@rachelandrew #cssday

WHAT I DISCOVERED ABOUT LAYOUT VIA CSS GRID

2013
2014
2015
Venue
Programme
Attendees
Contact

CSS DAY

12th of June, 2015

Compagnietheater, Amsterdam

See our 2017 edition!

```
CSS Day is a one-day advanced CSS conference.

#speakers {
    Roy Tomeij: "sass";
    Stephen Hay: "best practices";
    Clarissa Peterson: "responsive color";
    Paul Robert Lloyd: "responsive principles";
    John Daggett: "typography";
    Rachel Andrew: "grid layout";
    Zoe Mickley Gillenwater: "flexbox";
    fantasai: "auto";
}

Our attendees have been working with CSS for years now. They
```

March 2017

March 2017

March 2017

March 2017

March 2017

Soooooon!

CSS Grid Layout - CR

Global 63.92% + 5.39% = 69.3%

unprefixed: 63.92%

63.92%

Method of using a grid concept to lay out content, providing a mechanism for authors to divide available space for layout into columns and rows using a set of predictable sizing behaviors

Current aligned | Usage relative | Date relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
			🟥 [1] 49						
			🟥 [1] 56						
			🟩 [4] 57			🟥 9.2		🟥 4.4	
	🟨 [2] 14	🟩 [4] 52	🟩 58			🟥 10.2		🟥 4.4.4	
🟨 [2] 11	🟨 [2] 15	🟩 [4] 53	🟩 59	🟩 10.1	🟩 45	🟩 10.3	🟥 all	🟥 56	🟩 59
	🟨 [2] 16	🟩 54	🟩 60	🟩 11	🟩 46	🟩 11			
		🟩 55	🟩 61	🟩 TP	🟩 47				
		🟩 56	🟩 62						

Notes | Known issues (0) | Resources (11) | Feedback

Supported in WebKit Nightly with `-webkit-` prefix.

[1] Enabled in Chrome through the "experimental Web Platform features" flag in chrome://flags

It’s not CSS Vaporware!*

* I’m very happy about this

The more I know about CSS, the more I realise I don’t know.

CSS Grid and friends

- CSS Display
- Writing Modes
- Logical Properties
- Box Alignment
- Feature Queries

CSS Display: https://drafts.csswg.org/css-display/

“This module describes how the CSS formatting box tree is generated from the document element tree and defines the display property that controls it.”

CSS Display: https://drafts.csswg.org/css-display/

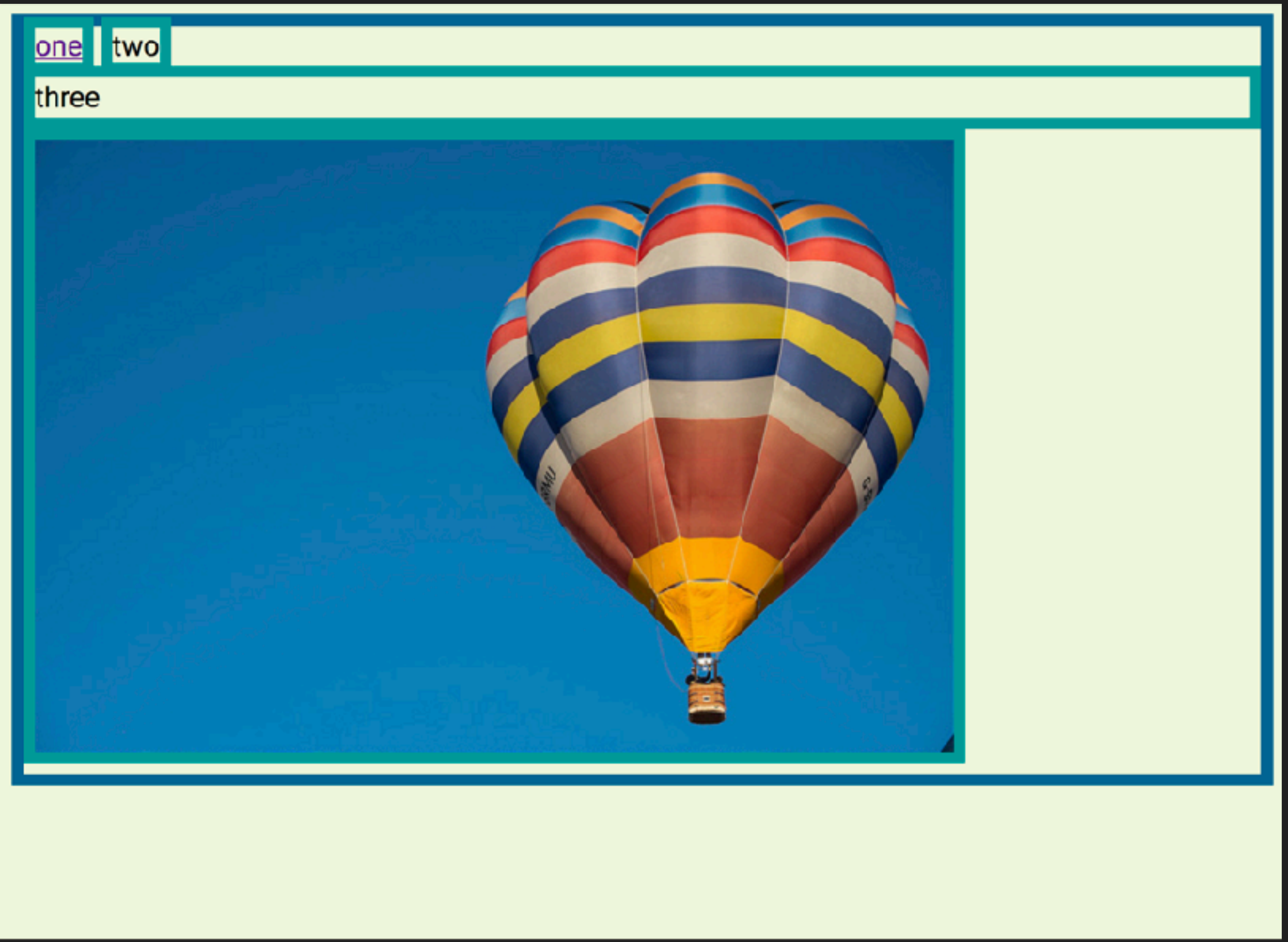
- **The Outer Display Type** - how does this box behave in relationship to its parent?
- **The Inner Display Type** - what formatting context does it create for its child elements?

