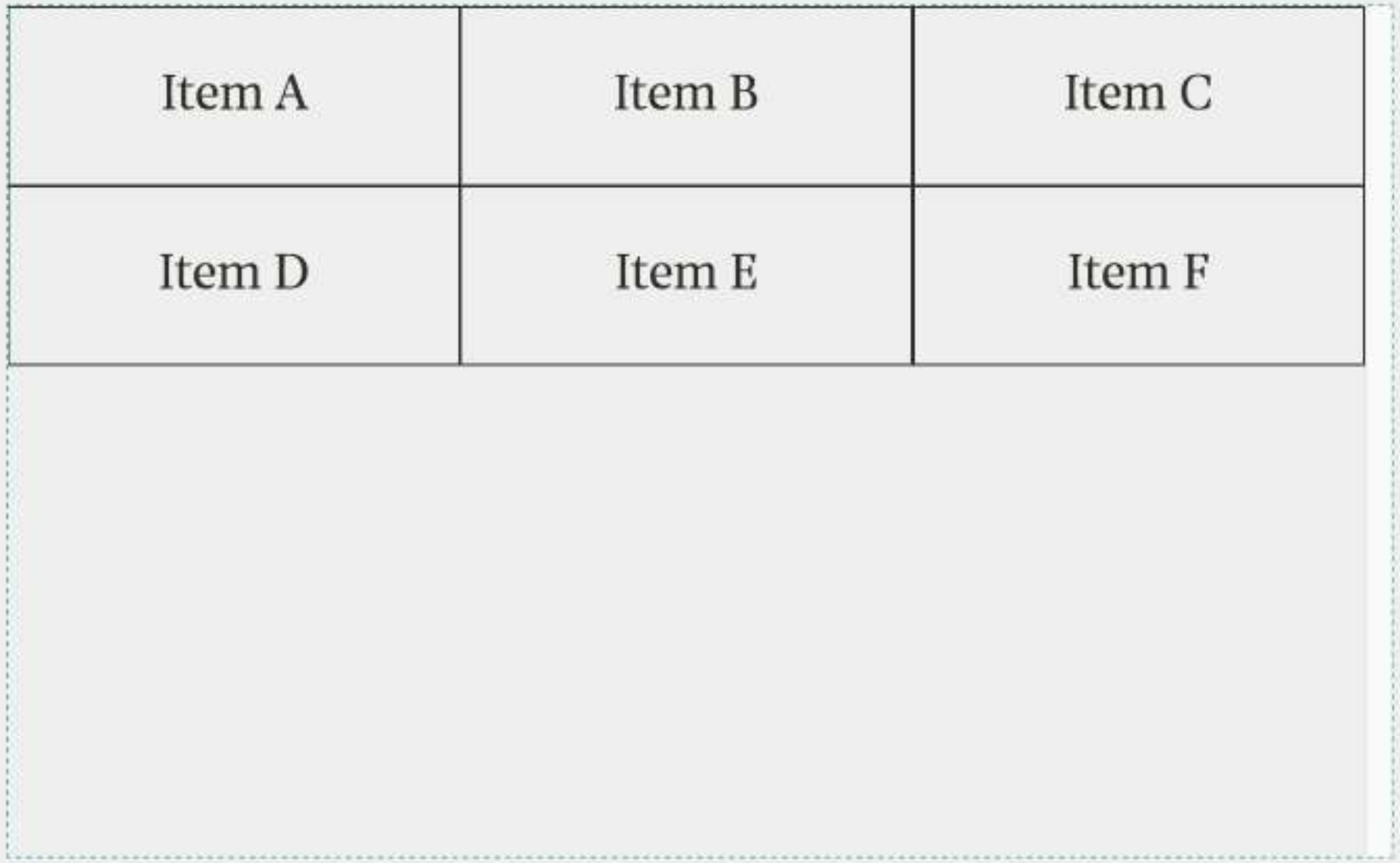
The container-child relationship



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

—Rachel Andrew

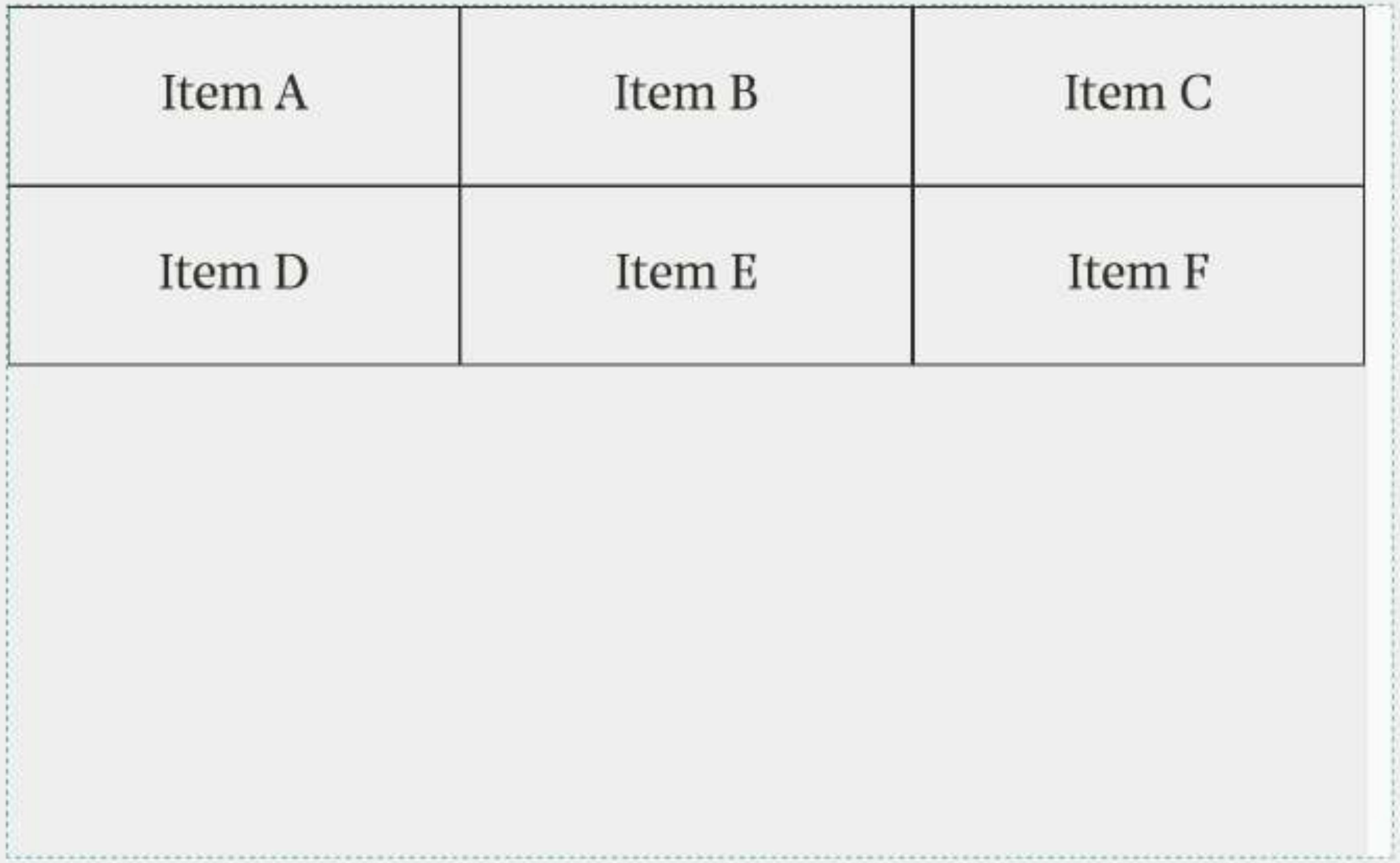
Layout technique: `inline-block`



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

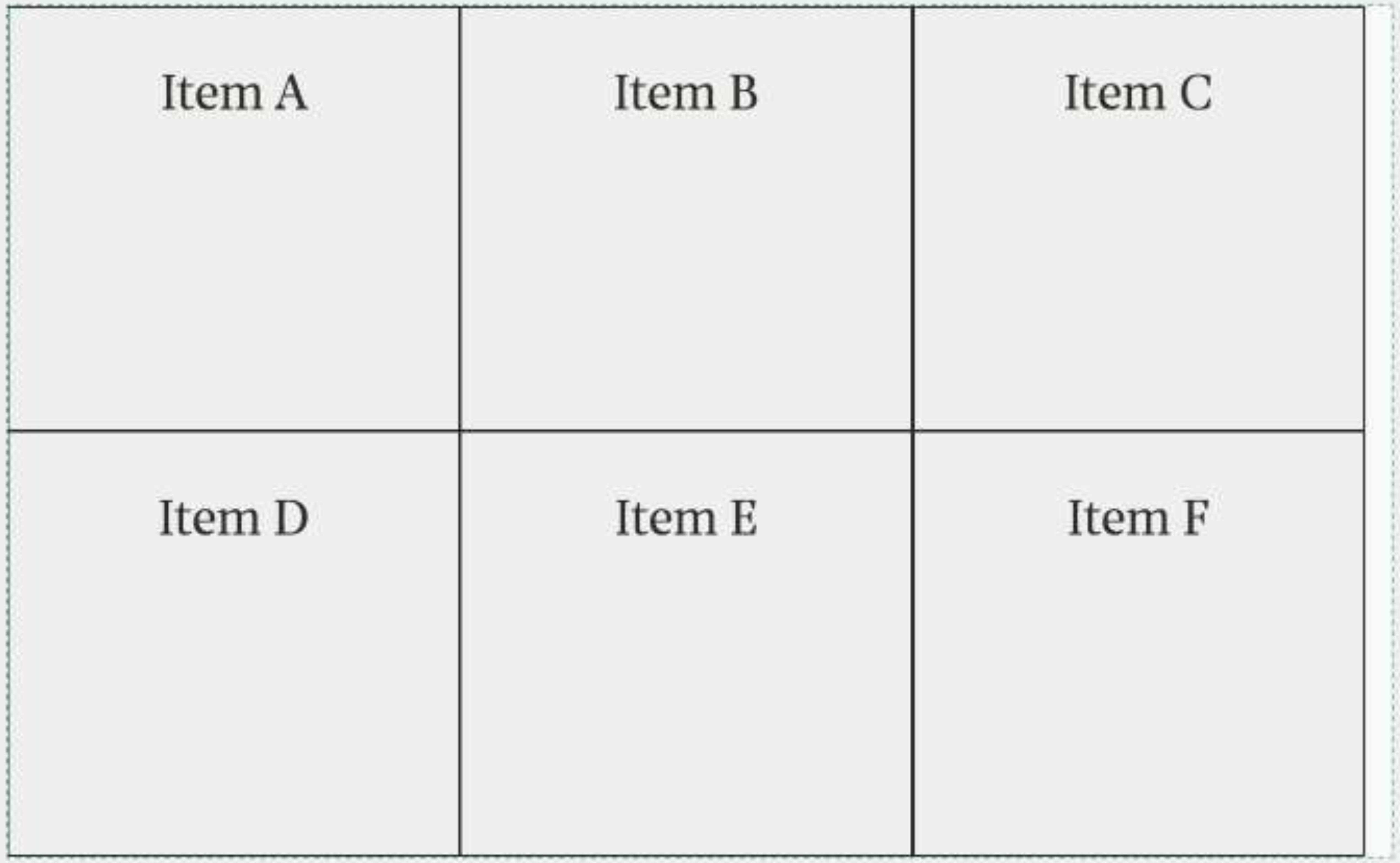
```
.inlineblock__item {
  display: inline-block;
  width: calc(100% / 3);
}
```

Layout technique: `float`



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

Layout technique: `flex`



```
.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.”*

1
2
3
4
5
6
7
8
9
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
-6
-5
-4
-3
-2
-1

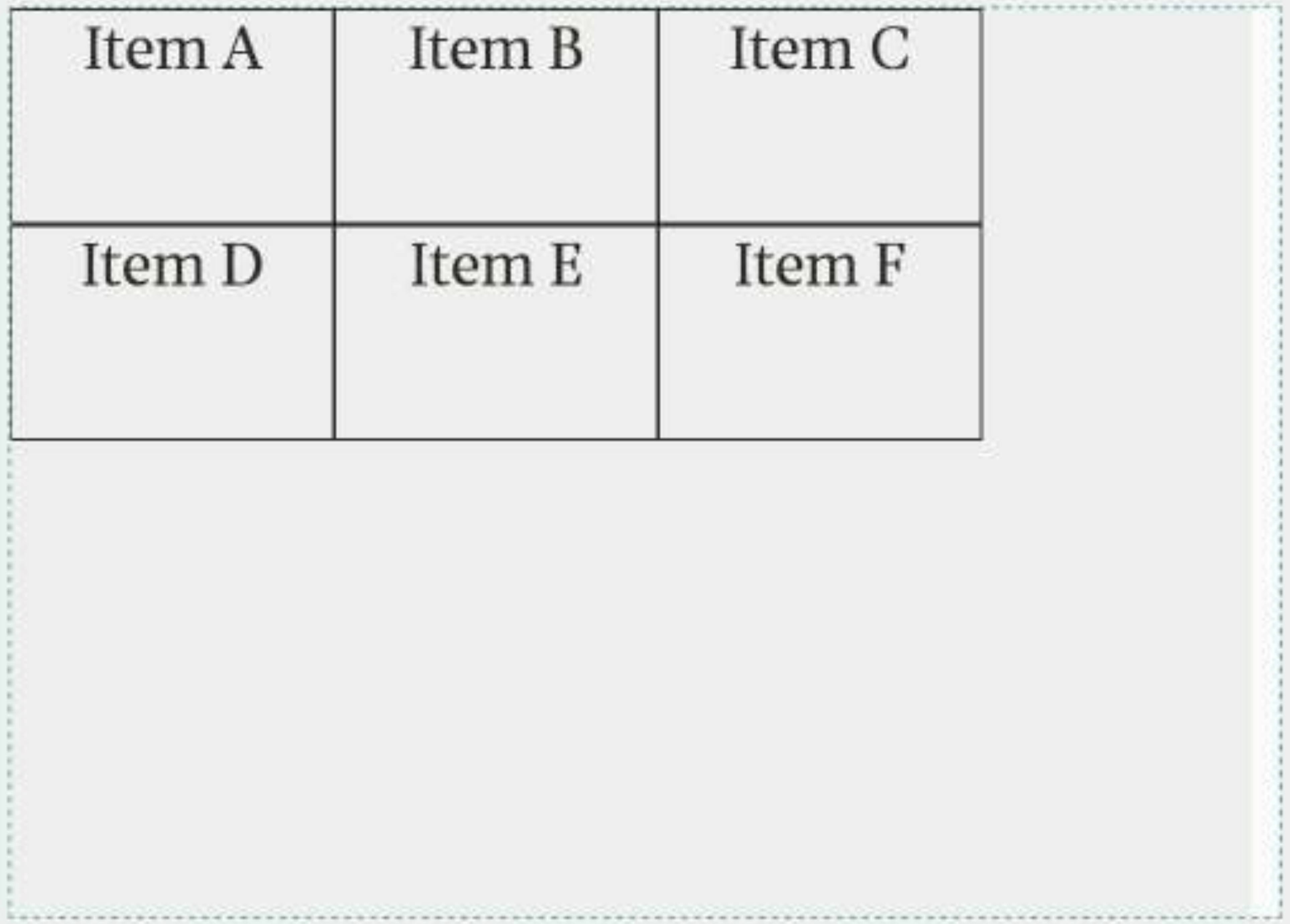
Firefox Grid Inspector

THE MOST BASIC GRID

Defining a grid

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