CSS Grid: https://www.w3.org/TR/css-grid-1/#grid-item-display

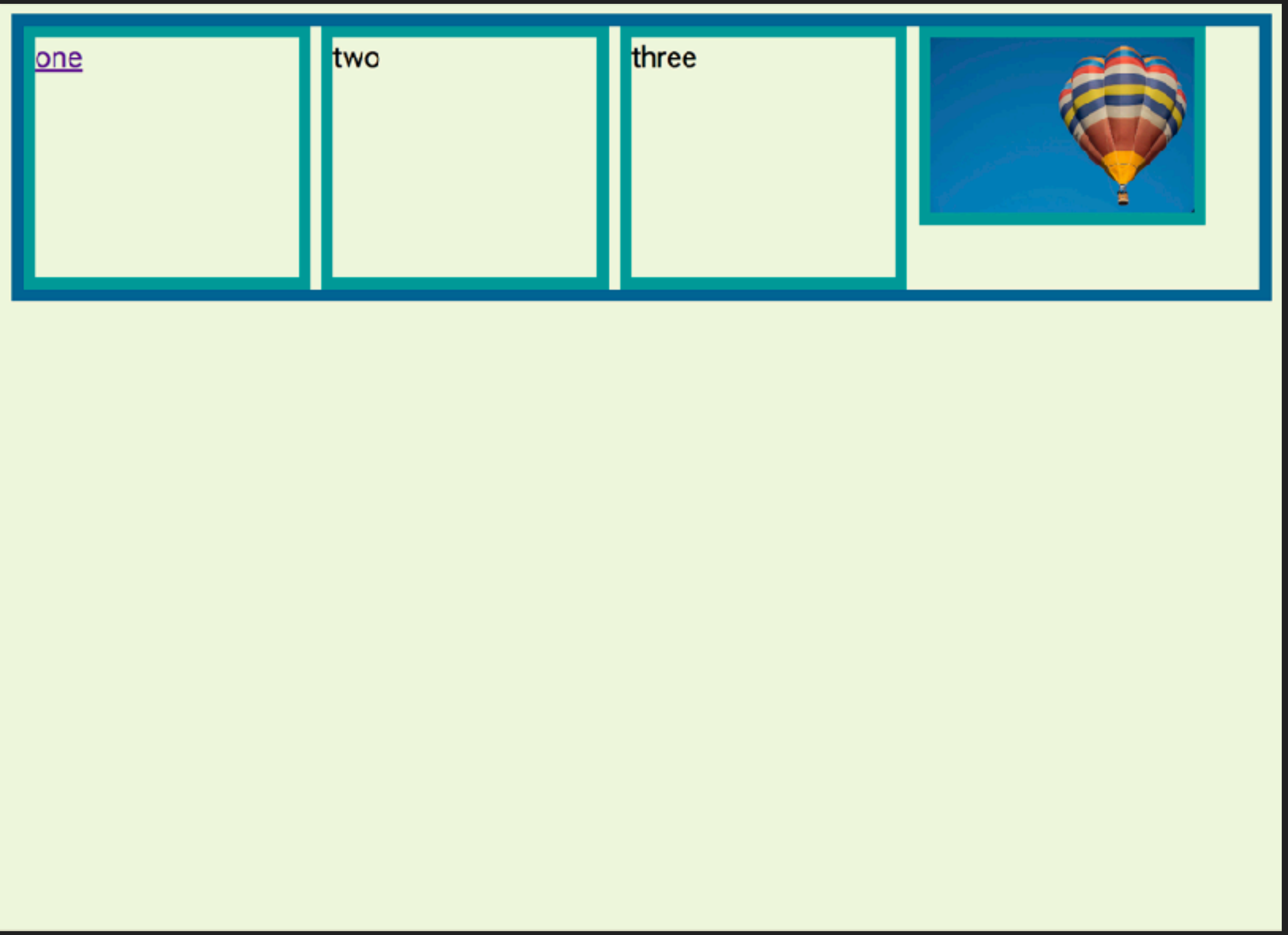
“The display value of a grid item is blockified: if the specified display of an in-flow child of an element generating a grid container is an inline-level value, it computes to its block-level equivalent. ”