```html
<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>
```



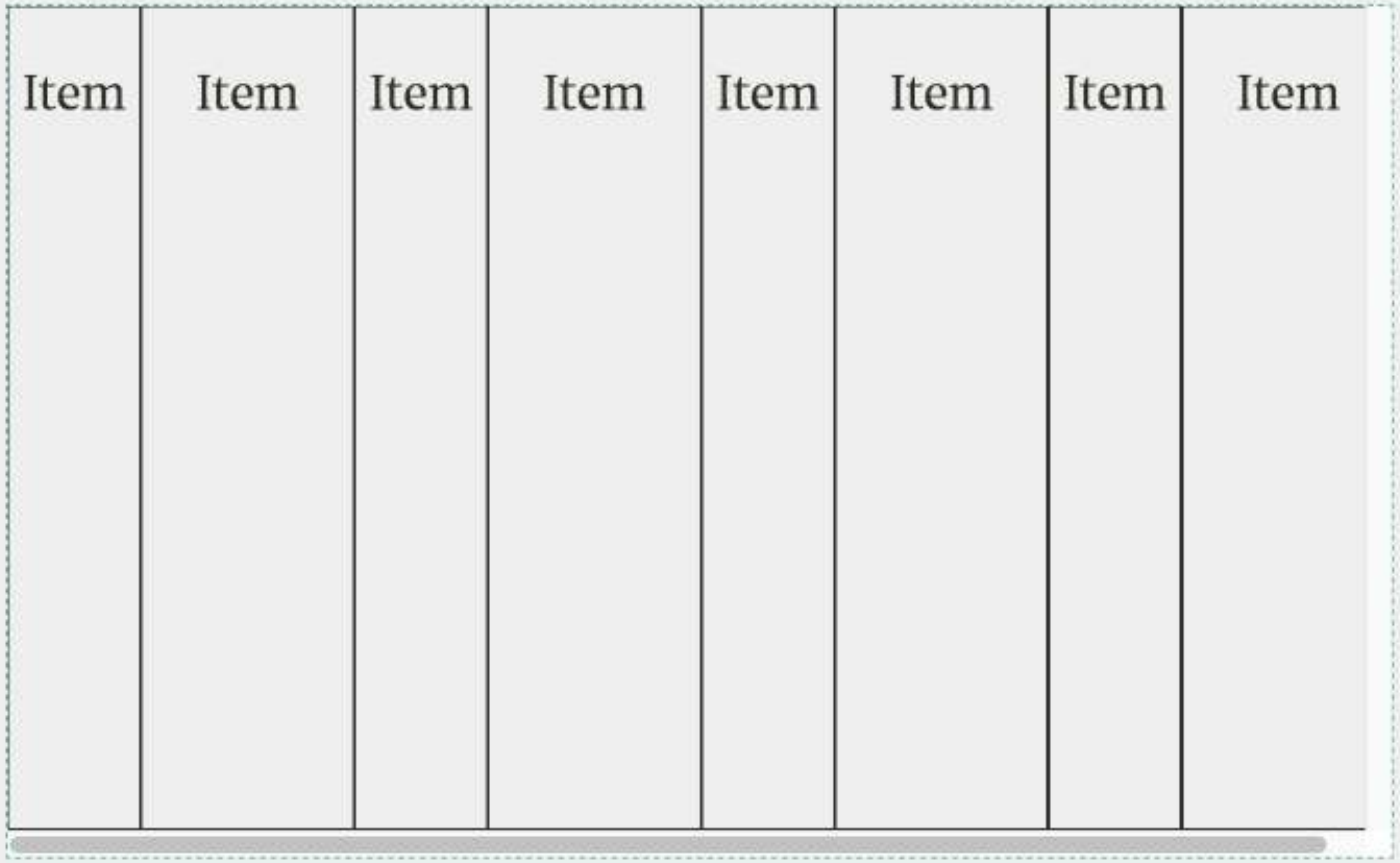
```css
.grid1 {
  display: grid;
  grid-template-columns: 150px 150px
150px;
  grid-template-rows: 100px 100px;
}
```

AUTO-PLACEMENT OF GRID ITEMS

(and the implicit grid)

The `repeat()` function

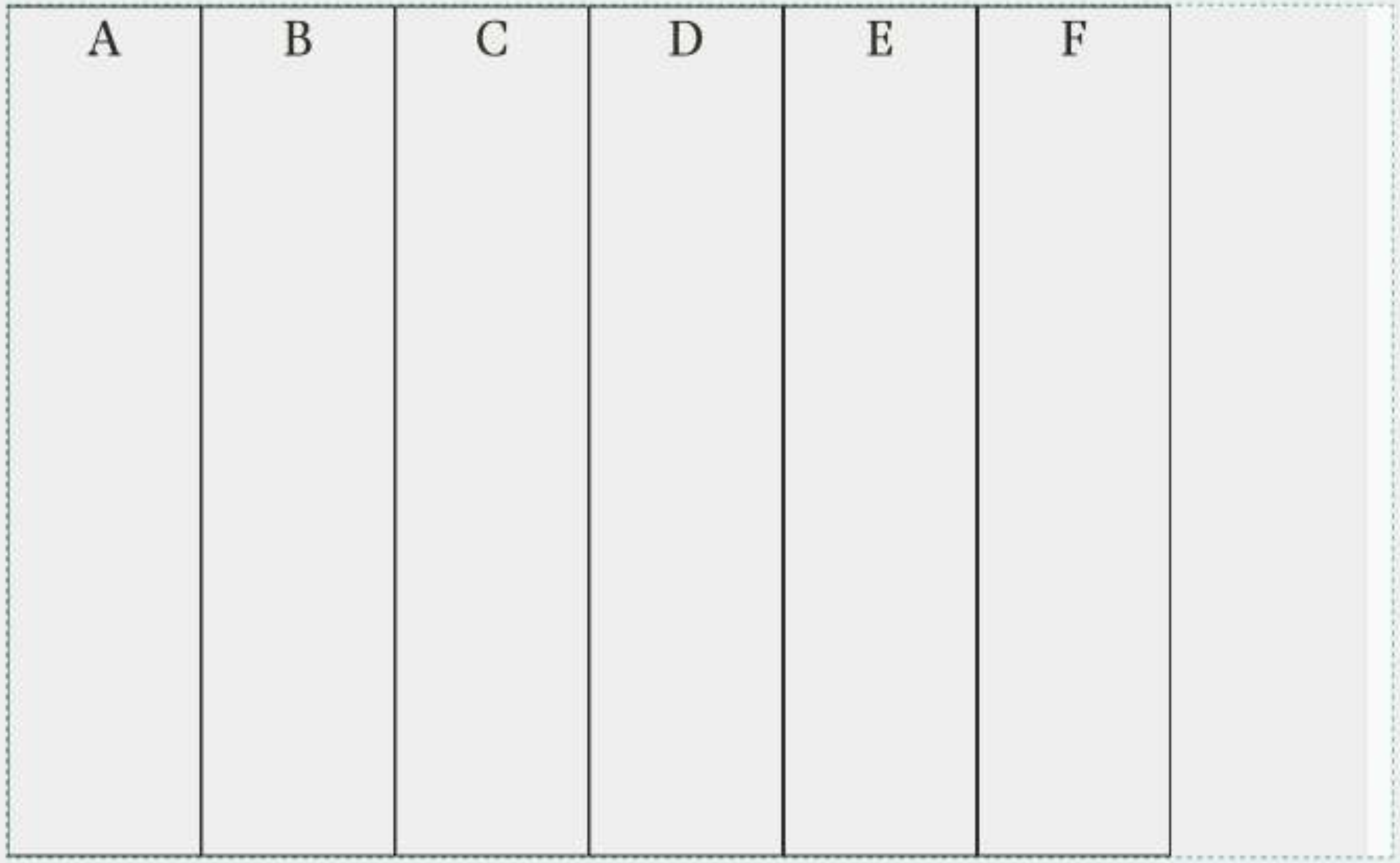
To specify a large number of columns or rows that follow a similar pattern



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

`auto-fill` versus `auto-fit`

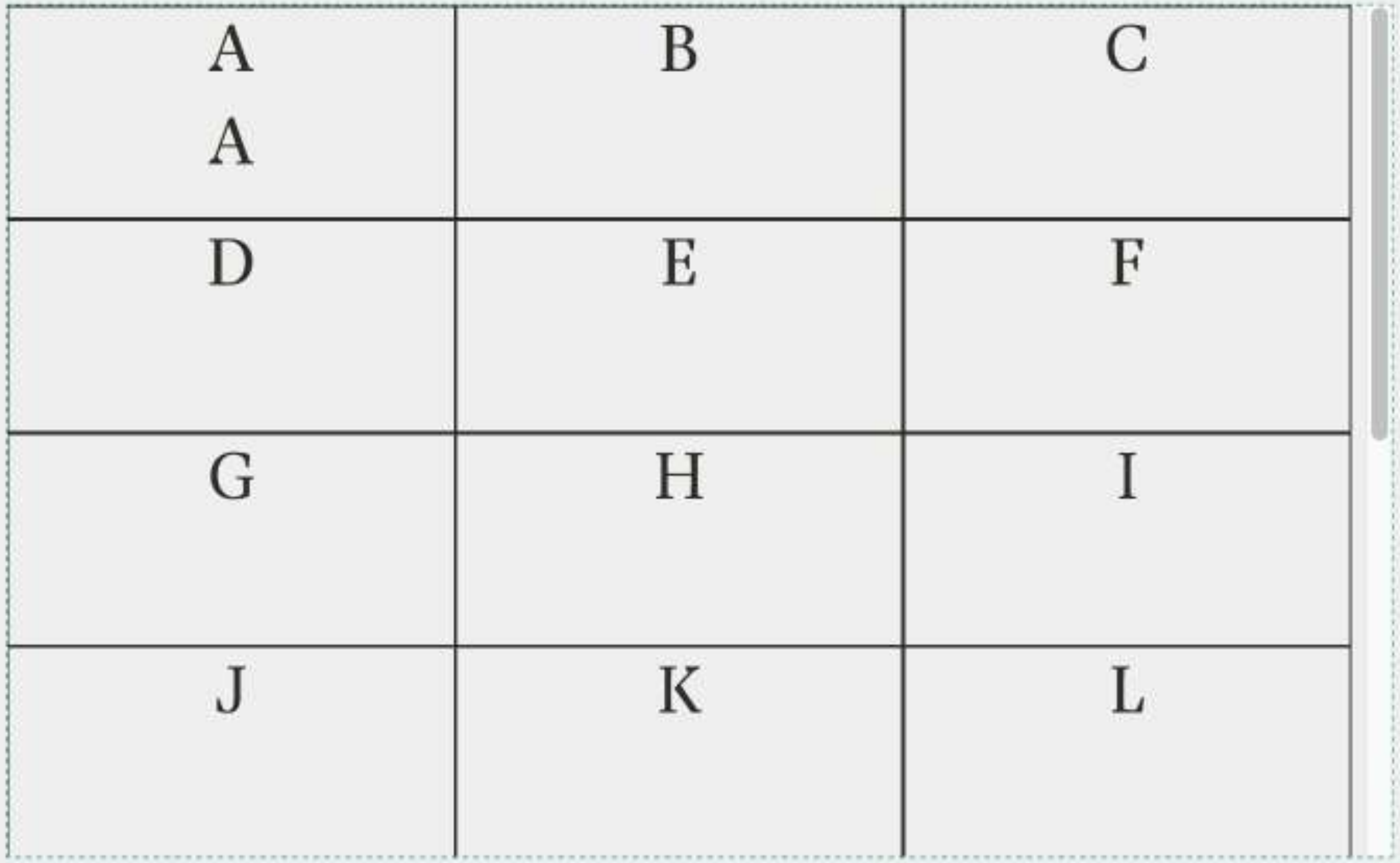
Letting the browser figure out the math



```
.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

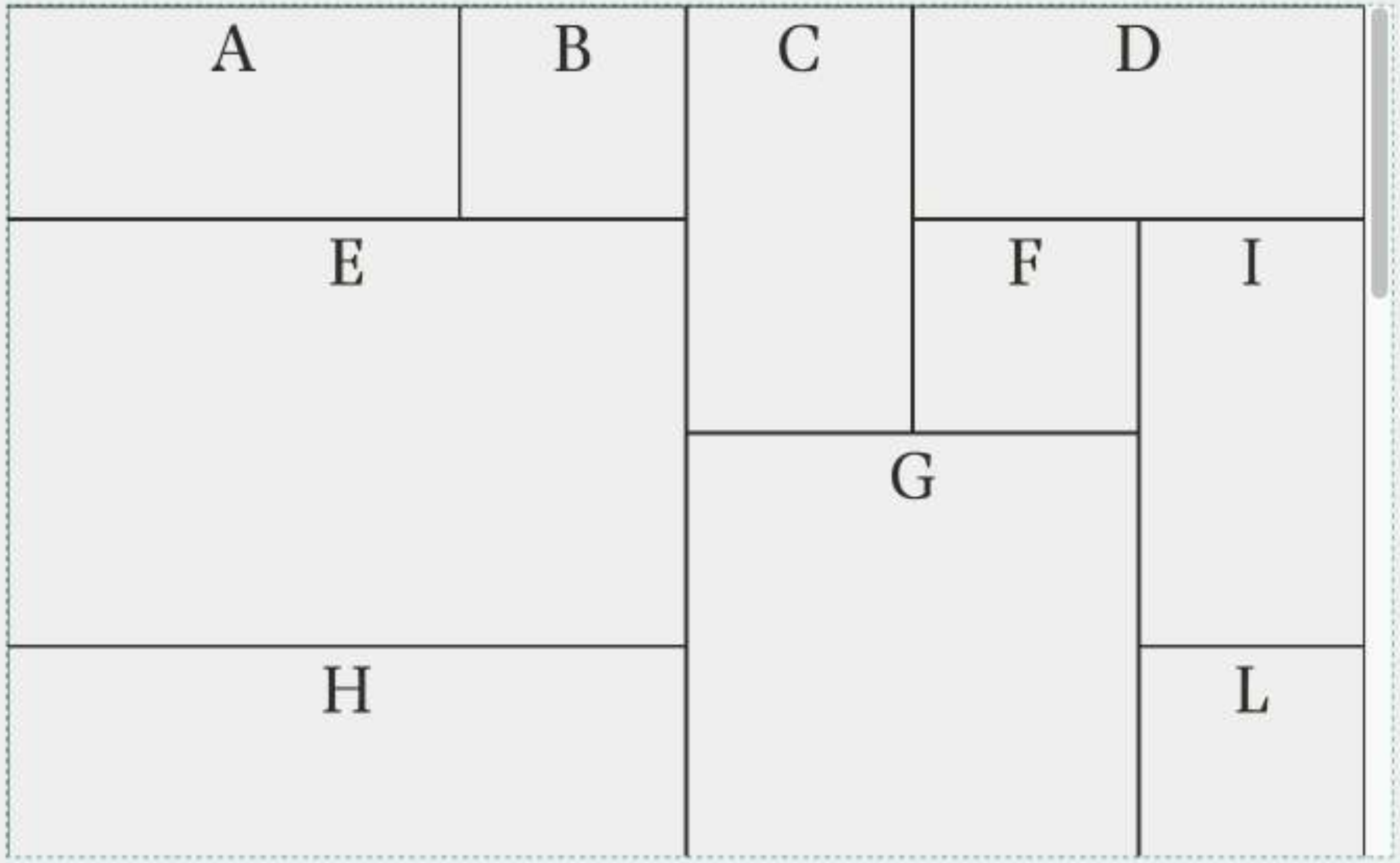


A A	B	C
D	E	F
G	H	I
J	K	L

```
.grid3 {
  display: grid;
  grid-template-columns:
33% 33% 33%;
  grid-auto-rows: 120px;
}
```

The `grid-auto-flow` property

Adjusting the direction and density of grid items

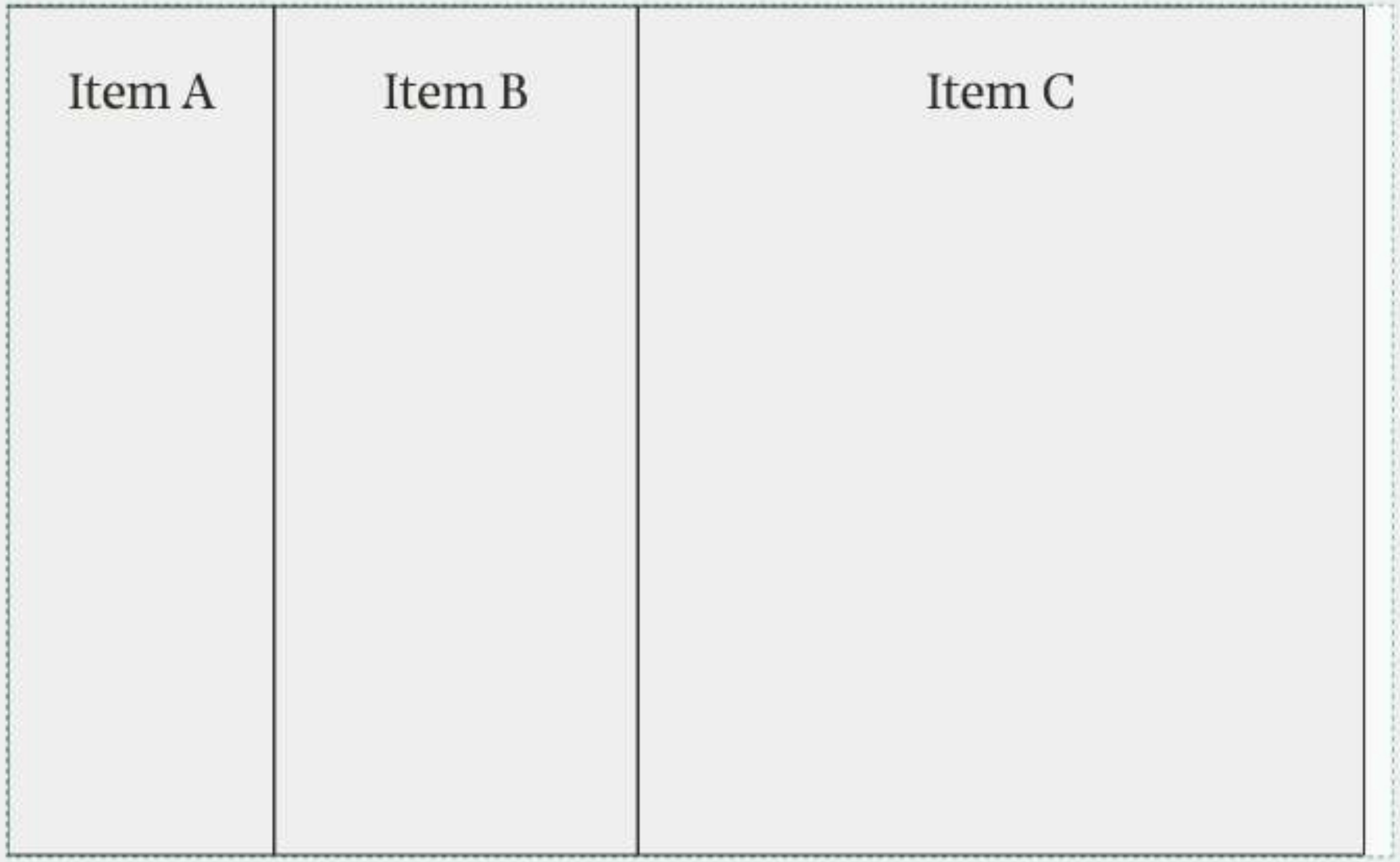


```
.grid4 {
  display: grid;
  grid-template-columns:
repeat(auto-fit,
minmax(120px, 1fr));
  grid-auto-rows: 120px;
  grid-auto-flow: dense;
}

.grid4__item:nth-
child(3n+1) {
  grid-column-end: span
2;
}
```

MORE WAYS TO SIZE GRID ROWS AND COLUMNS

The `fr` unit

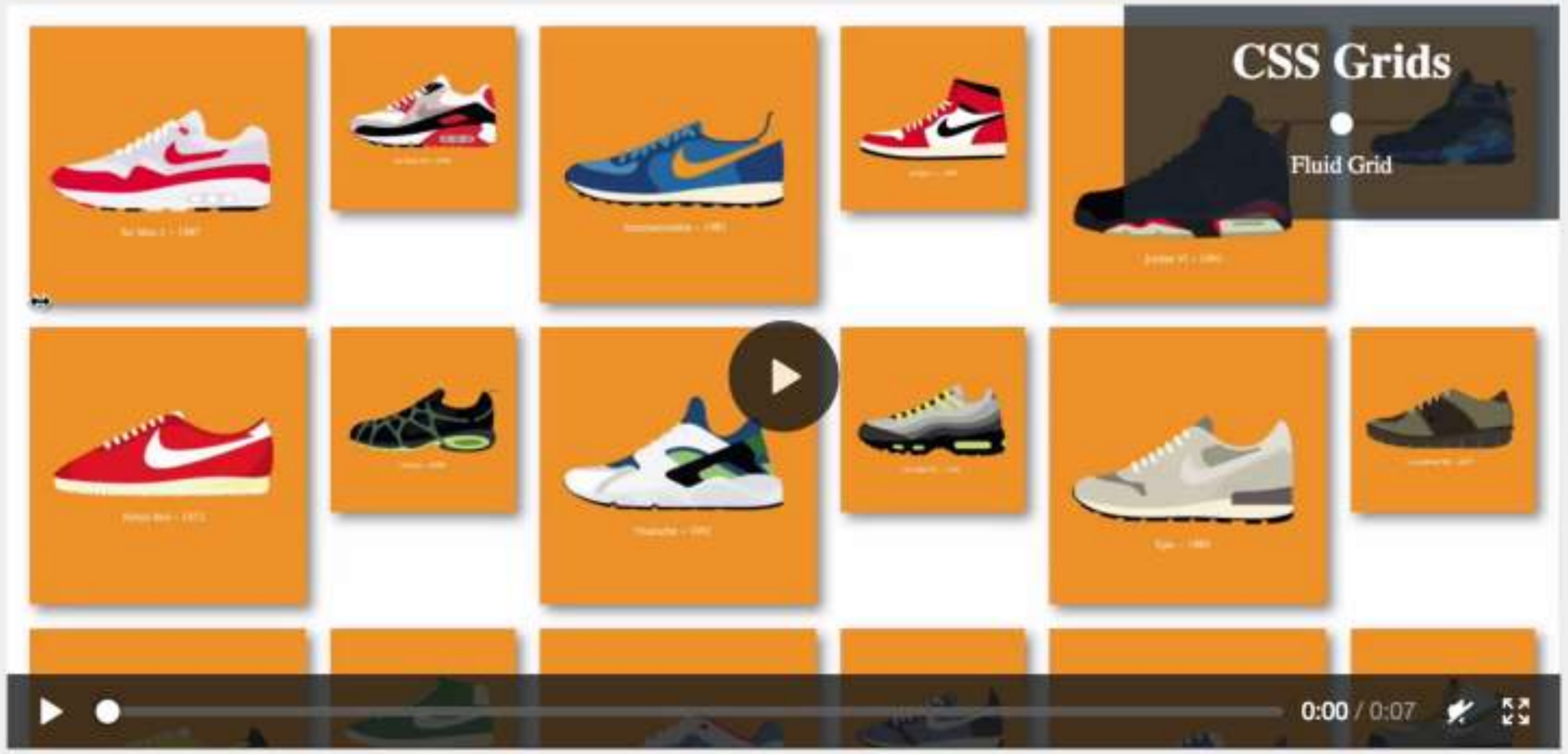
Represents a **fraction** of the **free space** in the grid container.