CSS Display: https://www.w3.org/TR/css-display/#transformations

“Some layout effects
require ***blockification*** or ***inlinification*** of the box
type, which sets the box’s outer display type, if it
is not none or contents,
to block or inline (respectively).”



```
<div class="grid">
  <a href="">one</a>
  <span>two</span>
  <div>three</div>
  <img src="img.png" alt="placeholder">
</div>
```



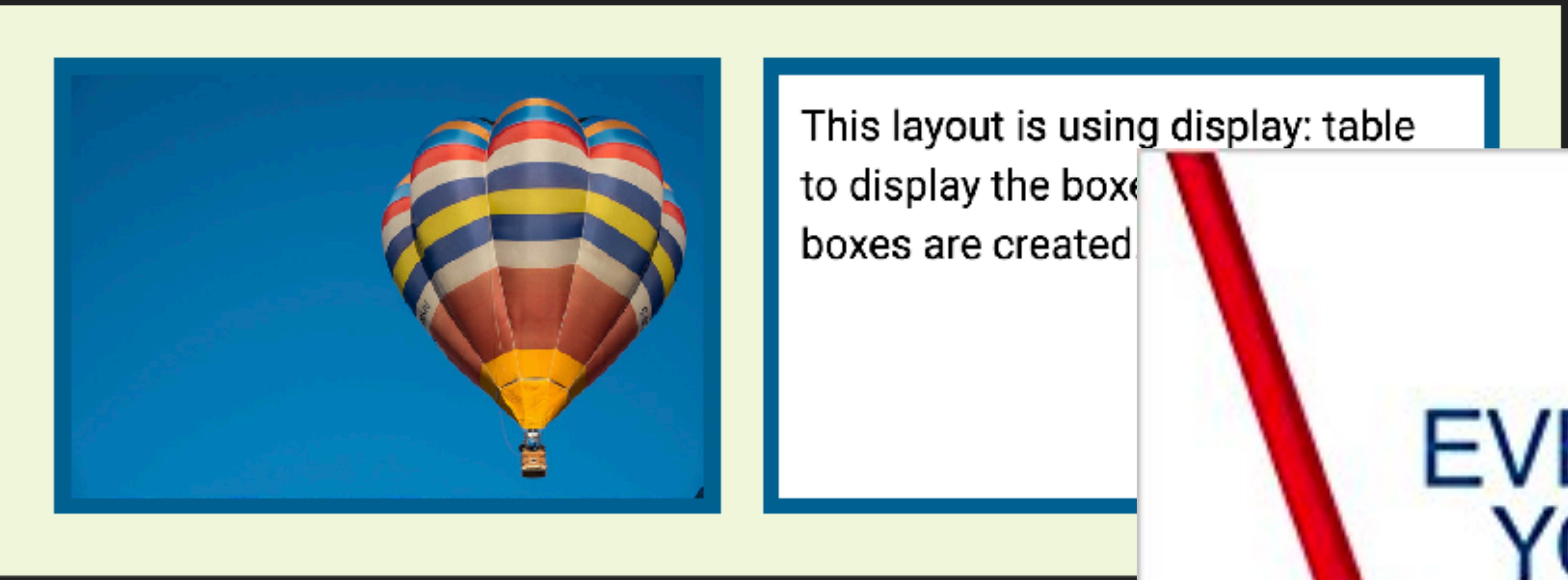
```
.grid {
  display: grid;
  grid-template-columns: repeat(4,200px);
  grid-gap: 8px;
  height: 200px;
  border: 8px solid rgb(3,99,143);
}
```

Why is knowing this useful?

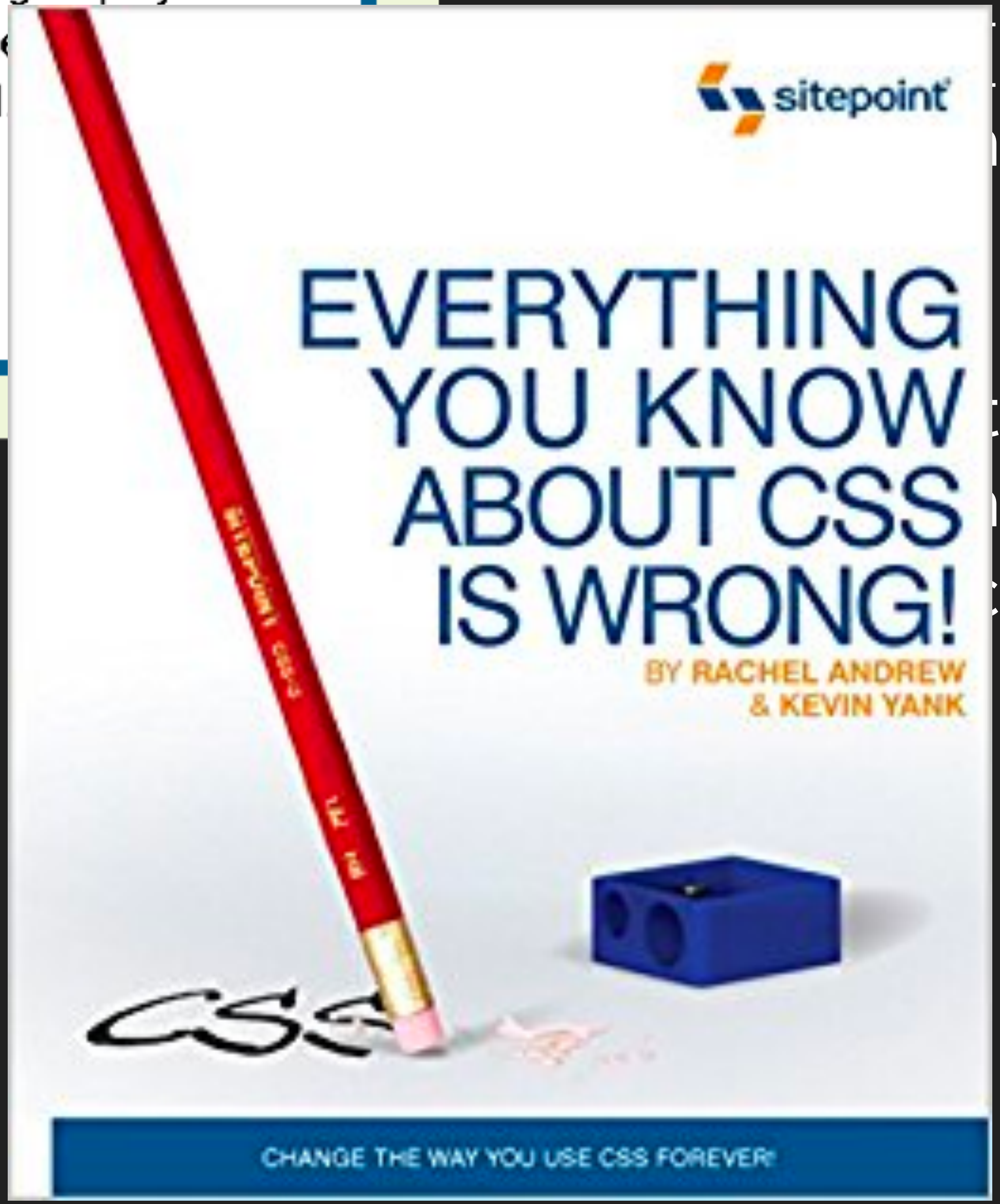
```
.wrapper {
  max-width: 800px;
  border-spacing: 20px;
}
```

```
y: table-cell;
```

```
: {
y: table-cell;
:al-align: top;
```