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

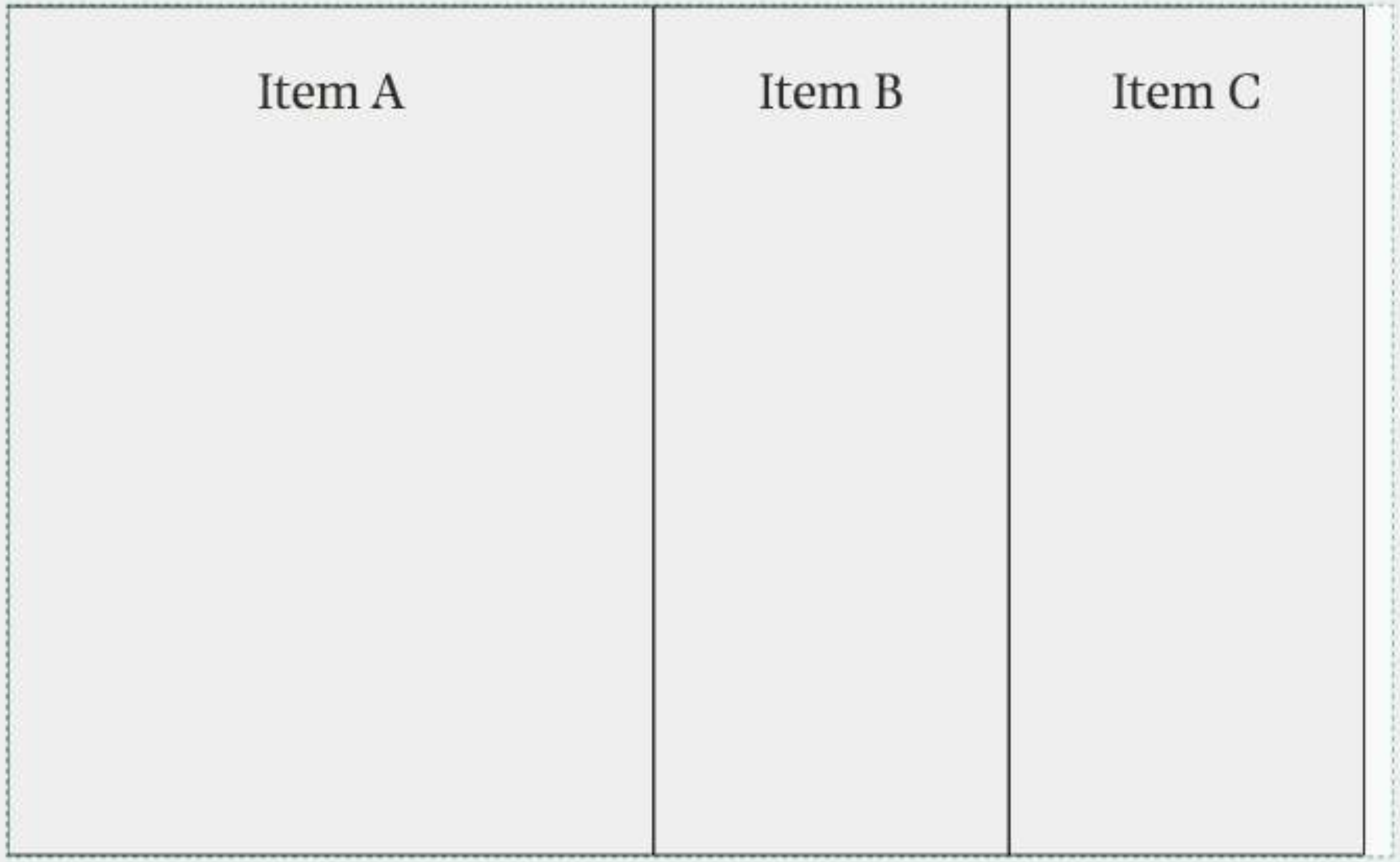
Fluid CSS grid

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



The `minmax()` function

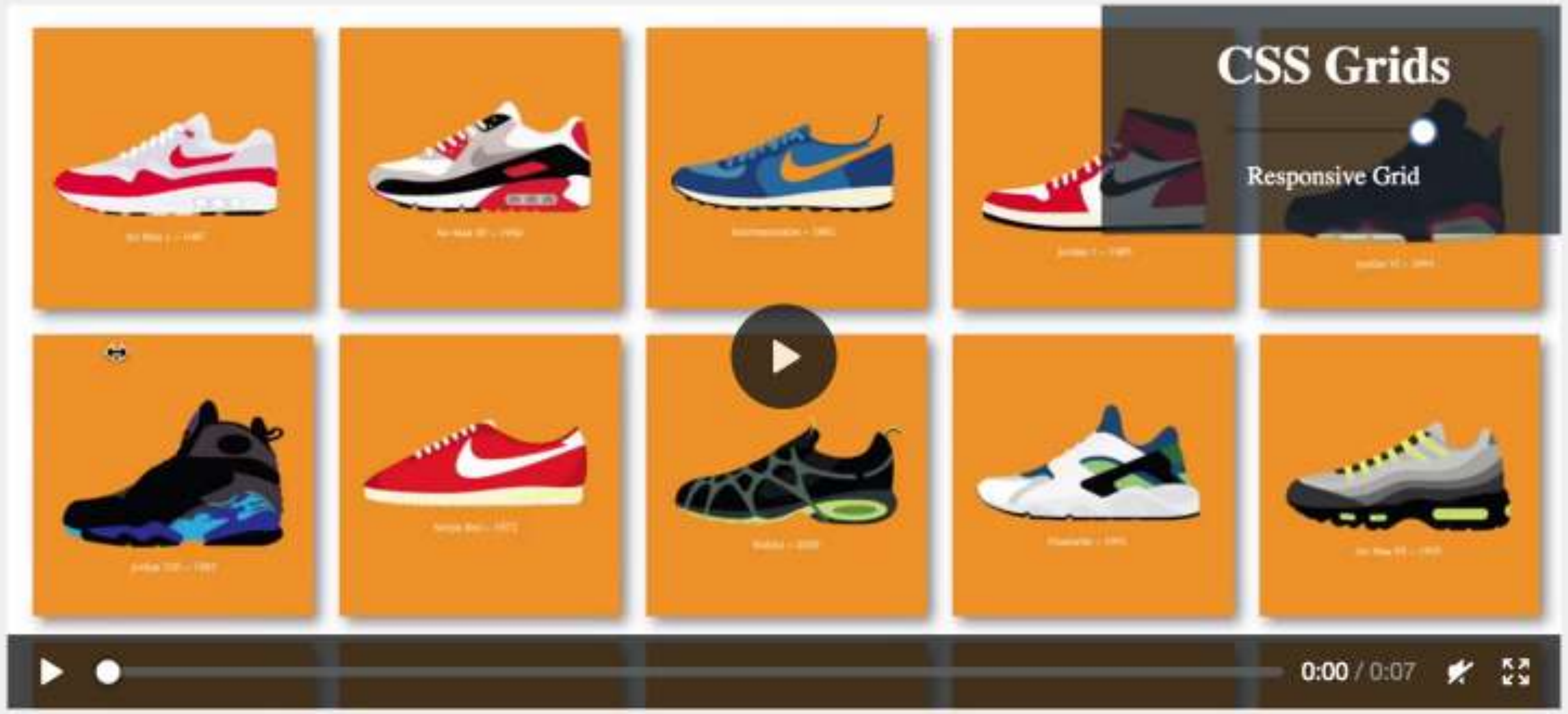
Defines a size range for columns or rows in the grid.



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

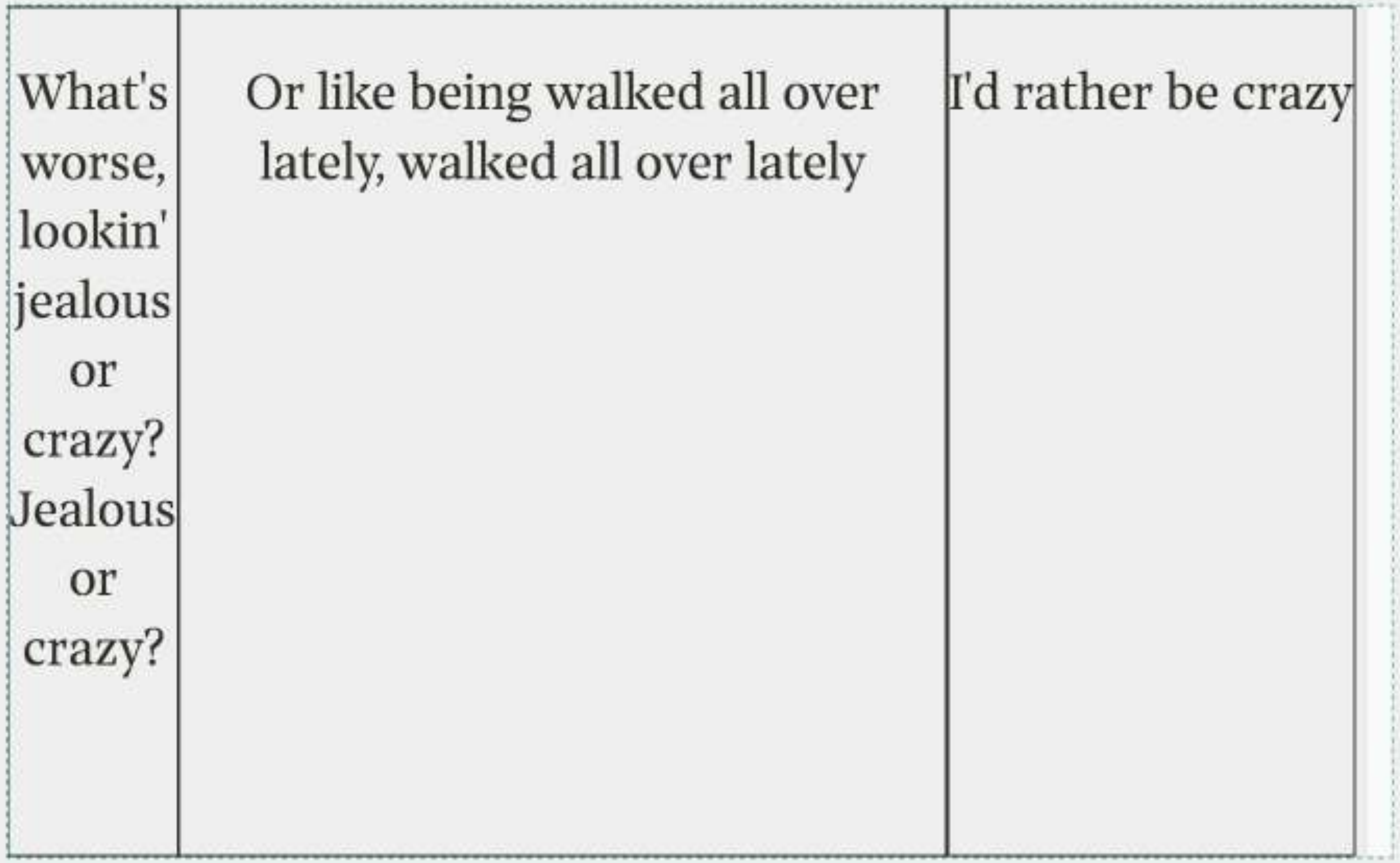
Responsive grid without media queries

```
.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

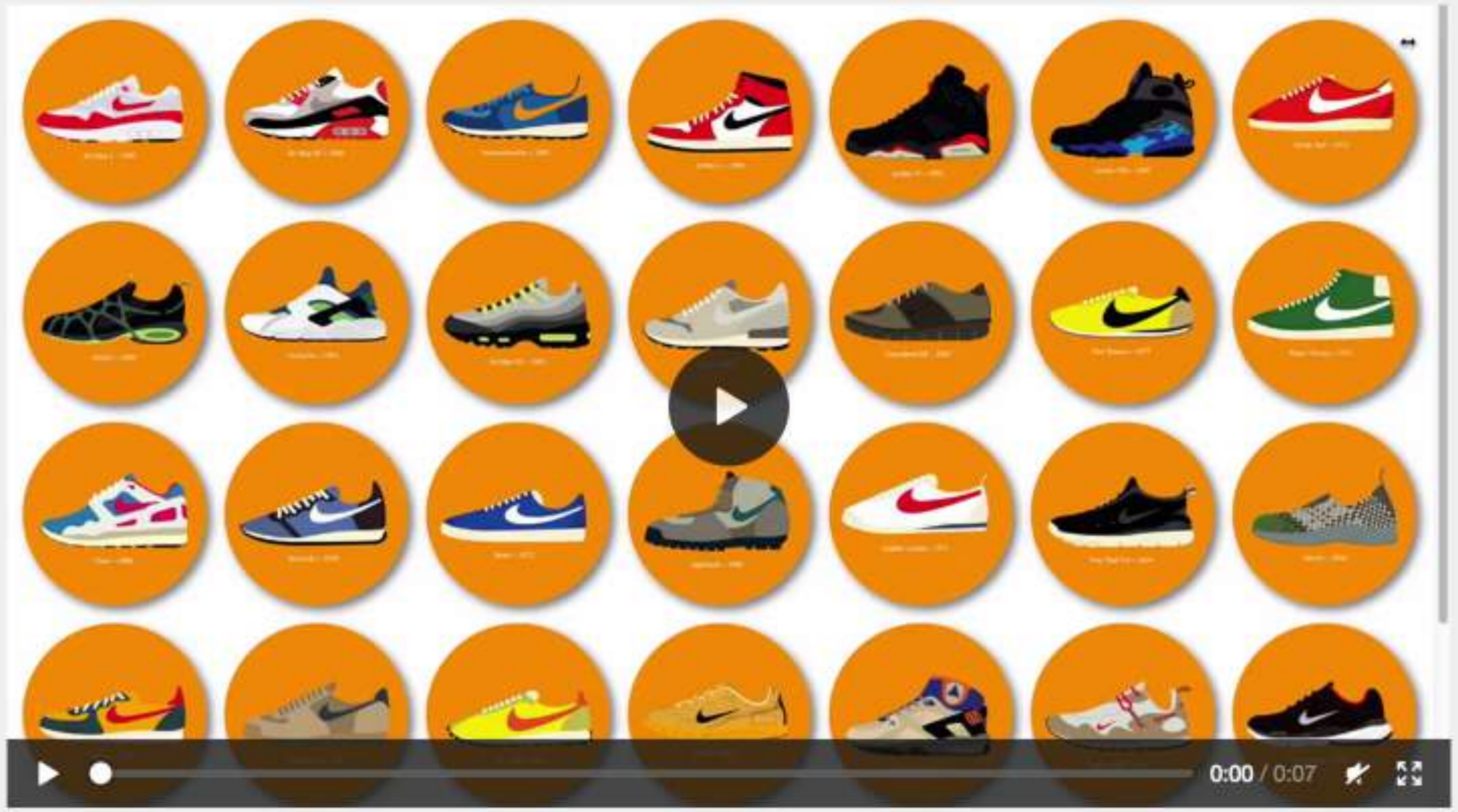

What's worse, lookin' jealous or crazy? Jealous or crazy?

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

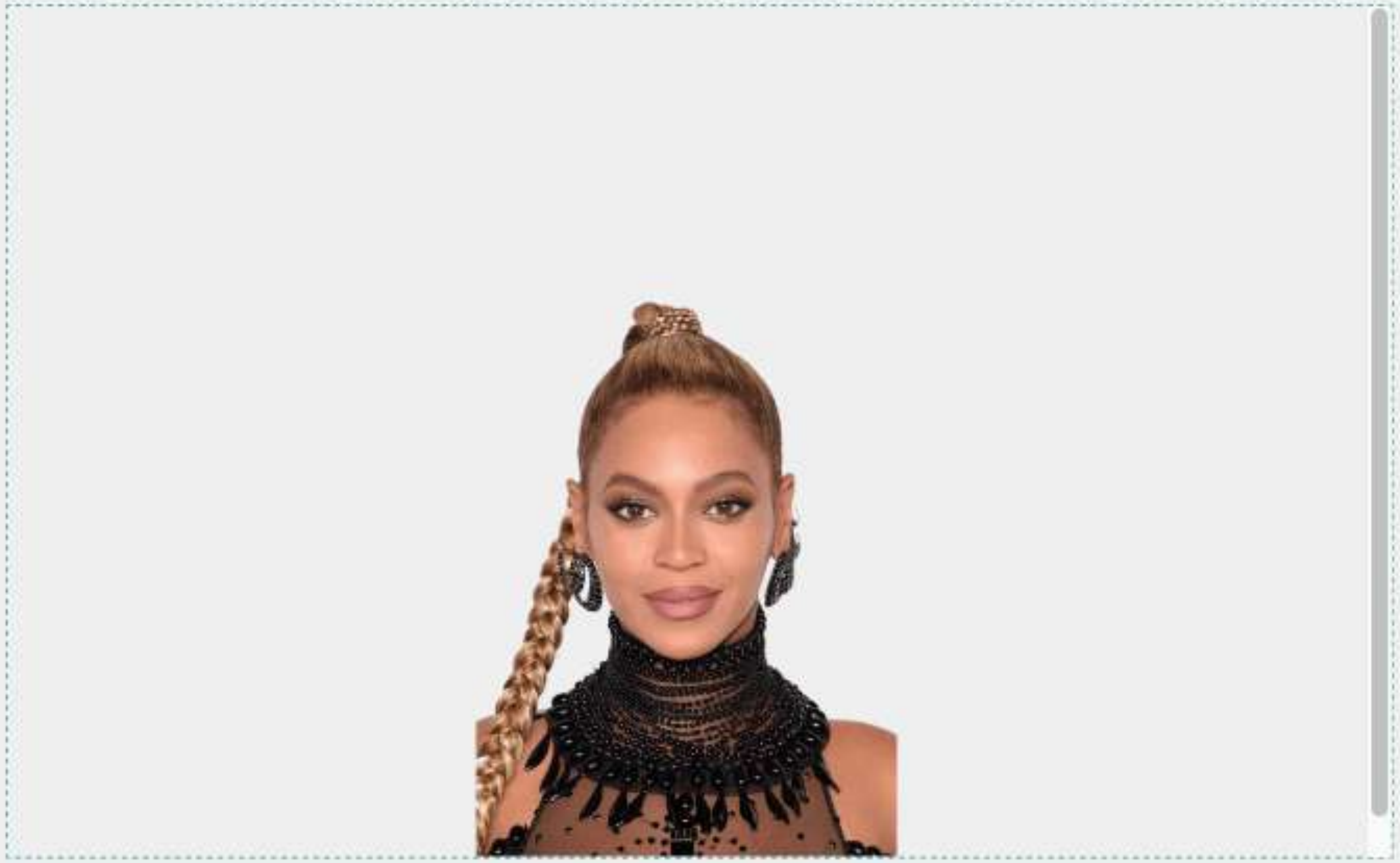


MANUAL PLACEMENT OF GRID ITEMS

(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-row-start: 2;
  grid-row-end: 4;
  grid-column-start: 2;
  grid-column-end: 3;
}
```

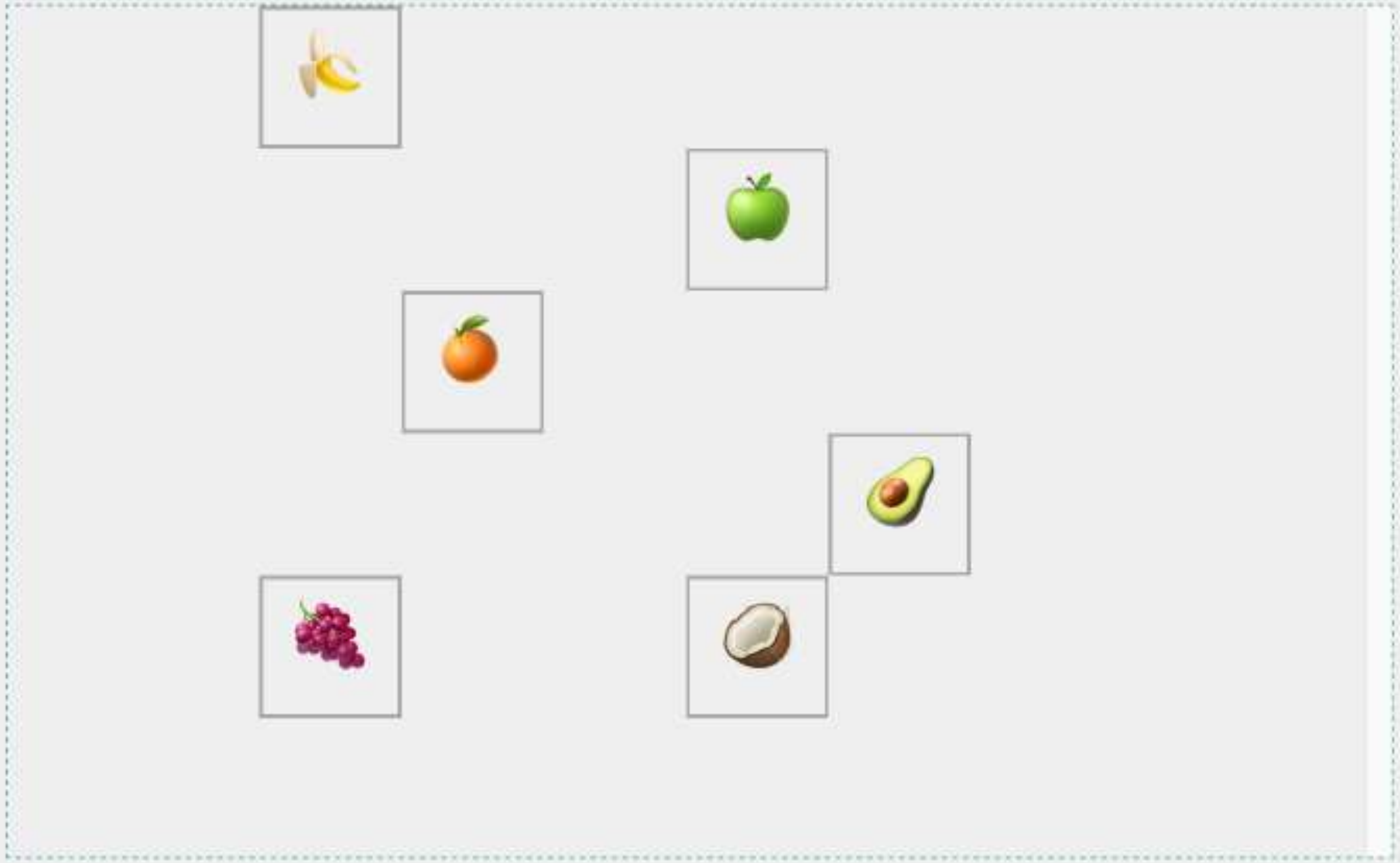
SOME GRID SHORTHANDS

(because brevity is a virtue...maybe)

¯_(ツ)_/¯

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

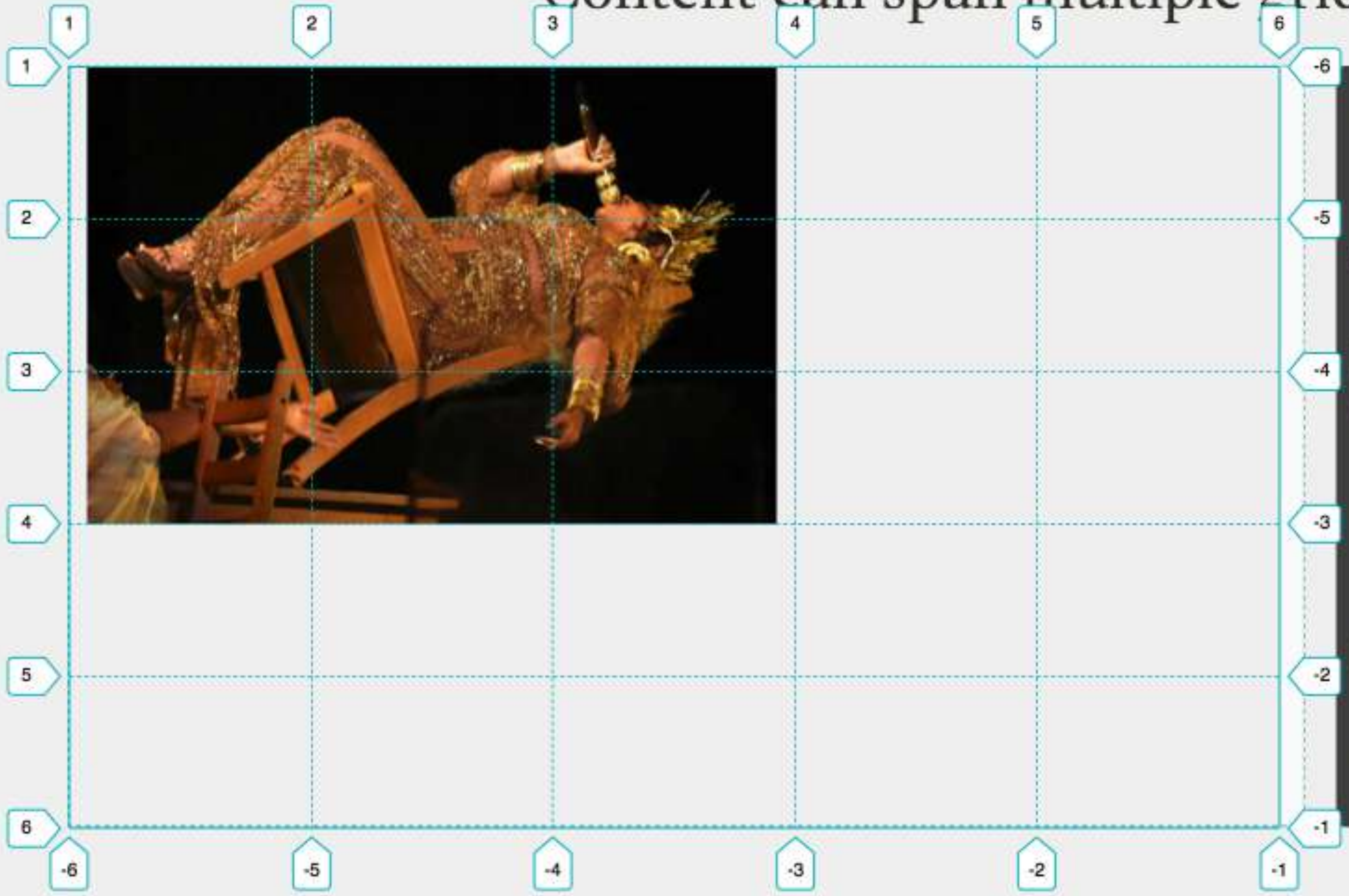
By default, grid items will take up the space of 1 grid cell



```
.grid9__item:nth-child(1)
{
  grid-row: 2;
  grid-column: 4;
}

.grid9__item:nth-child(2)
{
  grid-row: 3;
  grid-column: 2;
}

.grid9__item:nth-child(3)
{
  grid-row: 5;
```

The span keyword

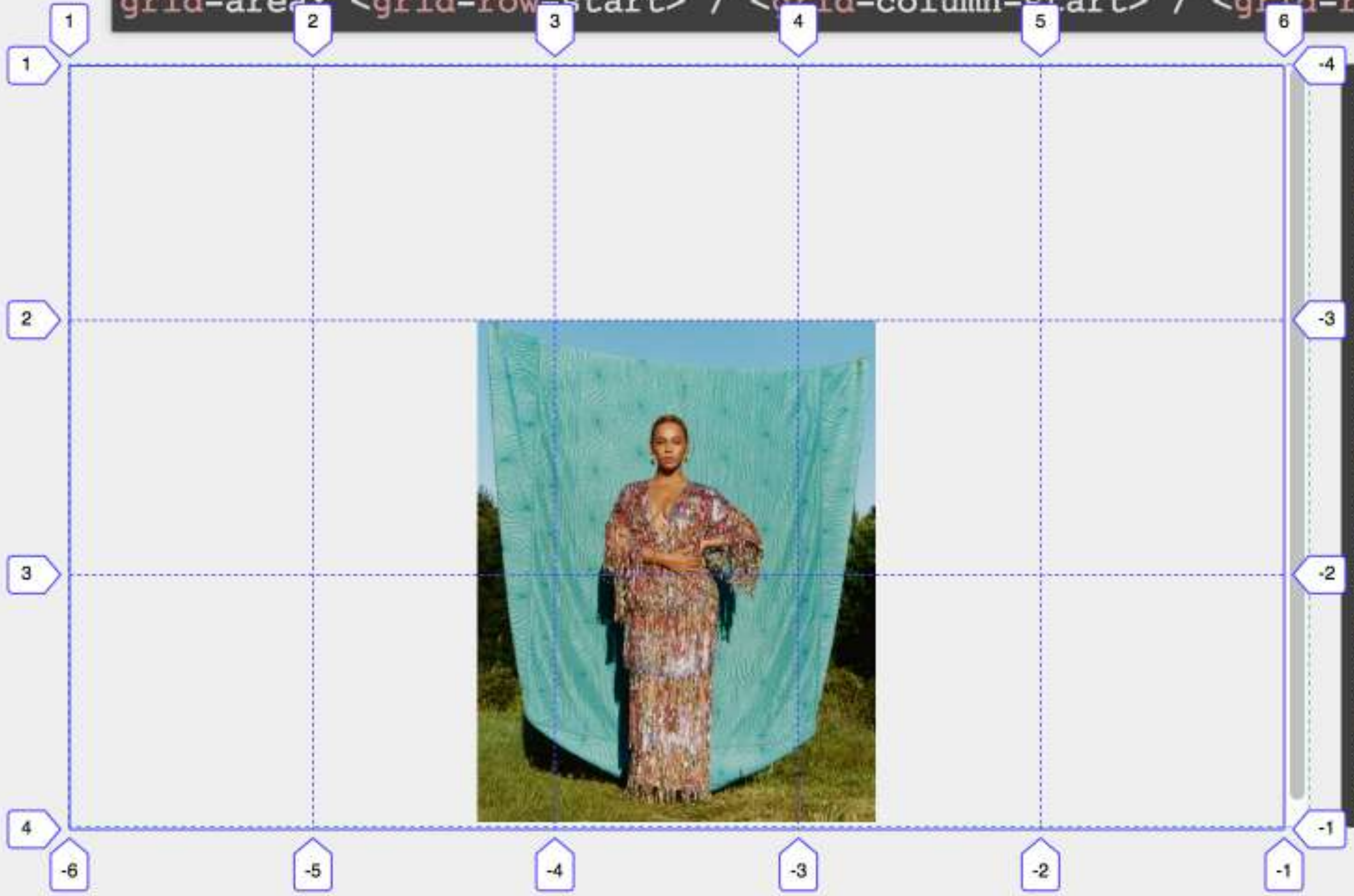
Content can span multiple grid cells



```
.grid10 {
  display: grid;
  grid-template-columns:
20% 20% 20% 20% 20%;
  grid-template-rows:
12vh 12vh 12vh 12vh 12vh;
}

.grid10__item {
  grid-row: span 3 / 4;
  grid-column: span 3 / 4
}

.grid10__item img {
  height: 100%;
```

Using the `grid-area` shorthand

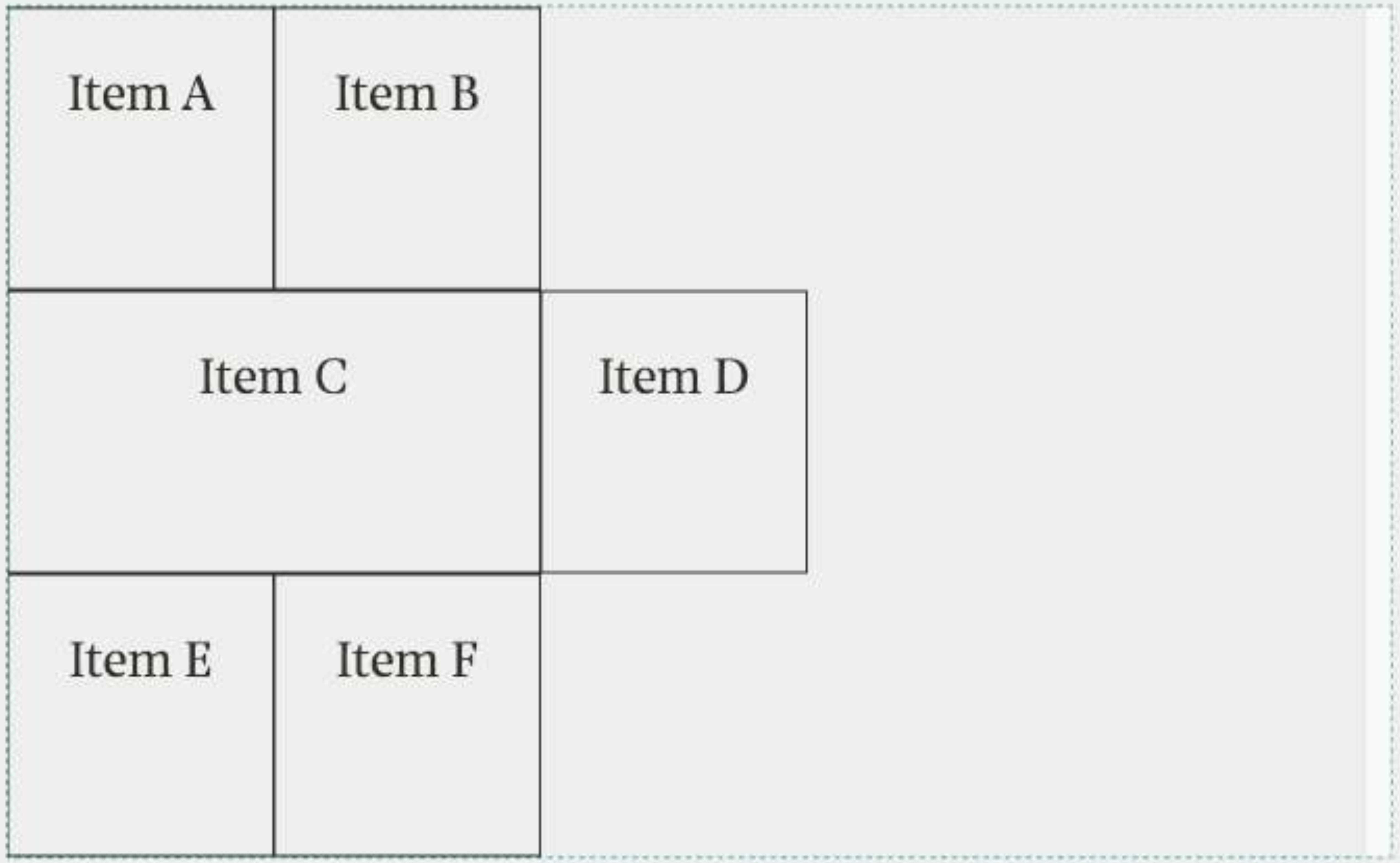
```
grid-area: <grid-row-start> / <grid-column-start> / <grid-row-end> / <grid-column-end>
```



```
  display: grid;
  grid-template-columns:
20% 20% 20% 20% 20%;
  grid-template-rows:
20vh 20vh 20vh;
}

.grid11__item {
  grid-area: 2 / 2 / 4 /
5;
}

.grid11_item img {
  height: 100%;
}
```

ASSIGNING NAMES TO GRID THINGS

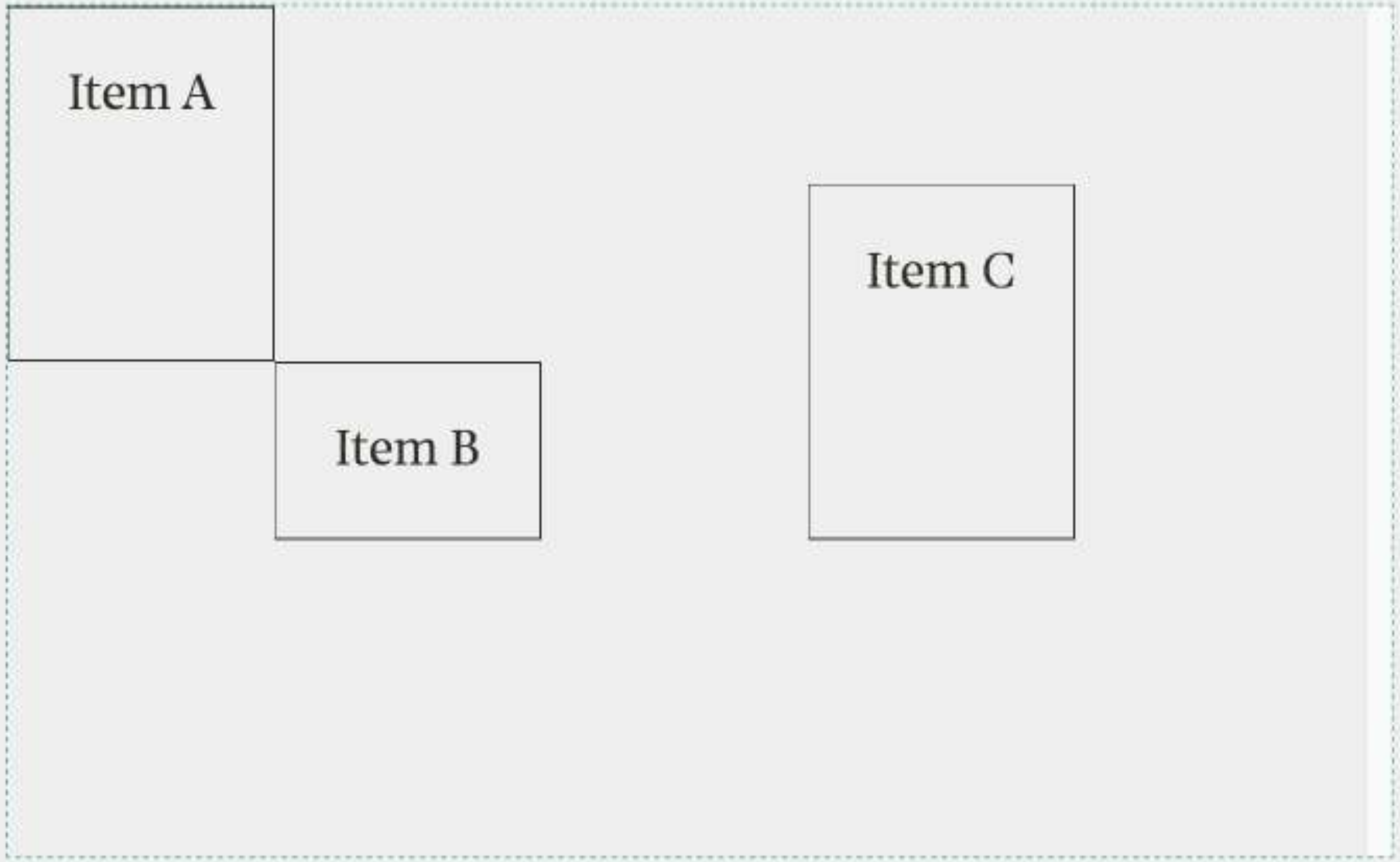
Naming grid lines



```
.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;
}
```

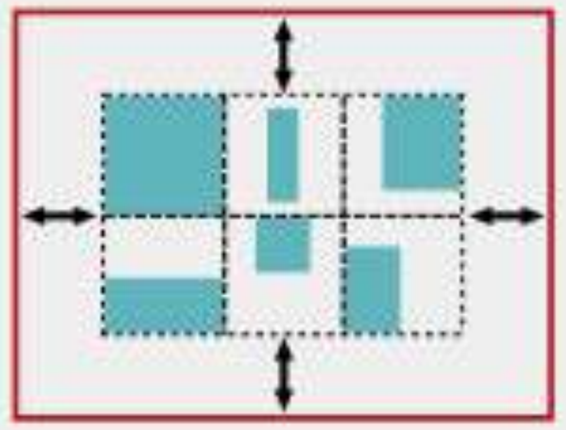
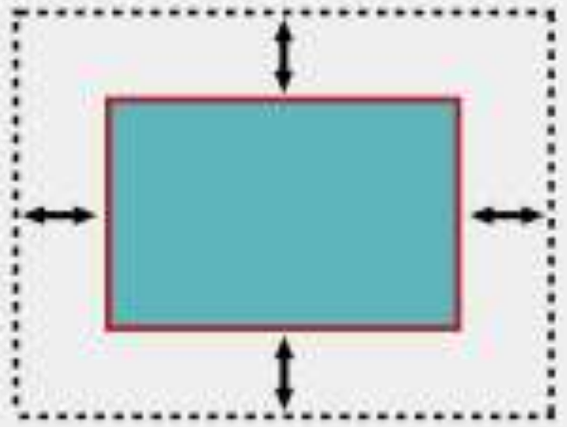
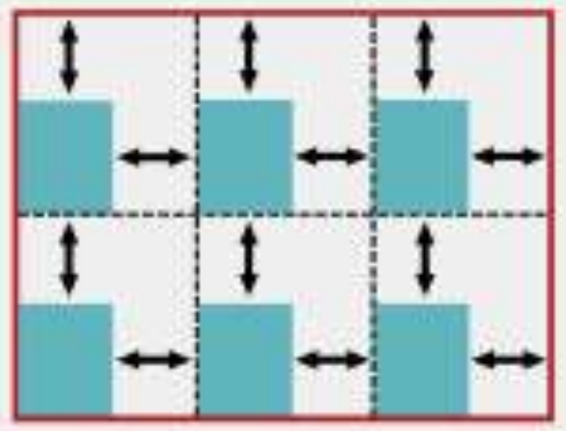
Naming grid areas