This layout is using display: table to display the boxe boxes are created



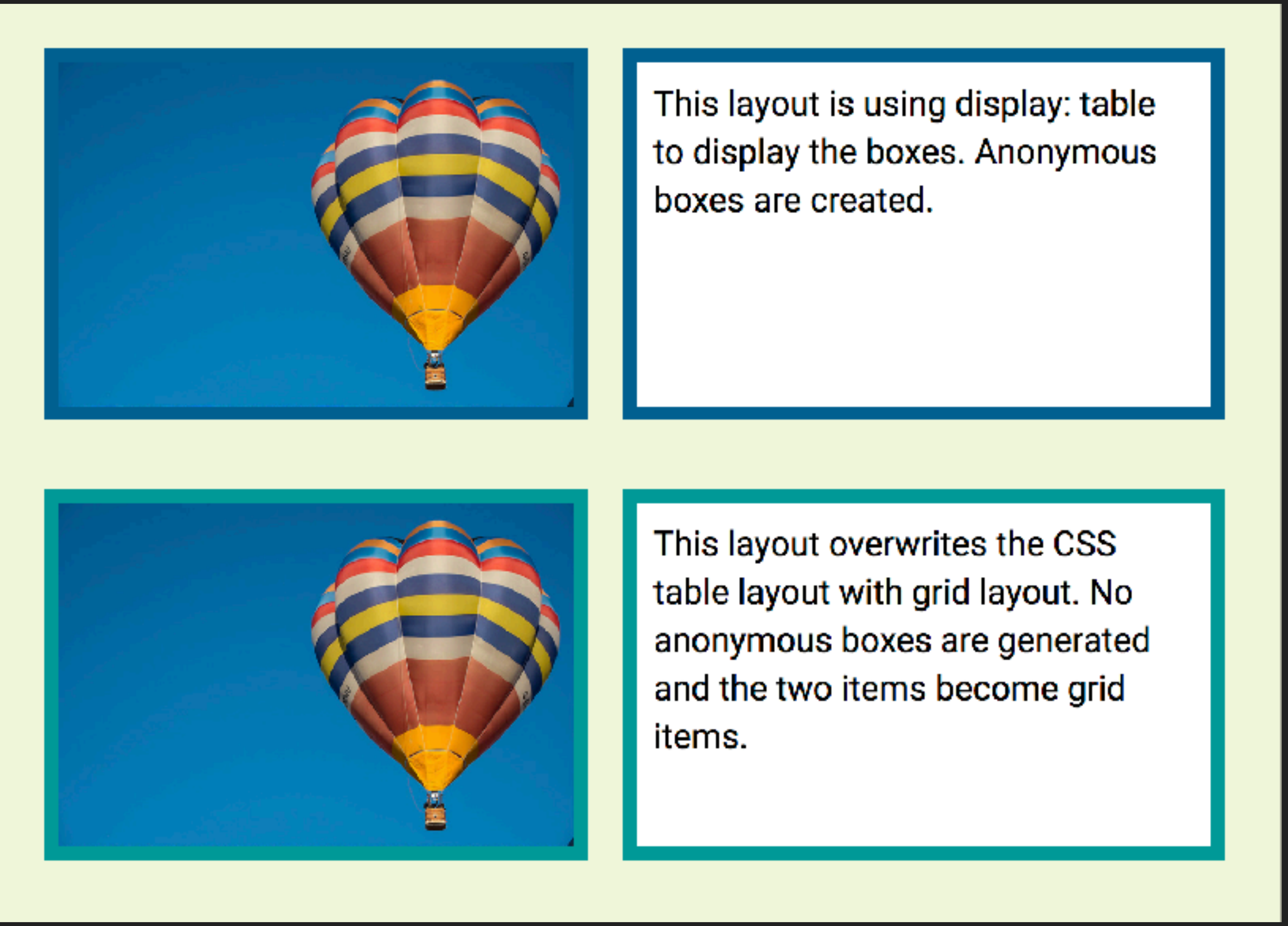
sitepoint
EVERYTHING YOU KNOW ABOUT CSS IS WRONG!
BY RACHEL ANDREW & KEVIN YANK
CHANGE THE WAY YOU USE CSS FOREVER!

https://www.w3.org/TR/CSS2/tables.html#anonymous-boxes

“Any table element will automatically generate necessary anonymous table objects around itself, consisting of at least three nested objects corresponding to a 'table'/'inline-table' element, a 'table-row' element, and a 'table-cell' element.”