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:
```

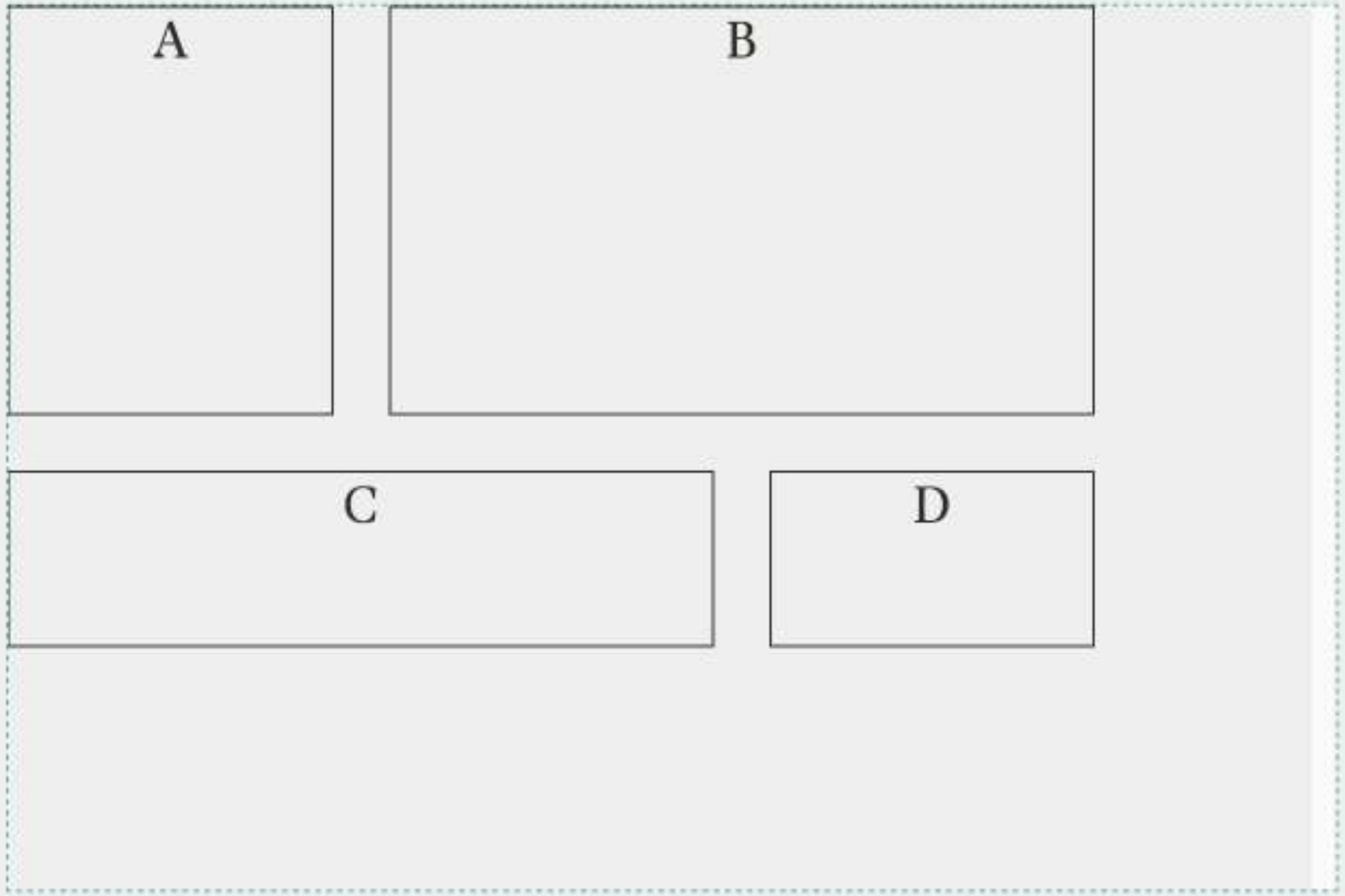
MAKING SENSE OF BOX ALIGNMENT

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

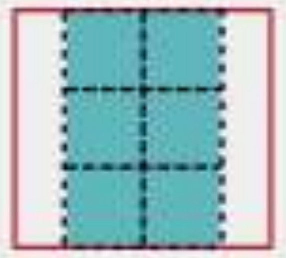
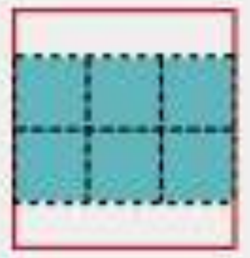
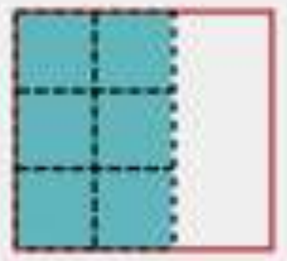
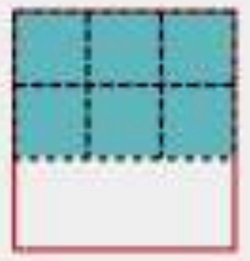
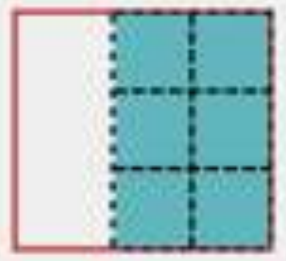
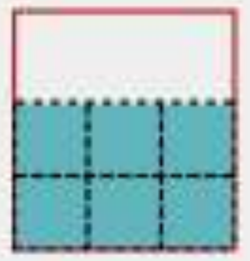
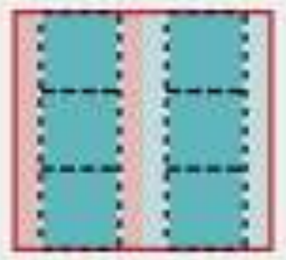
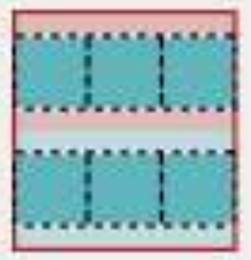
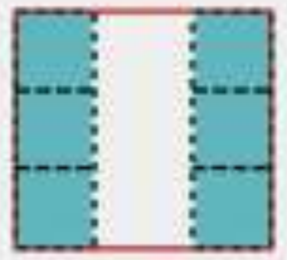
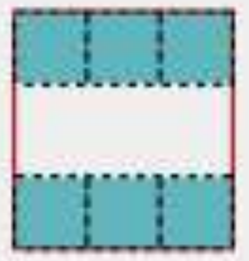
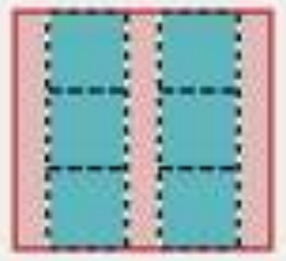
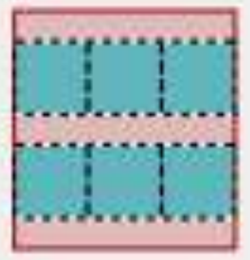
Source: CSS Box Alignment Module Level 3

justify/align-content

content-distribution properties



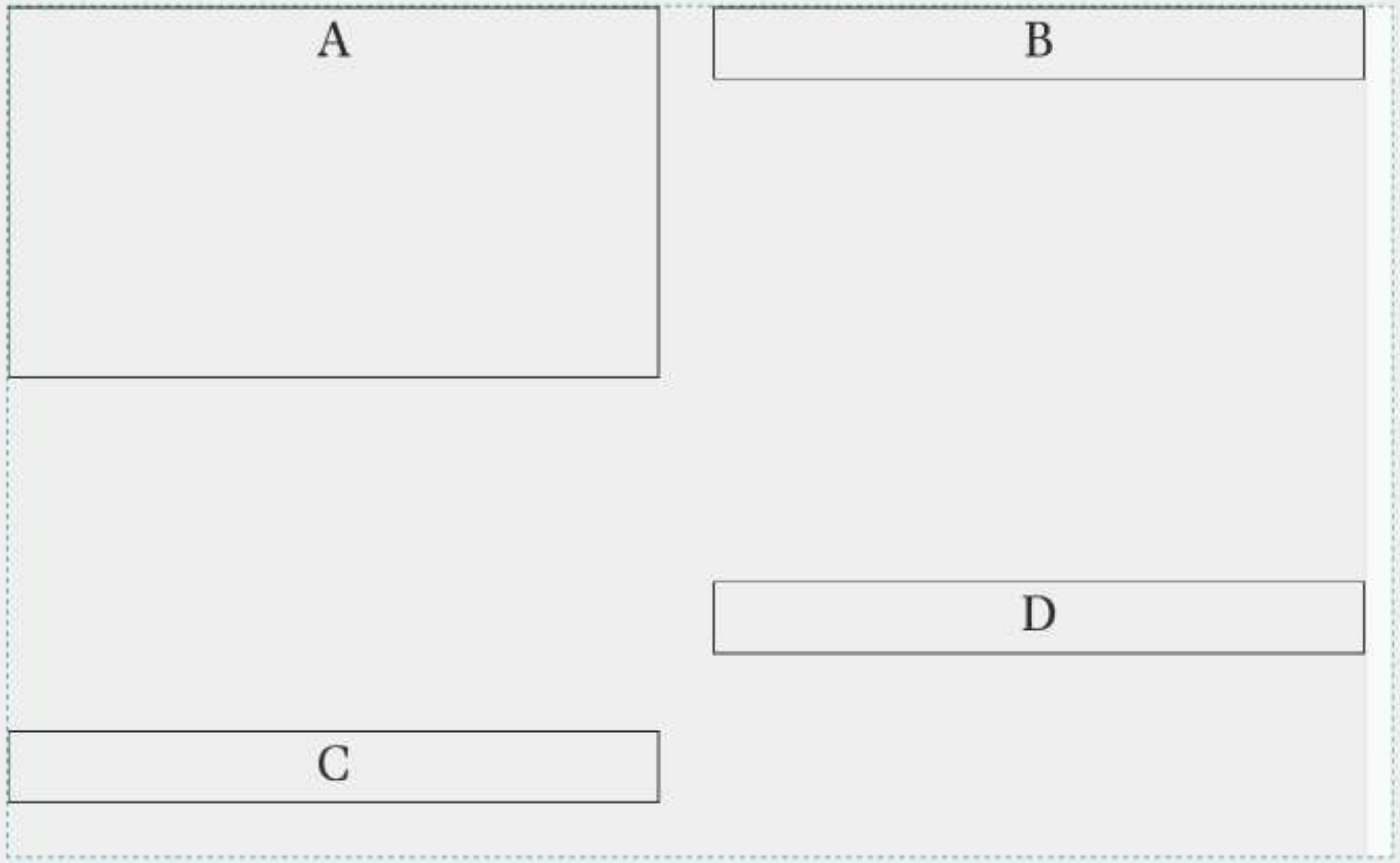
```
.content {
  justify-content: normal;
  align-content: normal;

  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";
}
```

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

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

justify/align-self

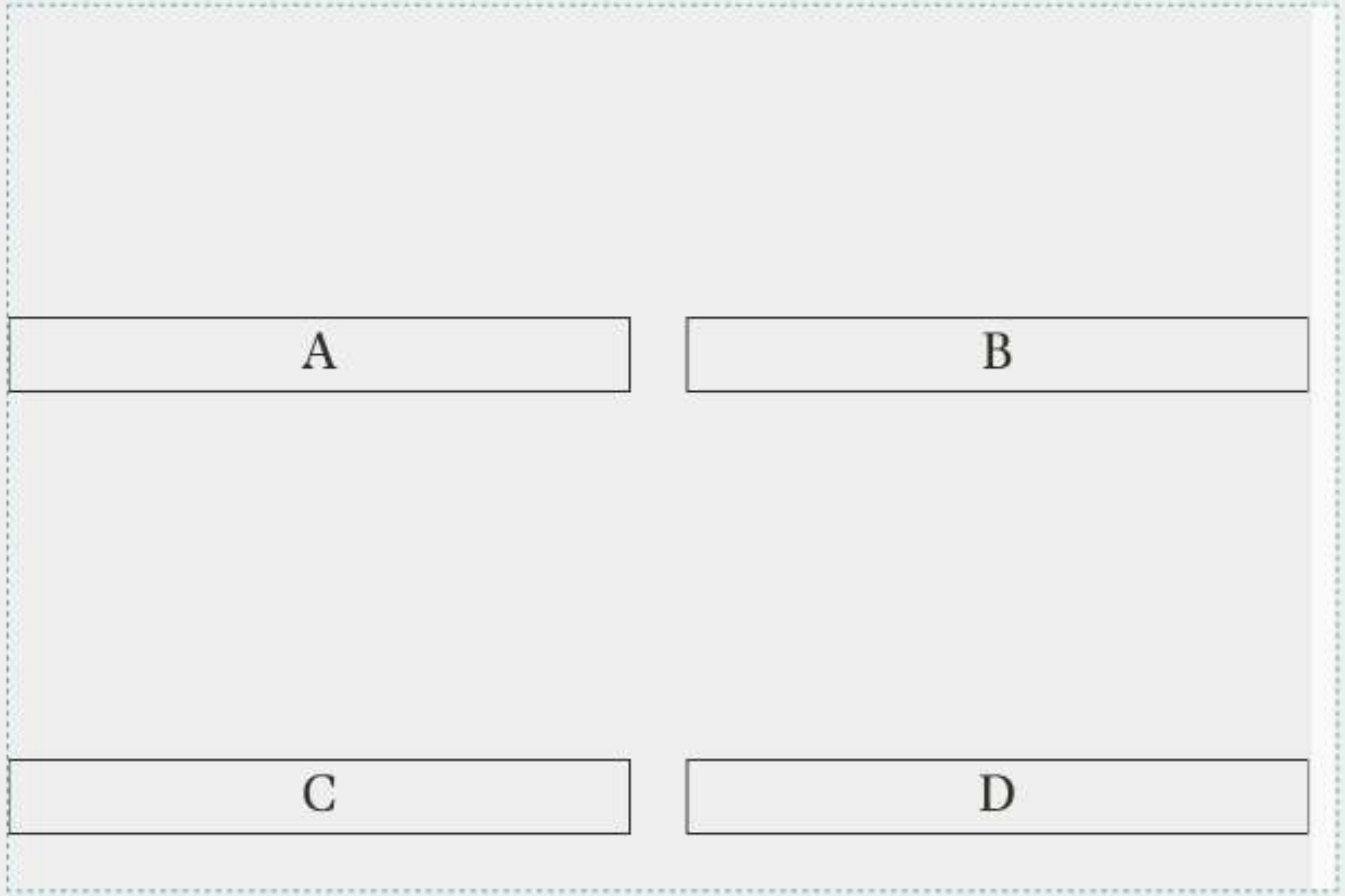
self-alignment properties



```
.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 {
```

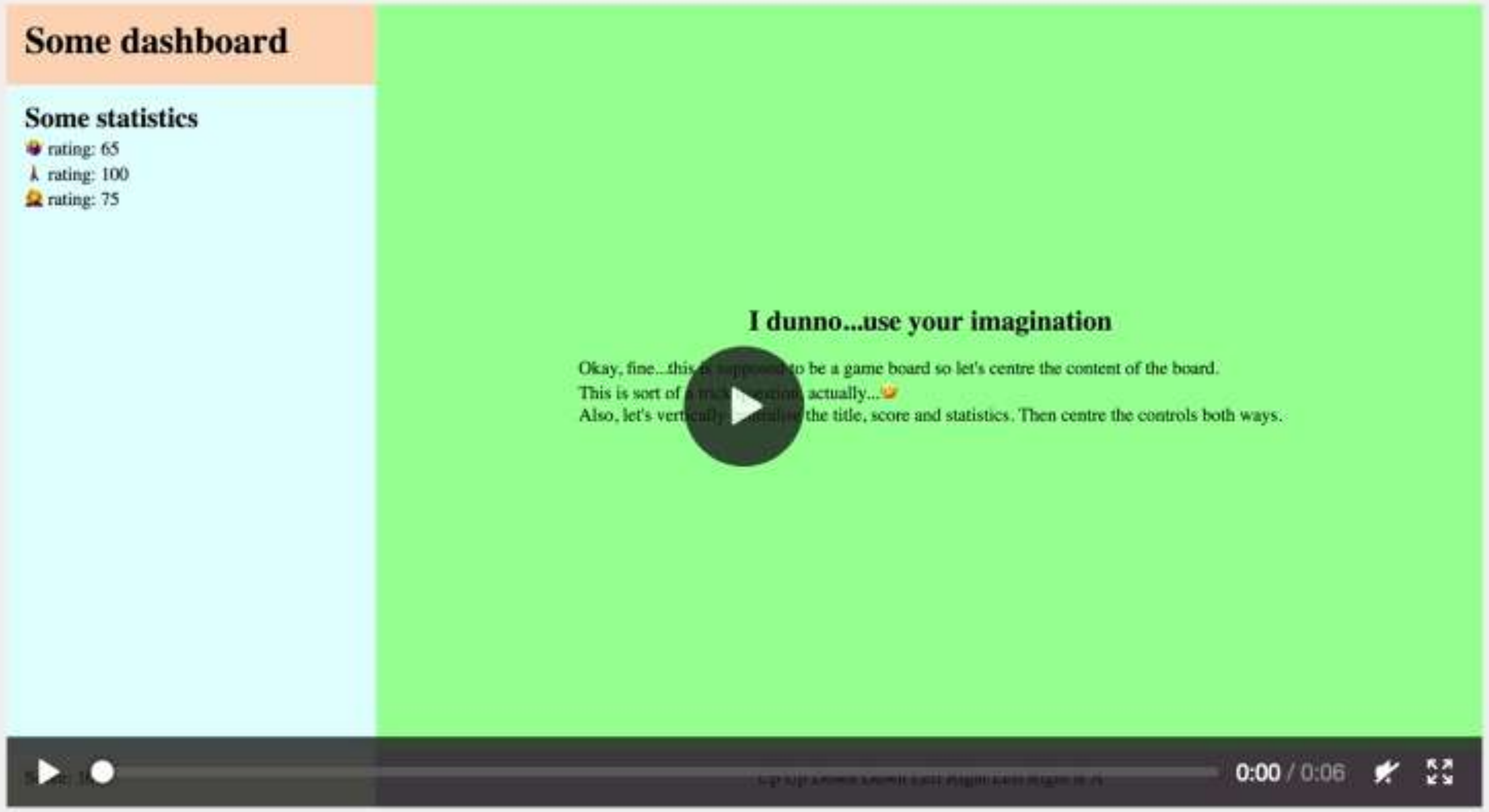
justify/align-items

defaults for justify/align-self



```
.items {
  justify-items: normal;
  align-items: end;

  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";
```

Simple responsive dashboard


Some dashboard

Some statistics
rating: 65
rating: 100
rating: 75

I dunno...use your imagination

Okay, fine...this [illegible] to be a game board so let's centre the content of the board.
This is sort of [illegible] actually...
Also, let's vertically [illegible] the title, score and statistics. Then centre the controls both ways.

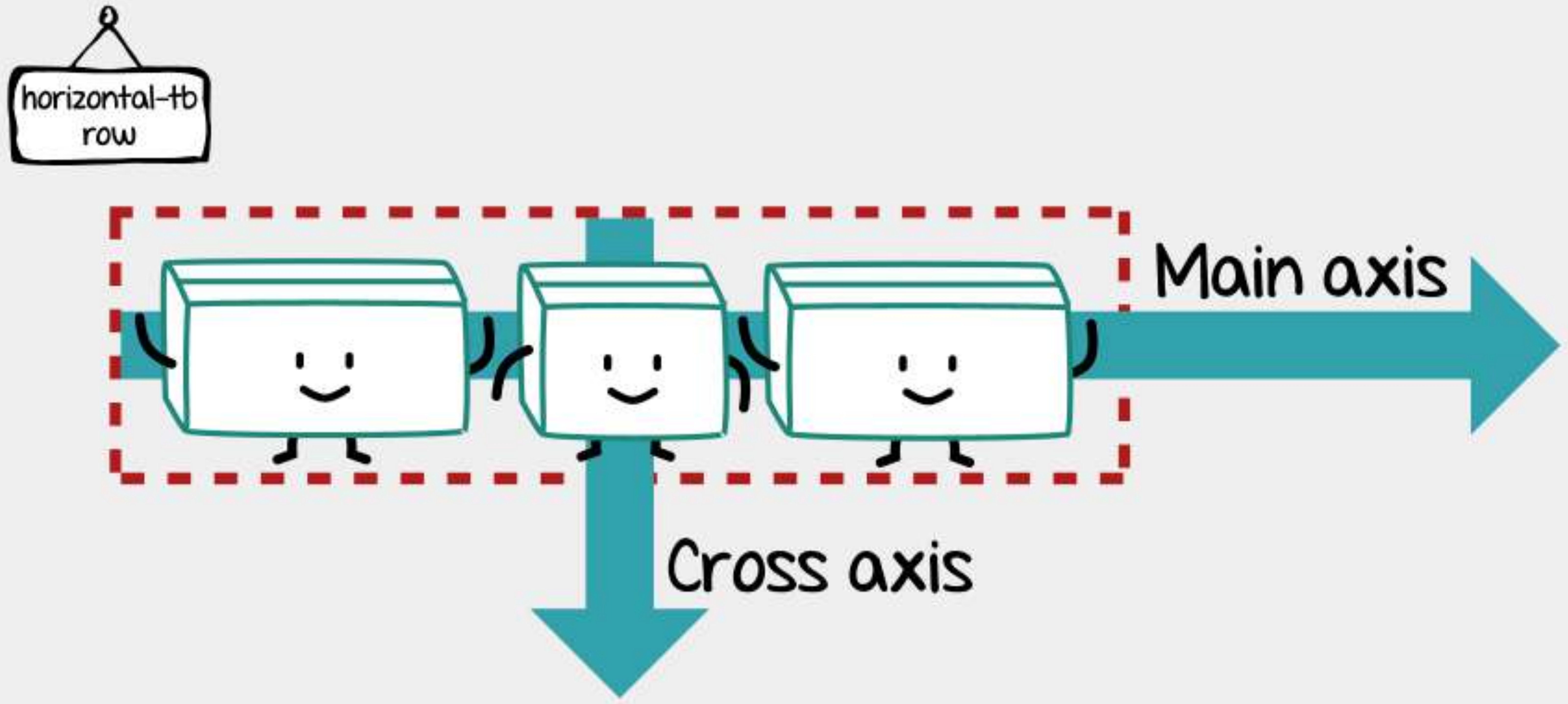
GRID VERSUS FLEXBOX?

WRONG QUESTION

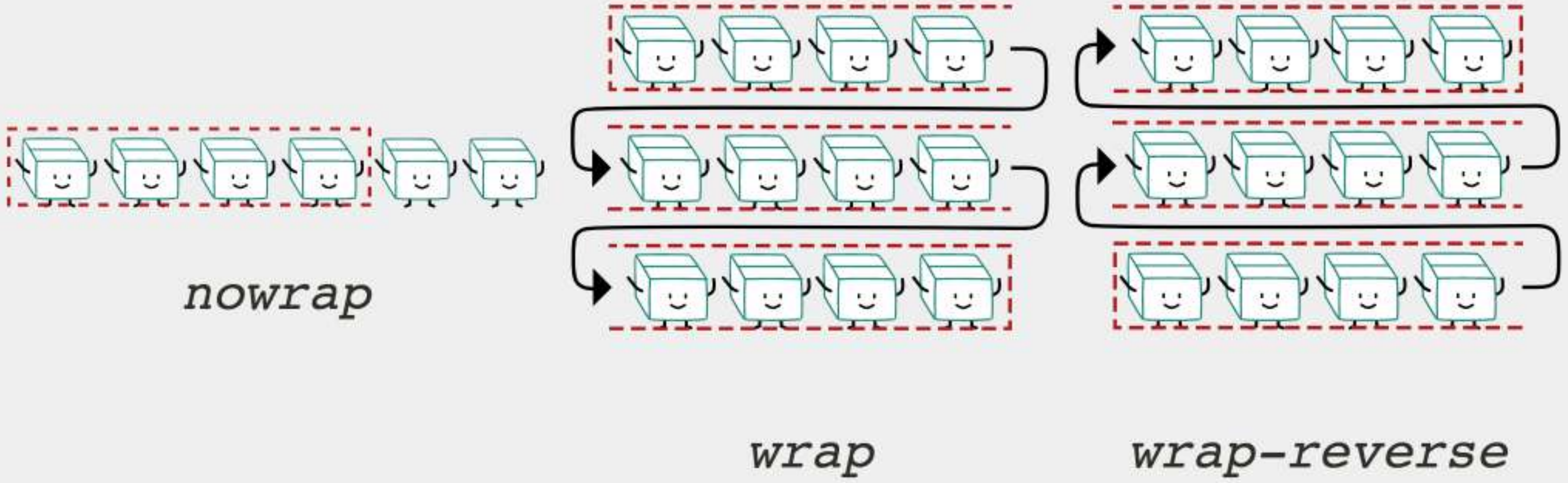
Grid AND Flexbox

FIGURING OUT FLEXBOX

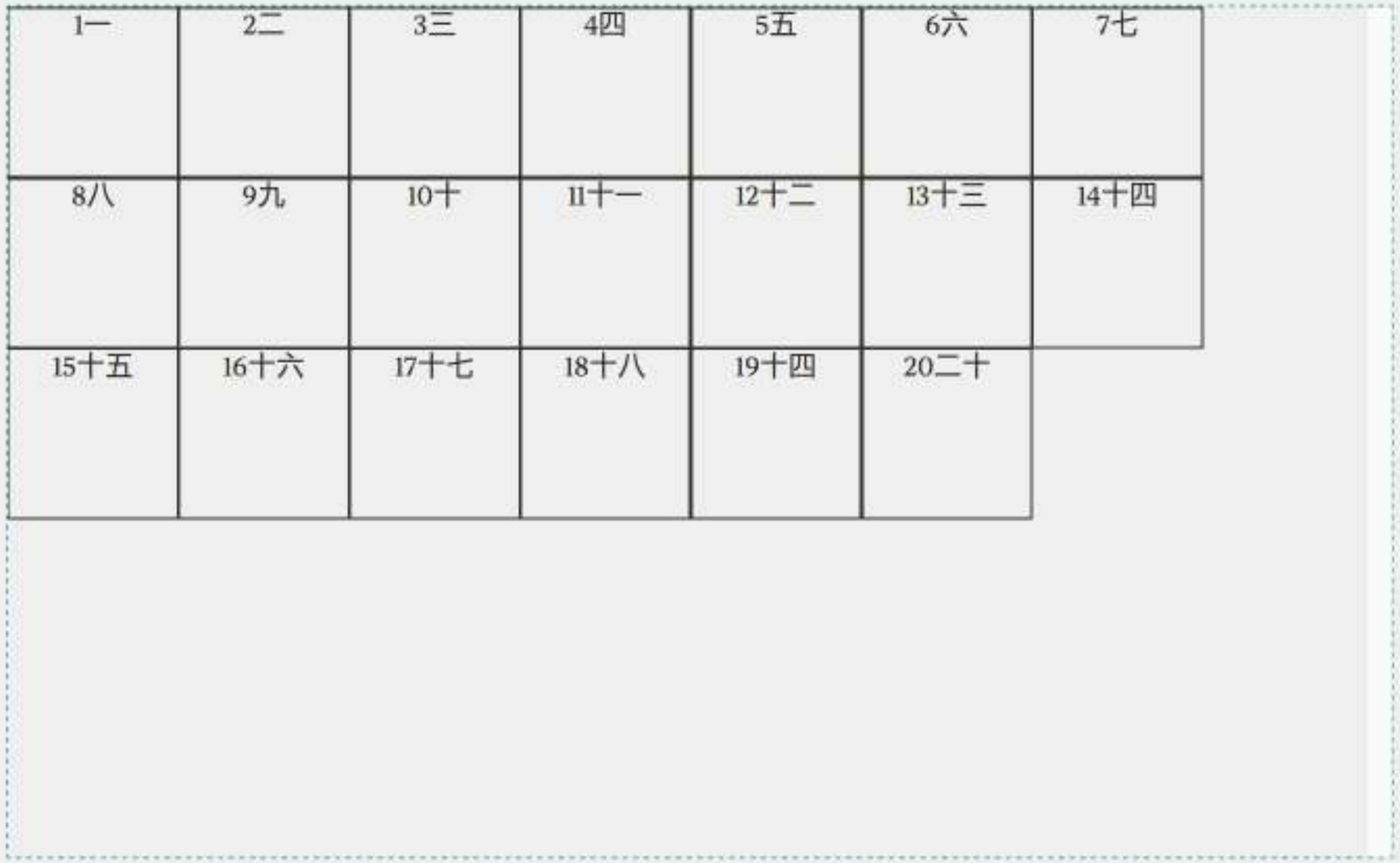
Flex axes



Flex lines



Flex directions



```
.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

`initial`	`0 1 auto`	cannot grow but can shrink when there isn't enough space
`auto`	`1 1 auto`	can grow and shrink to fit available space
`none`	`0 0 auto`	cannot grow or shrink, AKA inflexible
`<positive-number>`	`<positive-number> 1 0`	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

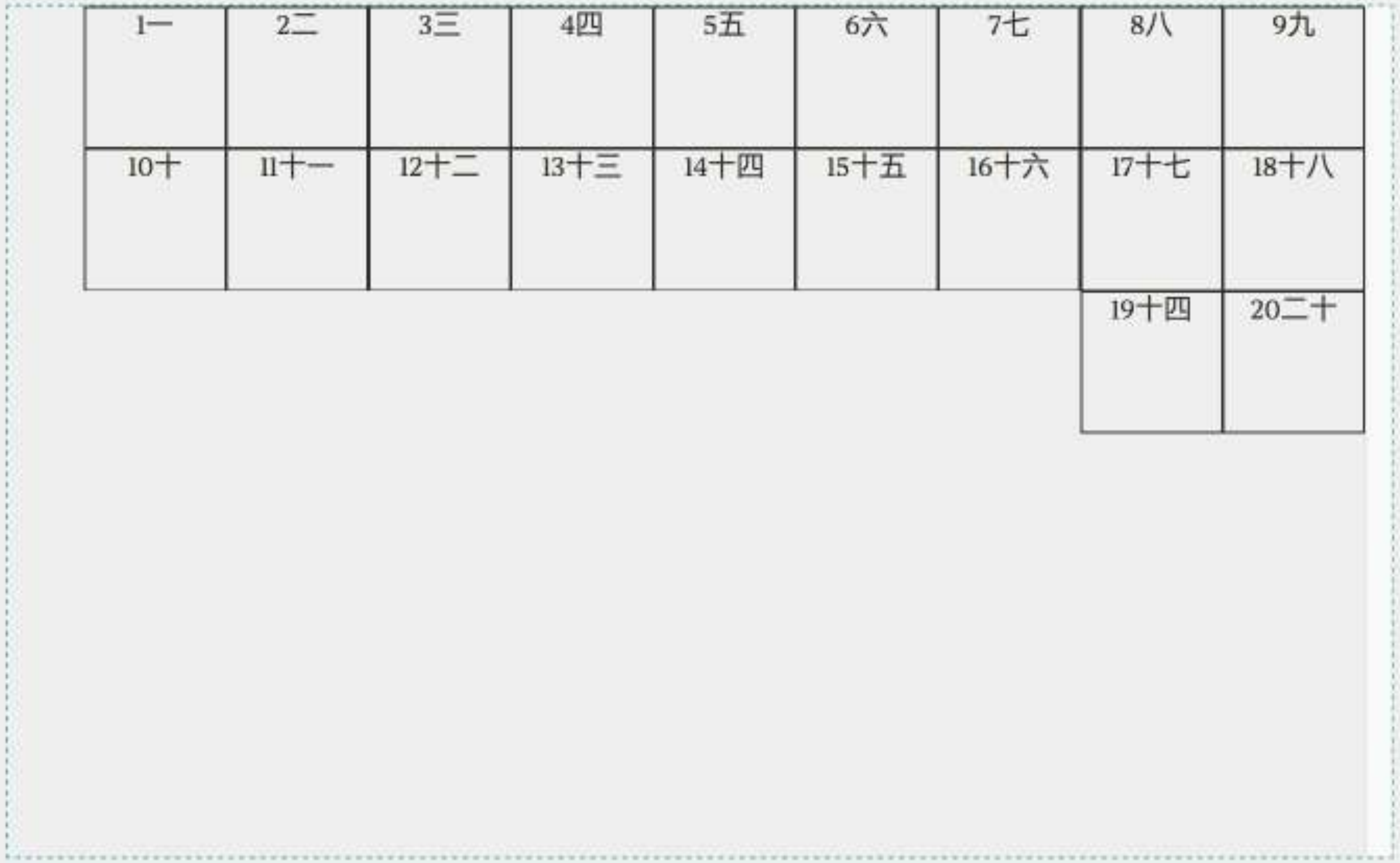
```
.automargin {
  display: flex;
}

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

Defining “auto” by Elika Etemad (AKA fantasai)

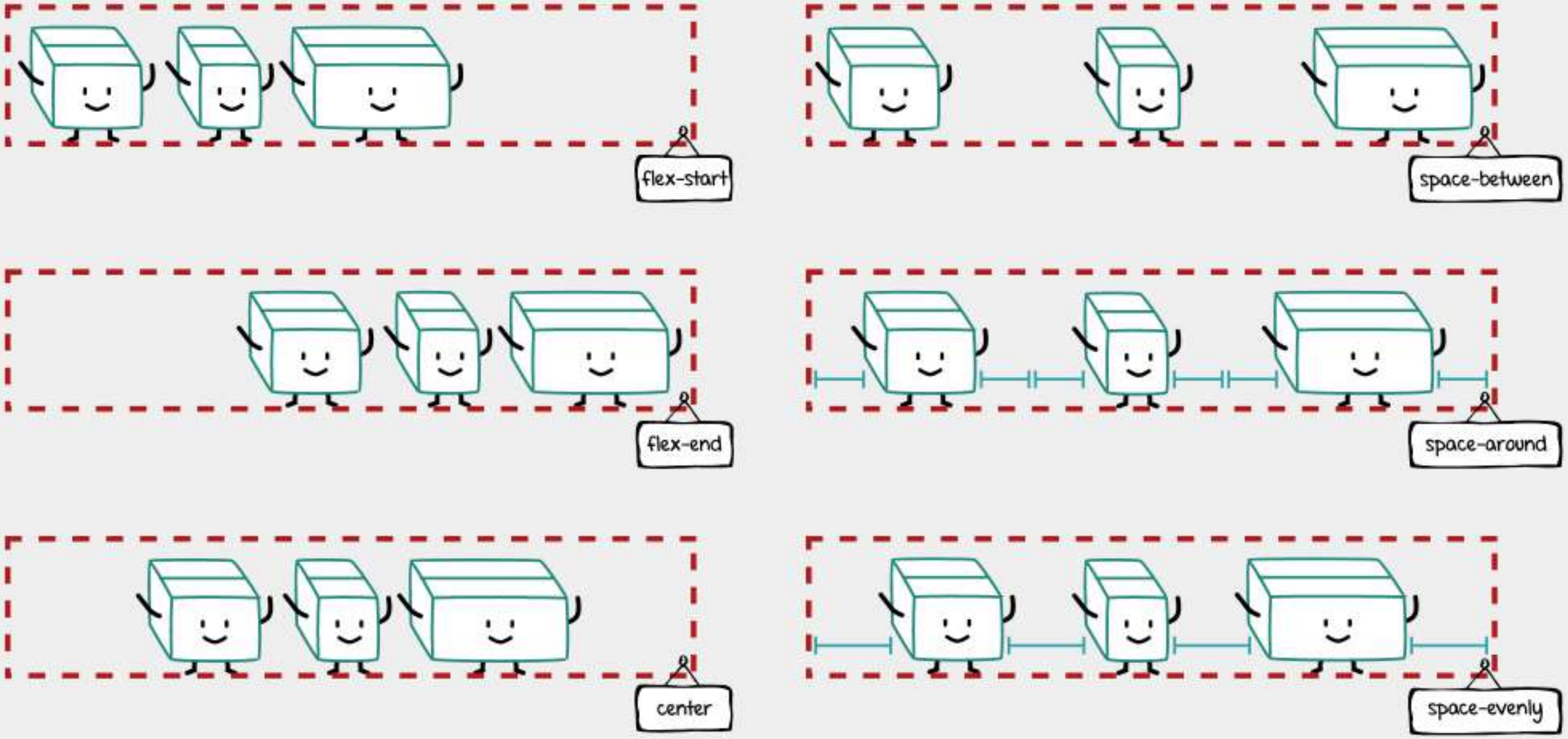
fantasai | Defining “auto” | CSS Day 2015
from Web Conferences Amsterdam
fantasai
Auto
52:43
vimeo

Aligning along the main axis

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

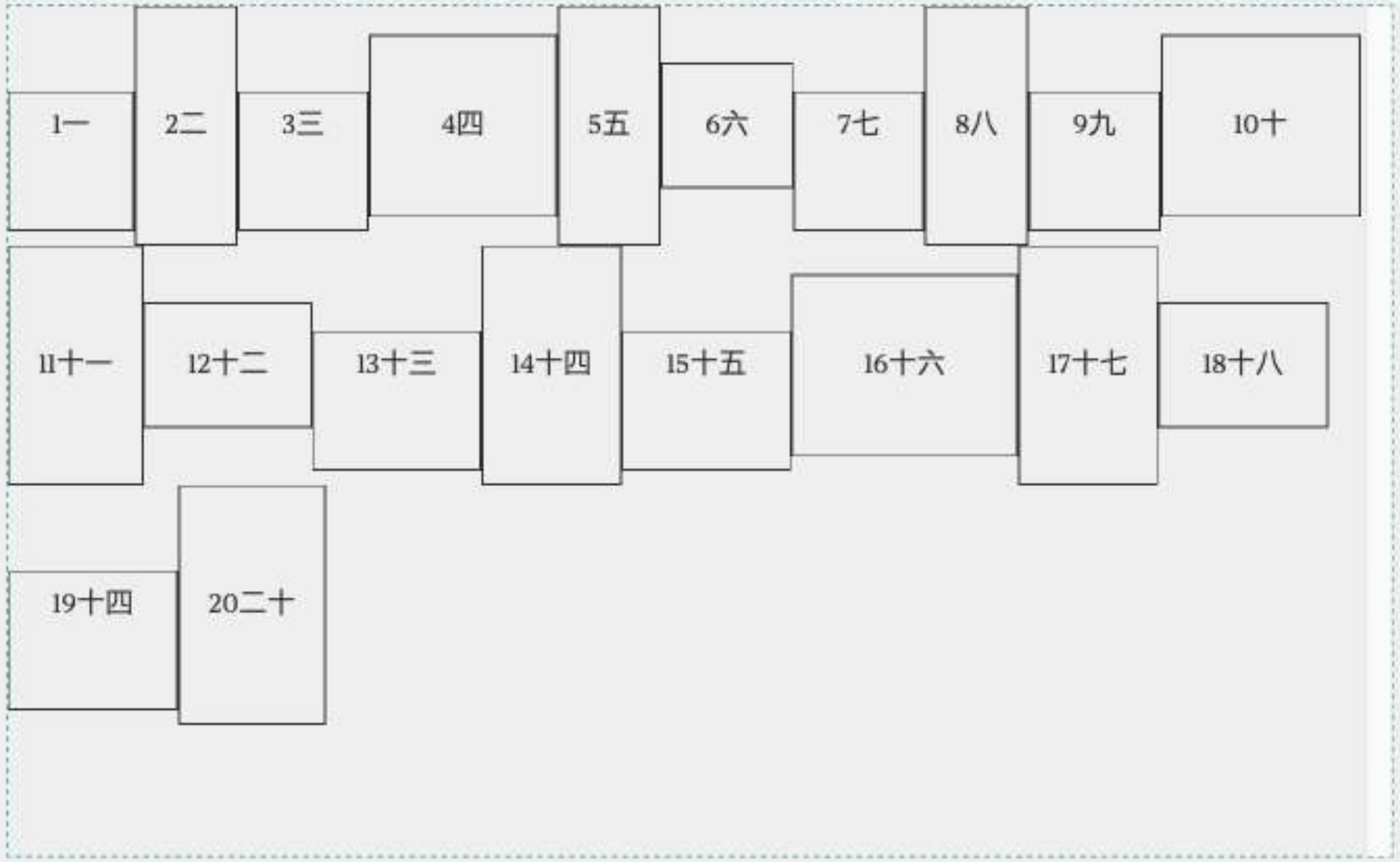


```
.mainaxis .wrapper {
  display: flex;
  flex-wrap: wrap;
  justify-content: flex-
end;
}

.mainaxis .box {
  height: 5em;
  width: 5em;
  border: 1px solid;
}
```

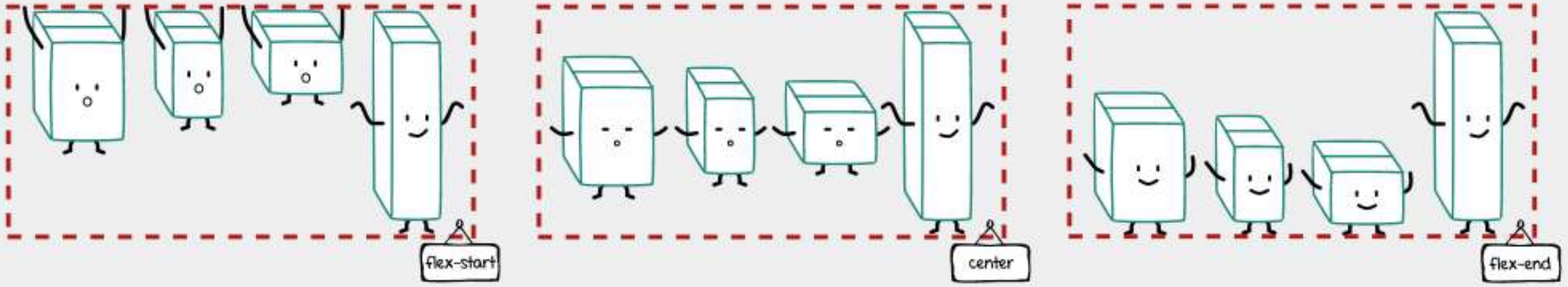

flex-start
space-between
flex-end
space-around
center
space-evenly

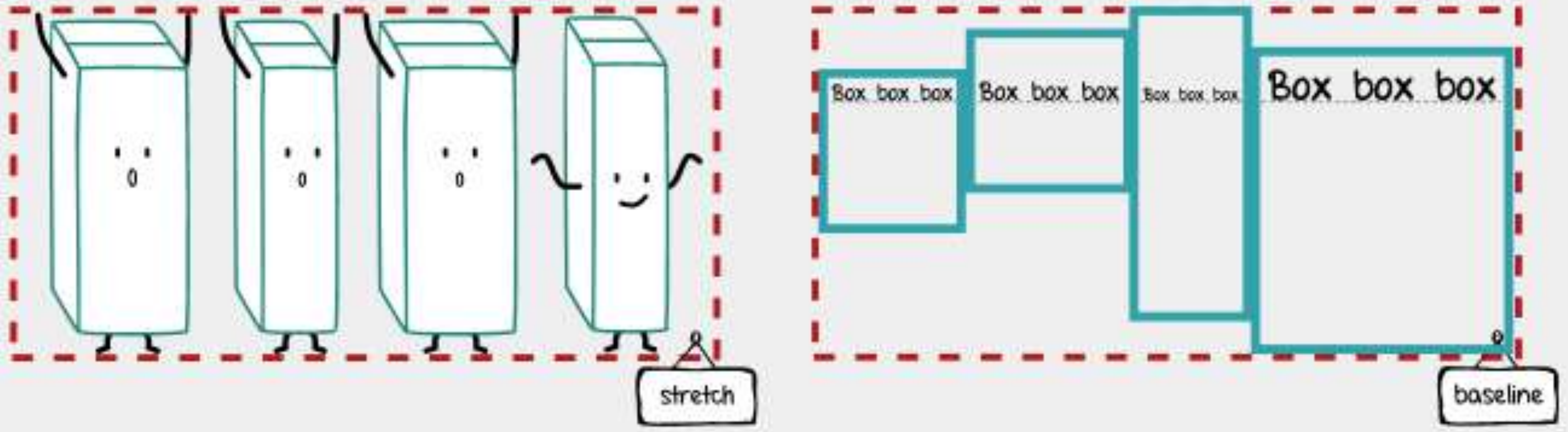
Aligning along the cross axis

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



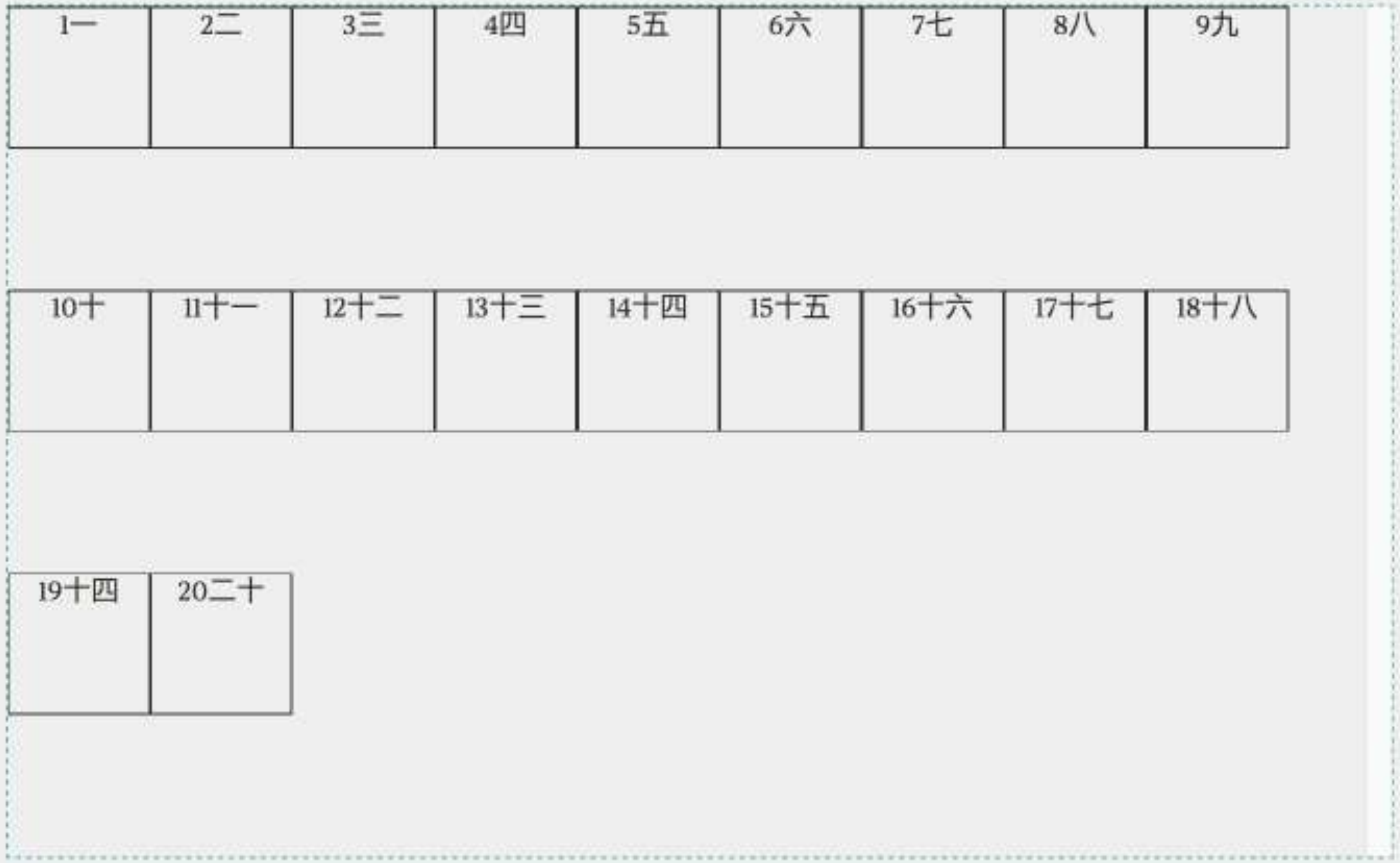
```
.crossaxis .wrapper {
  display: flex;
  flex-wrap: wrap;
  align-items: baseline;
}