CSS Grid: https://www.w3.org/TR/css-grid-1/#grid-item-display

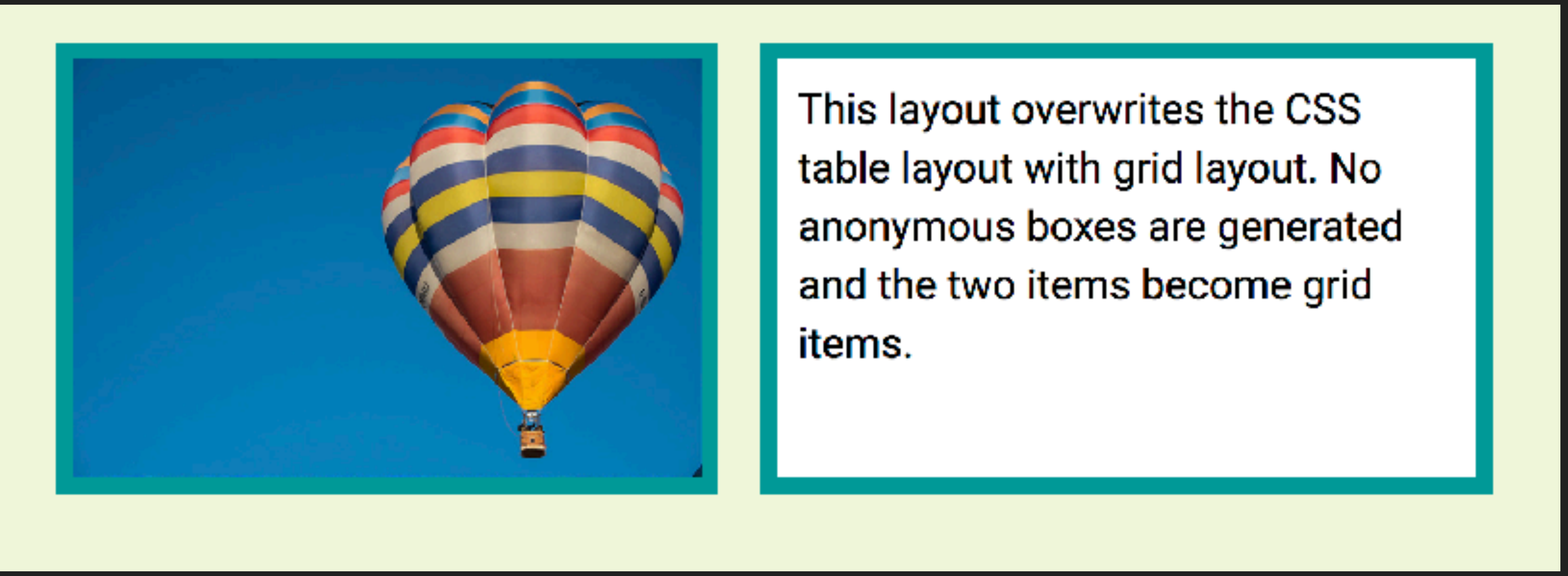
“Note: Some values of display normally trigger the creation of anonymous boxes around the original box. If such a box is a grid item, it is blockified first, and so anonymous box creation will not happen. For example, two contiguous grid items with display: table-cell will become two separate display: block grid items, instead of being wrapped into a single anonymous table.”


This layout is using display: table to display the boxes. Anonymous boxes are created.

This layout overwrites the CSS table layout with grid layout. No anonymous boxes are generated and the two items become grid items.

```
.wrapper {
  max-width: 800px;
  border-spacing: 20px;
  display: grid;
  grid-template-columns: auto 1fr;
  grid-gap: 20px;
}

.image {
  display: table-cell;
}

.content {
  display: table-cell;
  vertical-align: top;
}
```

https://codepen.io/rachelandrew/pen/KqMyzN



This layout overwrites the CSS table layout with grid layout. No anonymous boxes are generated and the two items become grid items.

```
.grid {
  max-width: 800px;
  border-spacing: 20px;
  display: grid;
  grid-template-columns: auto 1fr;
  grid-gap: 20px;
}

@supports (grid-gap: 20px) {
   .grid {
    margin: 20px;
  }
}

.image {
  display: table-cell;
}

.content {
  display: table-cell;
  vertical-align: top;
}
```

https://codepen.io/rachelandrew/pen/qjNVwG

Creating fallbacks

- You **do not** need to write two sets of code
- Write your fallback code and then write your grid code
- In many cases **the spec has you covered**
- Use Feature Queries to isolate things that would apply to both grid-supporting and non-supporting browsers

rachelandrew.co.uk ./CSS

Follow 21.5K followers

Grid “fallbacks” and overrides

Defined in the CSS Grid Specification are the ways in which grid interacts with other layout methods. These definitions mean that as soon as an item becomes a grid item, other behaviour that you may have used for older browsers is overwritten. This means that you do not have to completely fork your code and build two versions. Where you do need to overwrite CSS used for older browsers, you can do so inside a CSS Feature Query. This enables safe encapsulation of any CSS you only want a grid supporting browser to apply.

Here is a quick reference to the defined overrides with simple examples. See the resources at the end of this cheatsheet for more advice and examples.

Floated items

Float and clear have no effect on a grid item

If you float an item, in your CSS for non-grid browsers, when that item becomes a grid item float ceases to have any effect. The clear property applied to an item that becomes a grid item also no longer takes effect.

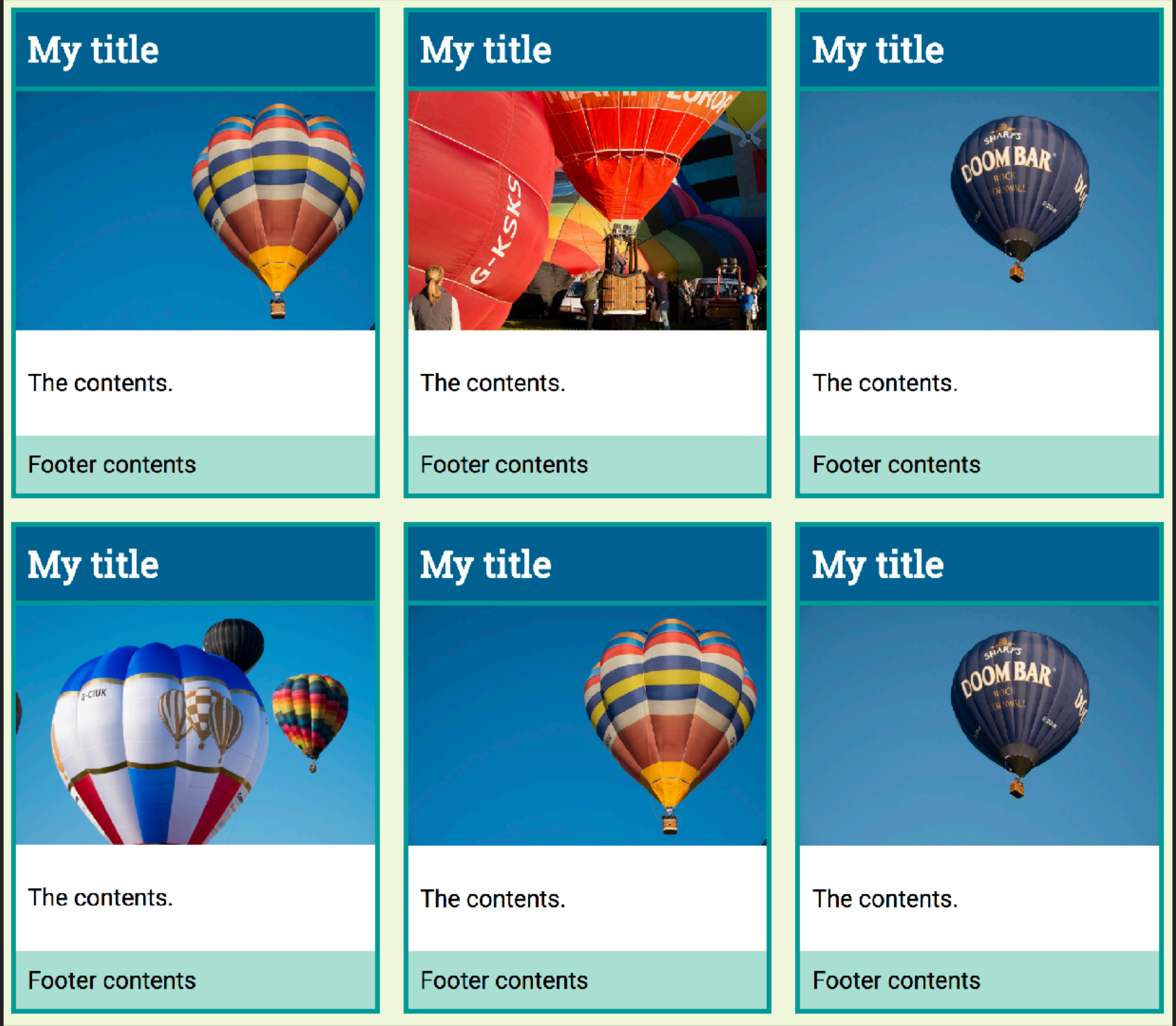
Example

```
<div class="grid">
  <div>One</div>
  <div>Two</div>
  <div>Three</div>
</div>
```

```
.grid > div {
  float: left;
}
```

https://rachelandrew.co.uk/css/cheatsheets/grid-fallbacks

What happened to subgrid?



My title

The contents.

Footer contents

My title

The contents.

Footer contents

My title

The contents.