.crossaxis .box {
  border: 1px solid
}

.crossaxis .box:nth-
child(2n) {
  padding: 2.5em;
}
```


flex-start
center
flex-end


stretch
Box box box
Box box box
Box box box
Box box box
baseline

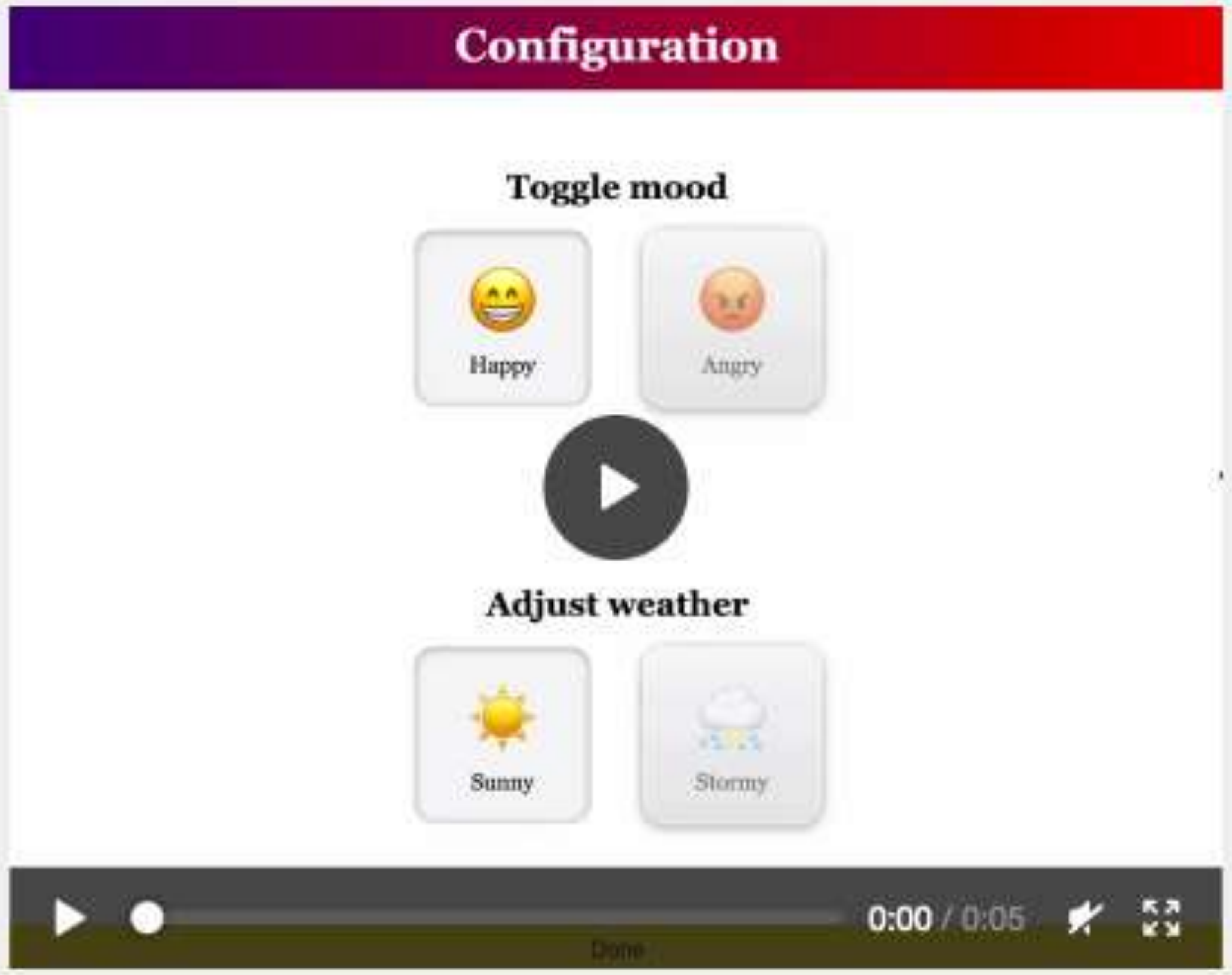
Packing flex lines

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



```
.packaxis .wrapper {
  display: flex;
  flex-wrap: wrap;
  align-content: stretch;
}

.packaxis .box {
  height: 5em;
  width: 5em;
  border: 1px solid;
}
```

Responsive configuration page



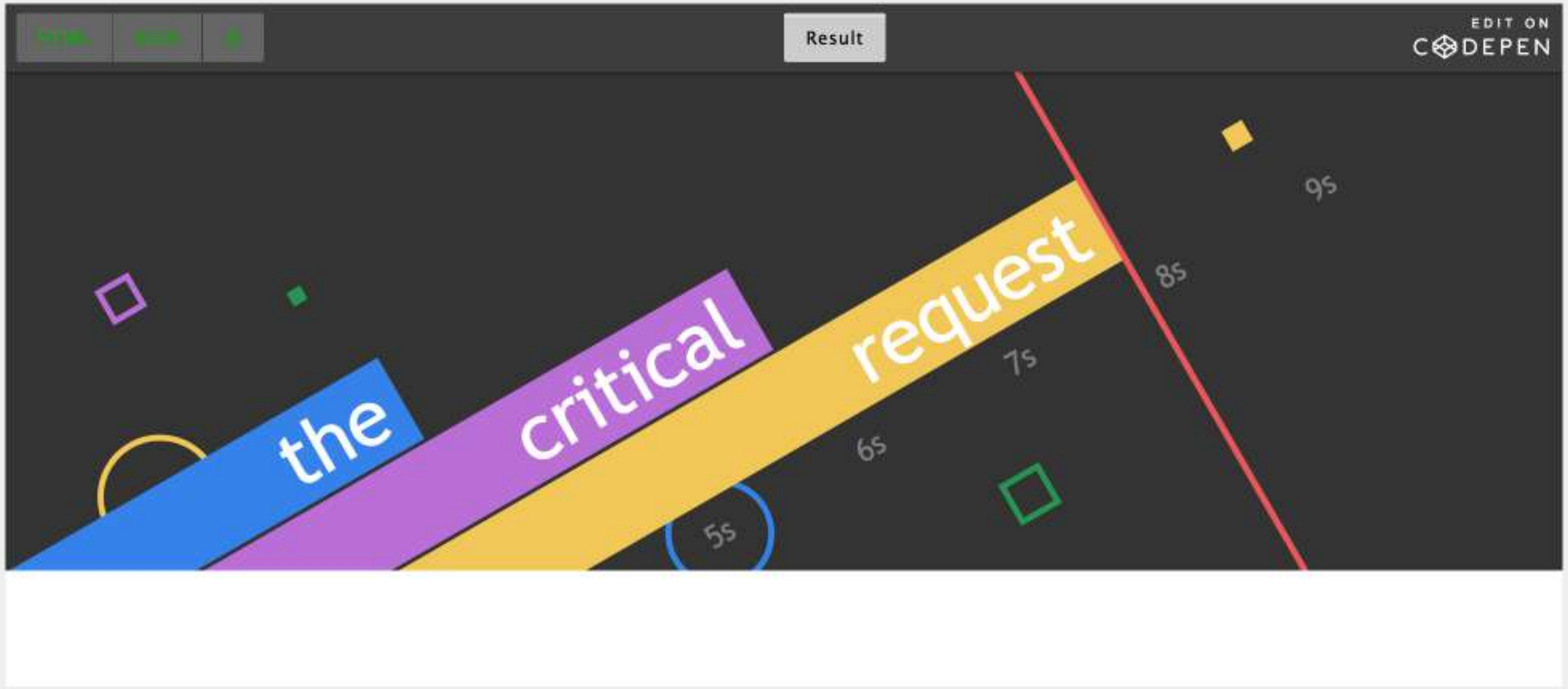
IS IT SAFE TO USE CSS GRID IN PRODUCTION?

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

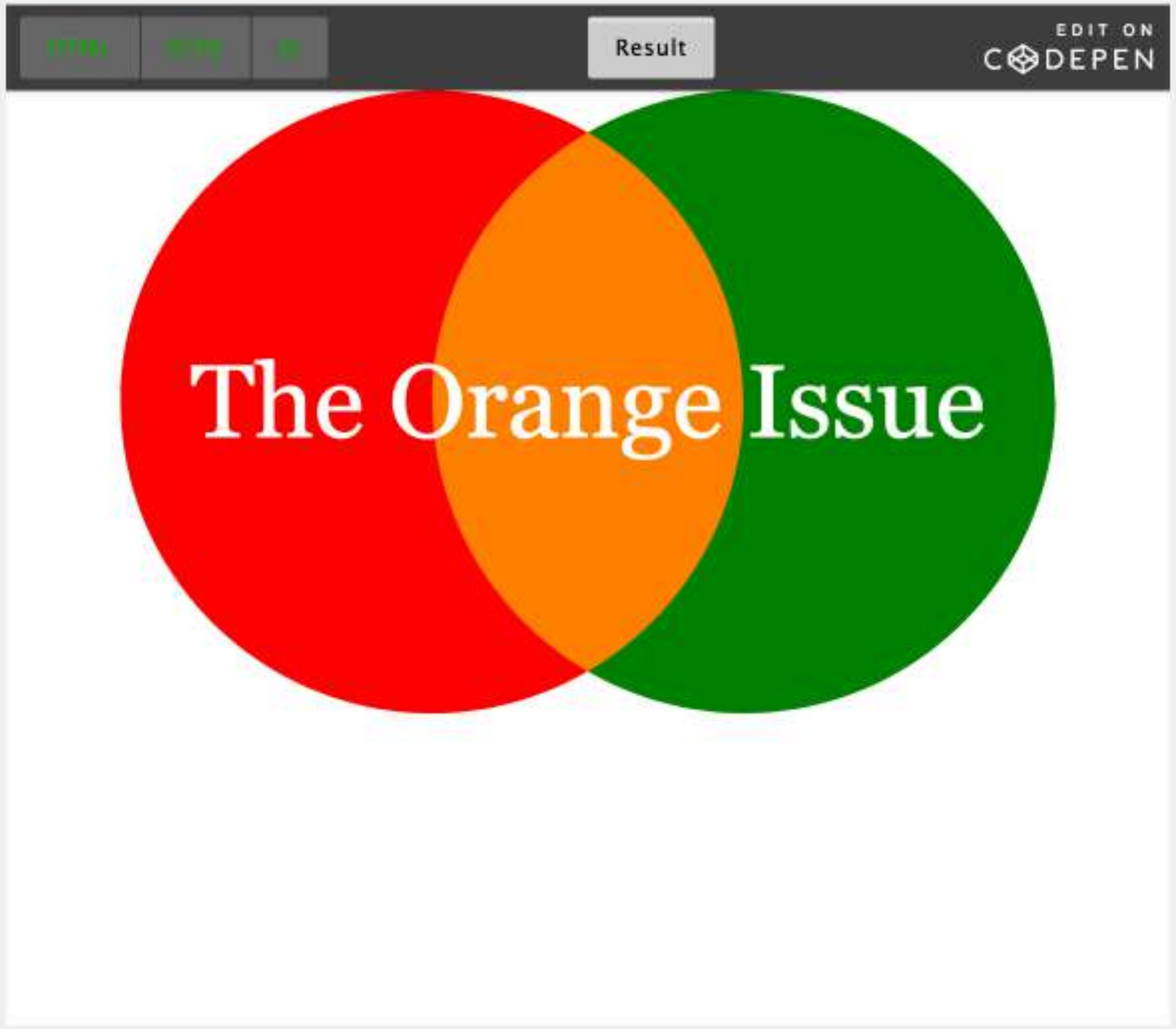


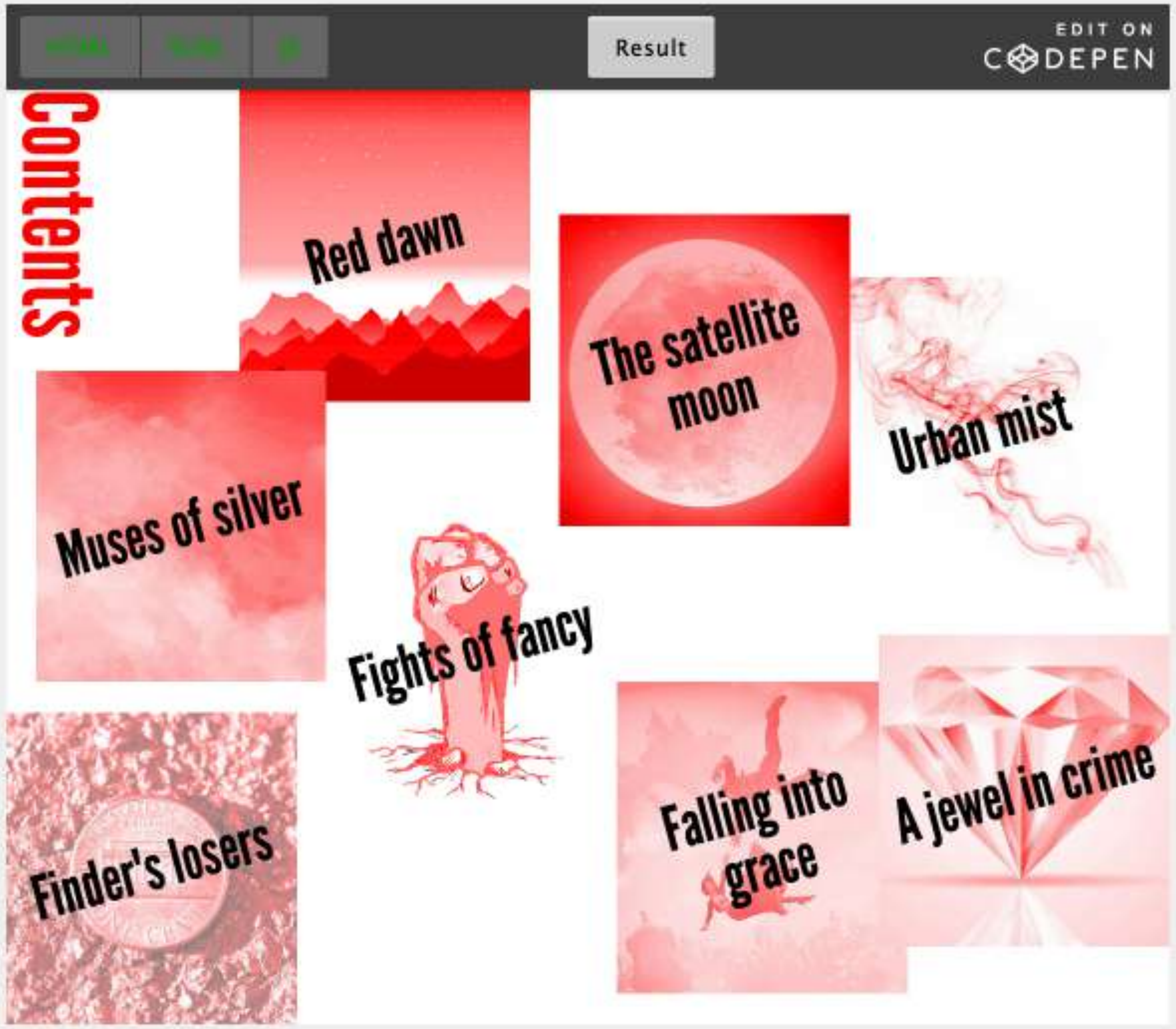
HOW WE CAN IMPROVE DESIGN USING GRIDS?

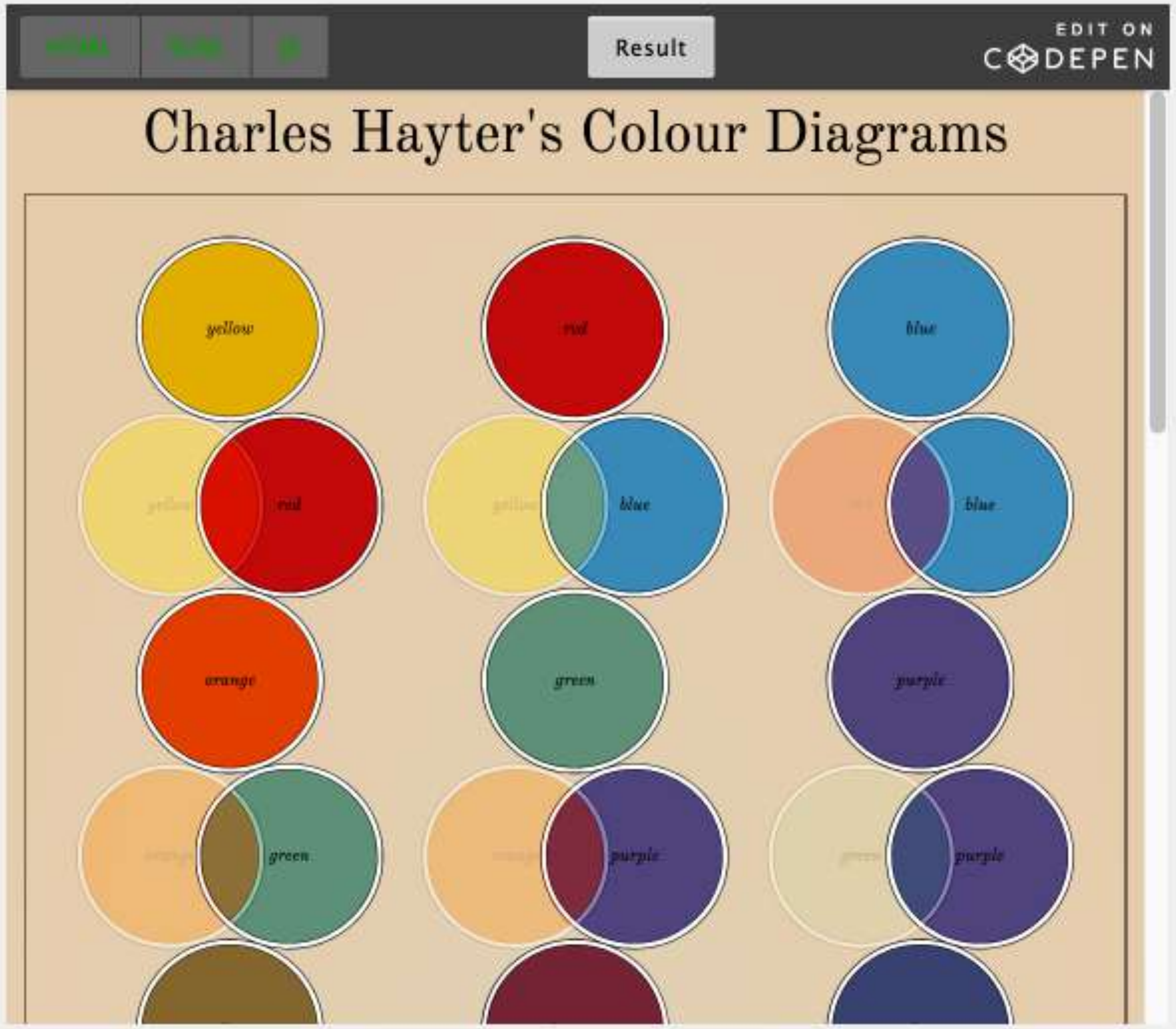
and

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

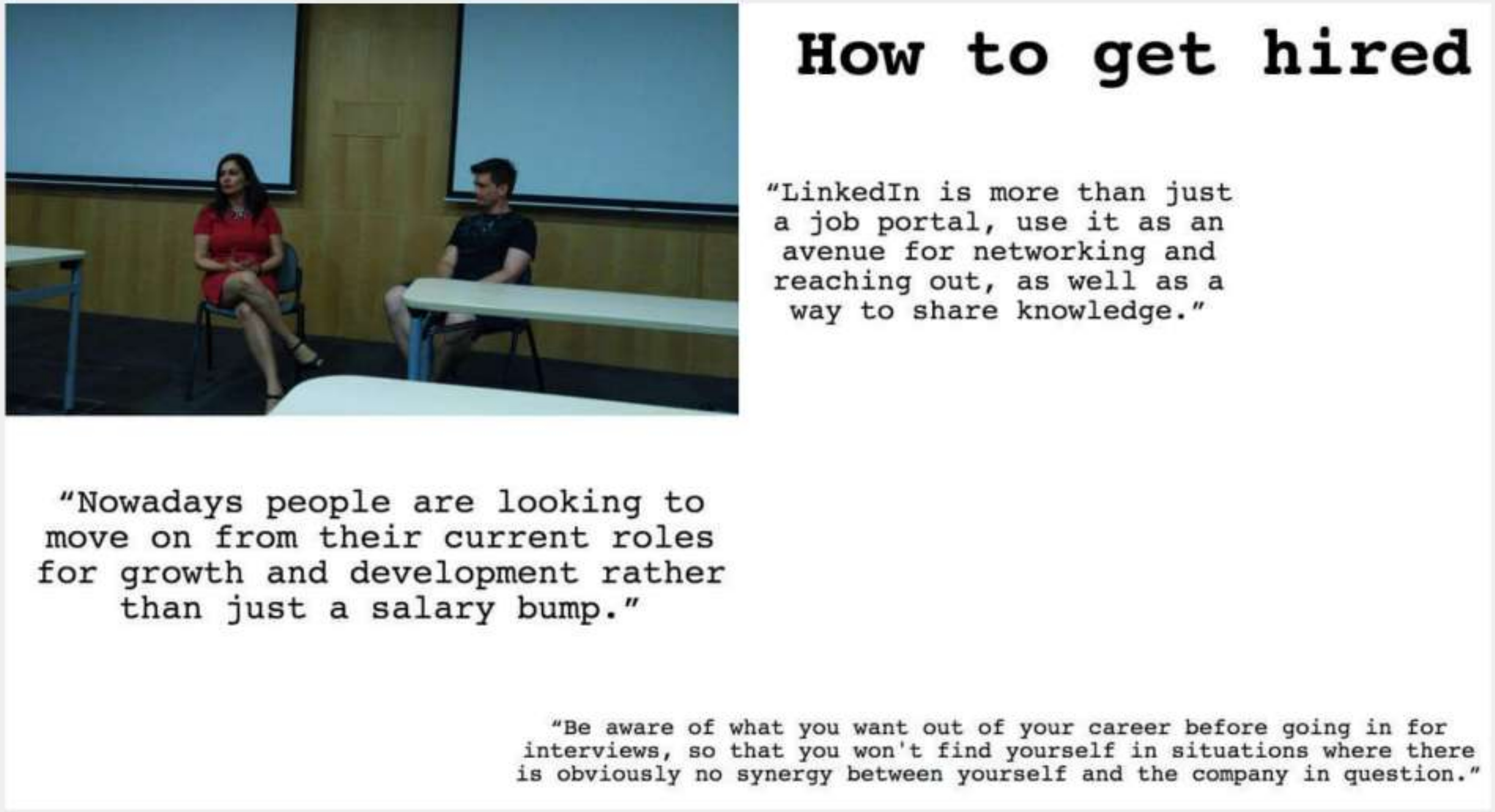
Overlap




HTML
SCSS
JS
Result
EDIT ON CODEPEN
Contents
Red dawn
The satellite moon
Urban mist
Muses of silver
Fights of fancy
Finder's losers
Falling into grace
A jewel in crime


HTML
SCSS
JS
Result
EDIT ON CODEPEN
Charles Hayter's Colour Diagrams
yellow
red
blue
yellow
red
yellow
blue
red
blue
orange
green
purple
orange
green
orange
purple
green
purple

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."

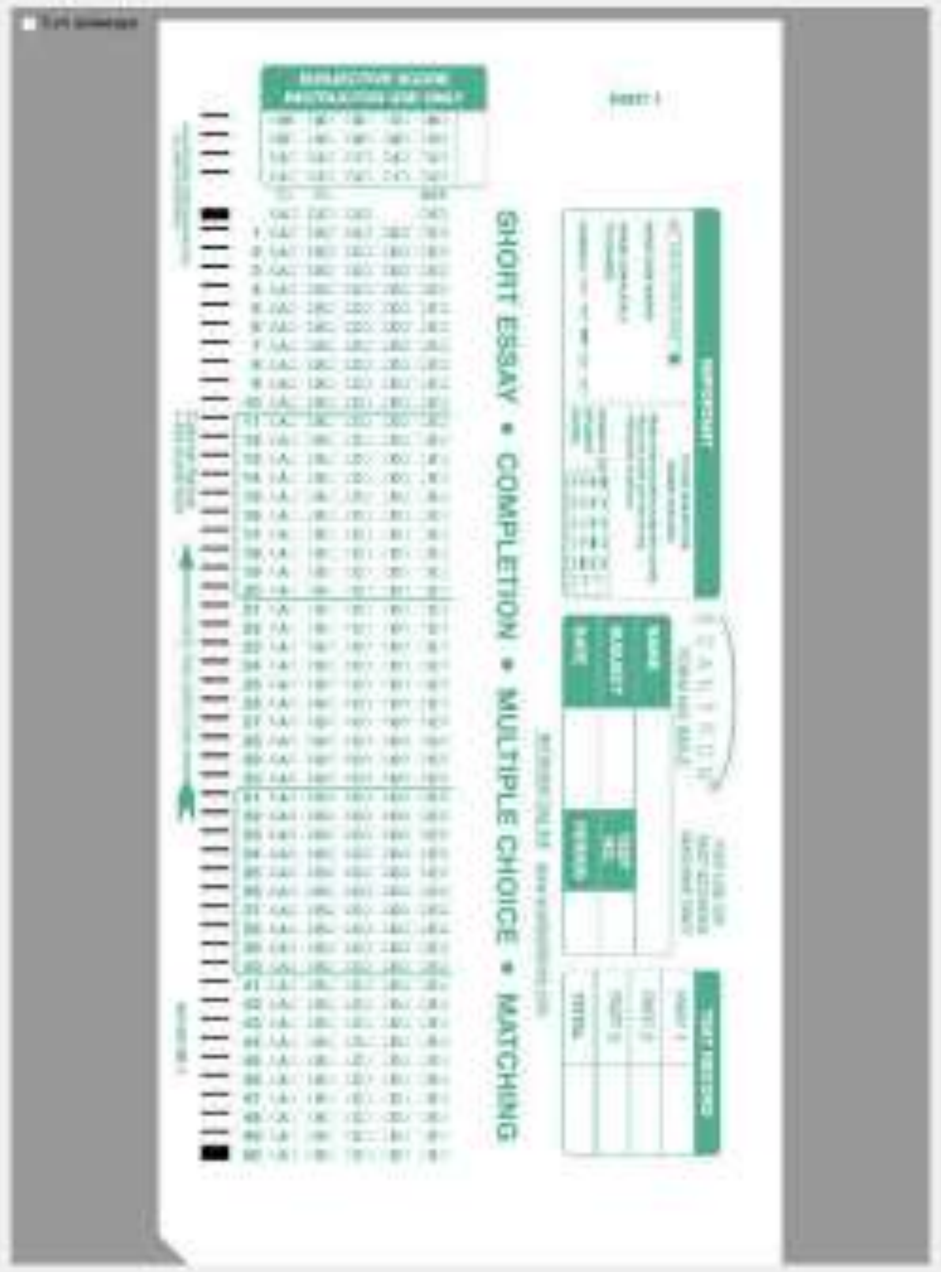
"Nowadays people are looking to move on from their current roles for growth and development rather than just a salary bump."

"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

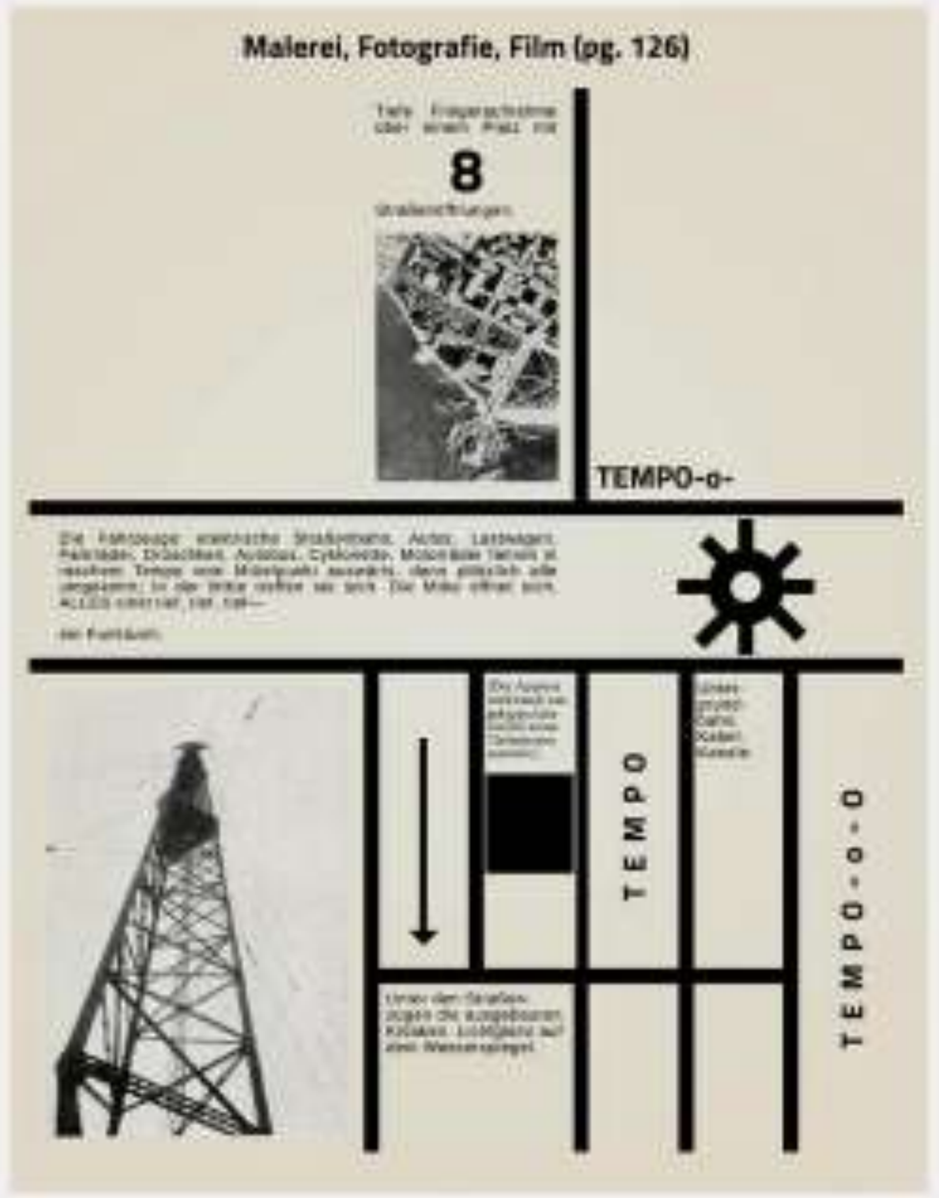


by Jon Kantner

by Yuan Chuan

by Andy Barefoot

CSS grid showcase



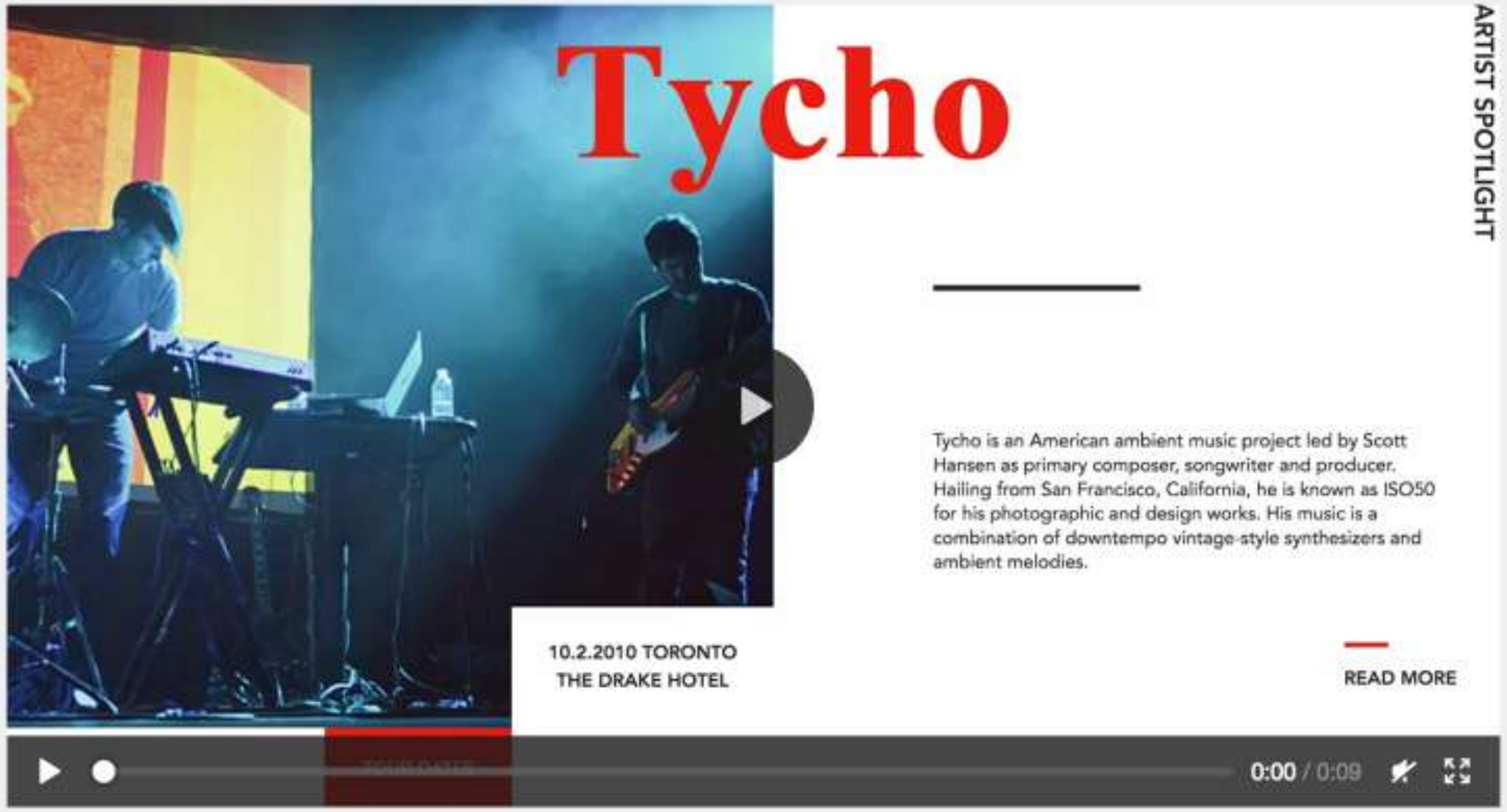
by Chen Hui Jing

by Adrian Roworth

by Aysha Anggraini

by Chen Hui Jing

Artist profile page



Tycho

ARTIST SPOTLIGHT

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.

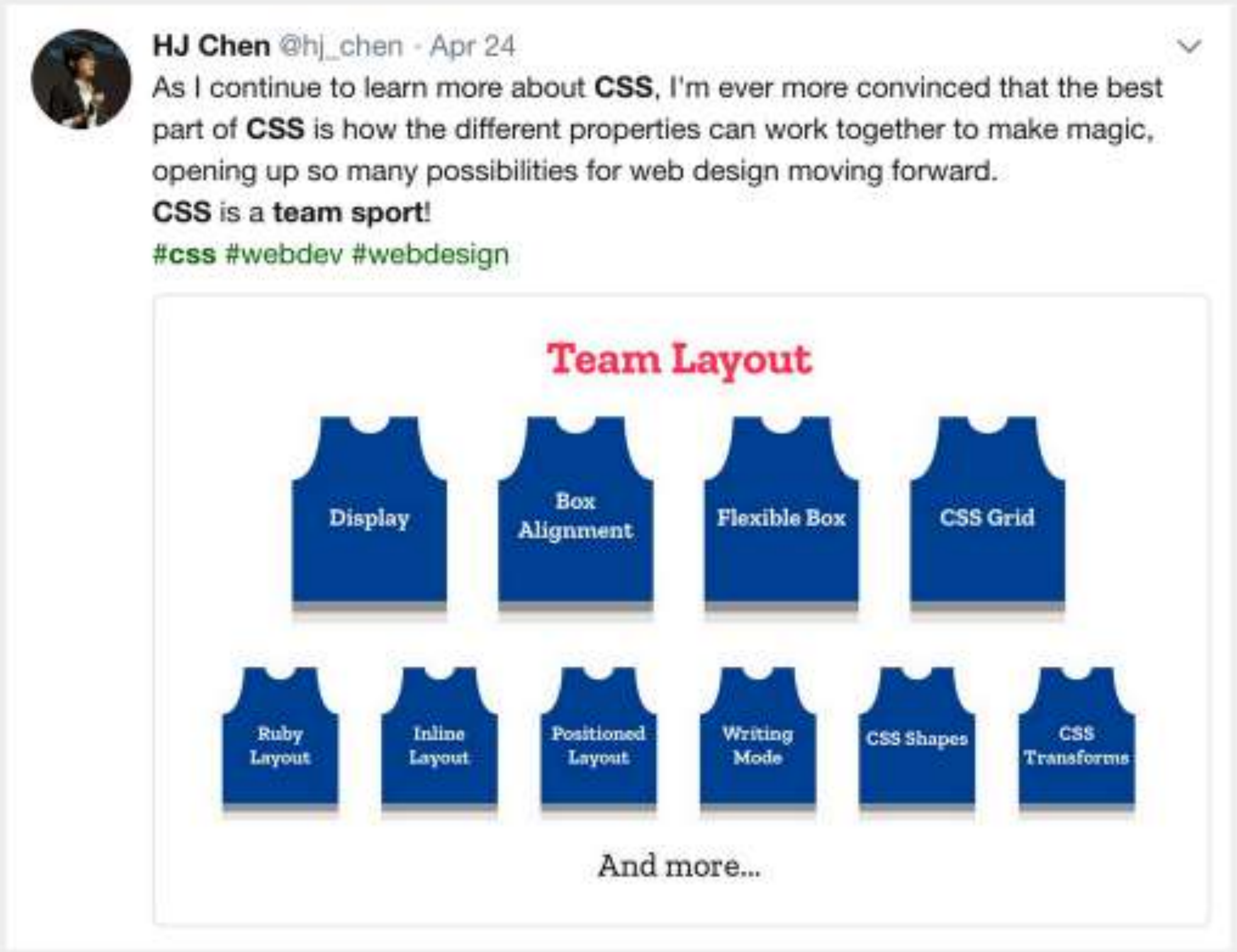
10.2.2010 TORONTO
THE DRAKE HOTEL

READ MORE

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