Footer contents

My title

The contents.

Footer contents

My title

The contents.

Footer contents

My title

The contents.

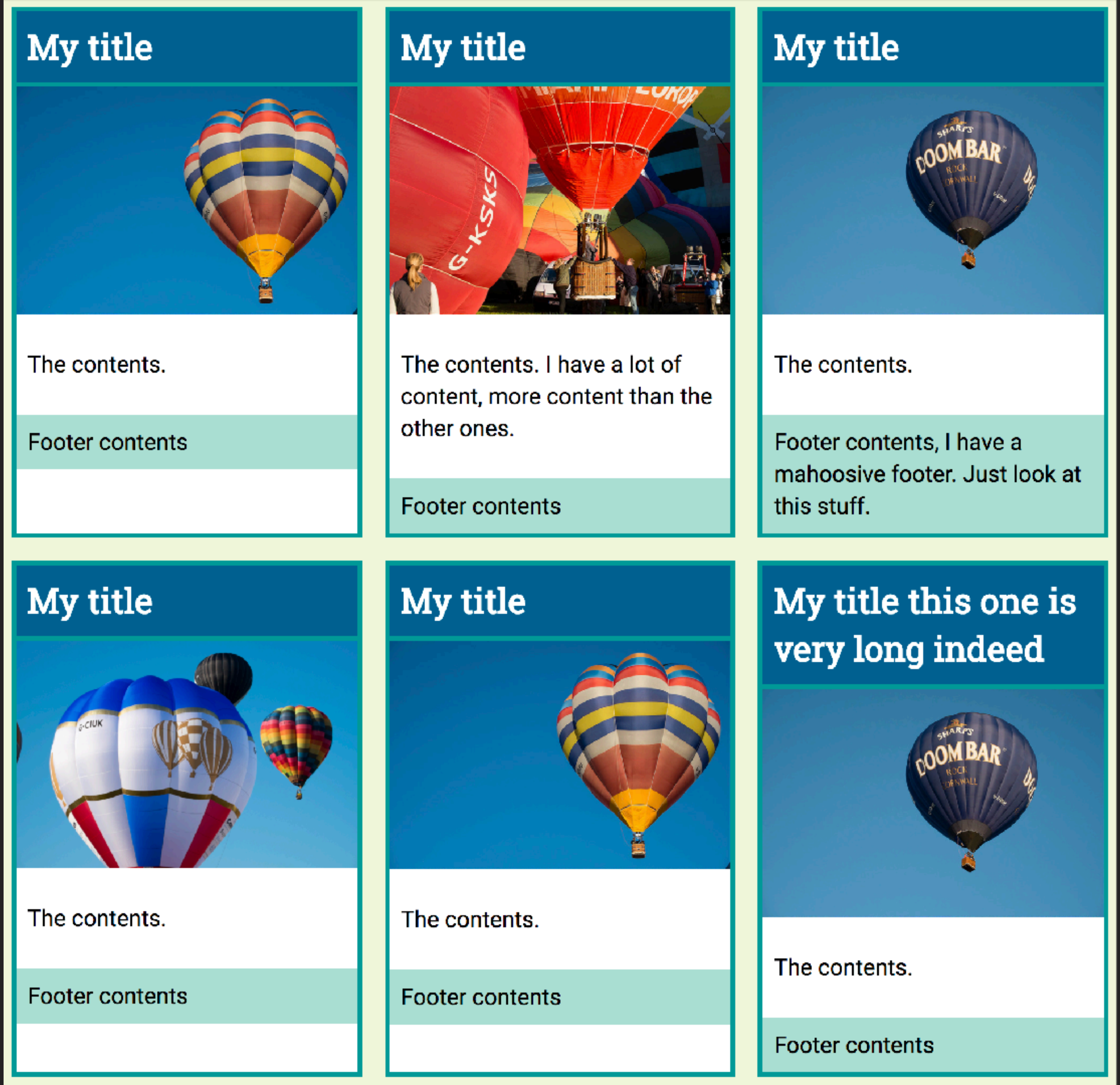
Footer contents

CSS Grid

Creating a three column layout with CSS Grid.

```
.grid {
  display: grid;
  max-width: 960px;
  margin: 0 auto;
  grid-template-columns: repeat(3, 1fr);
  grid-gap: 20px;
}
```

https://codepen.io/rachelandrew/pen/XgdydE



My title

The contents.

Footer contents

My title

The contents. I have a lot of content, more content than the other ones.

Footer contents

My title

The contents.

Footer contents, I have a mahoosive footer. Just look at this stuff.

My title

The contents.

Footer contents

My title

The contents.

Footer contents

My title this one is very long indeed

The contents.

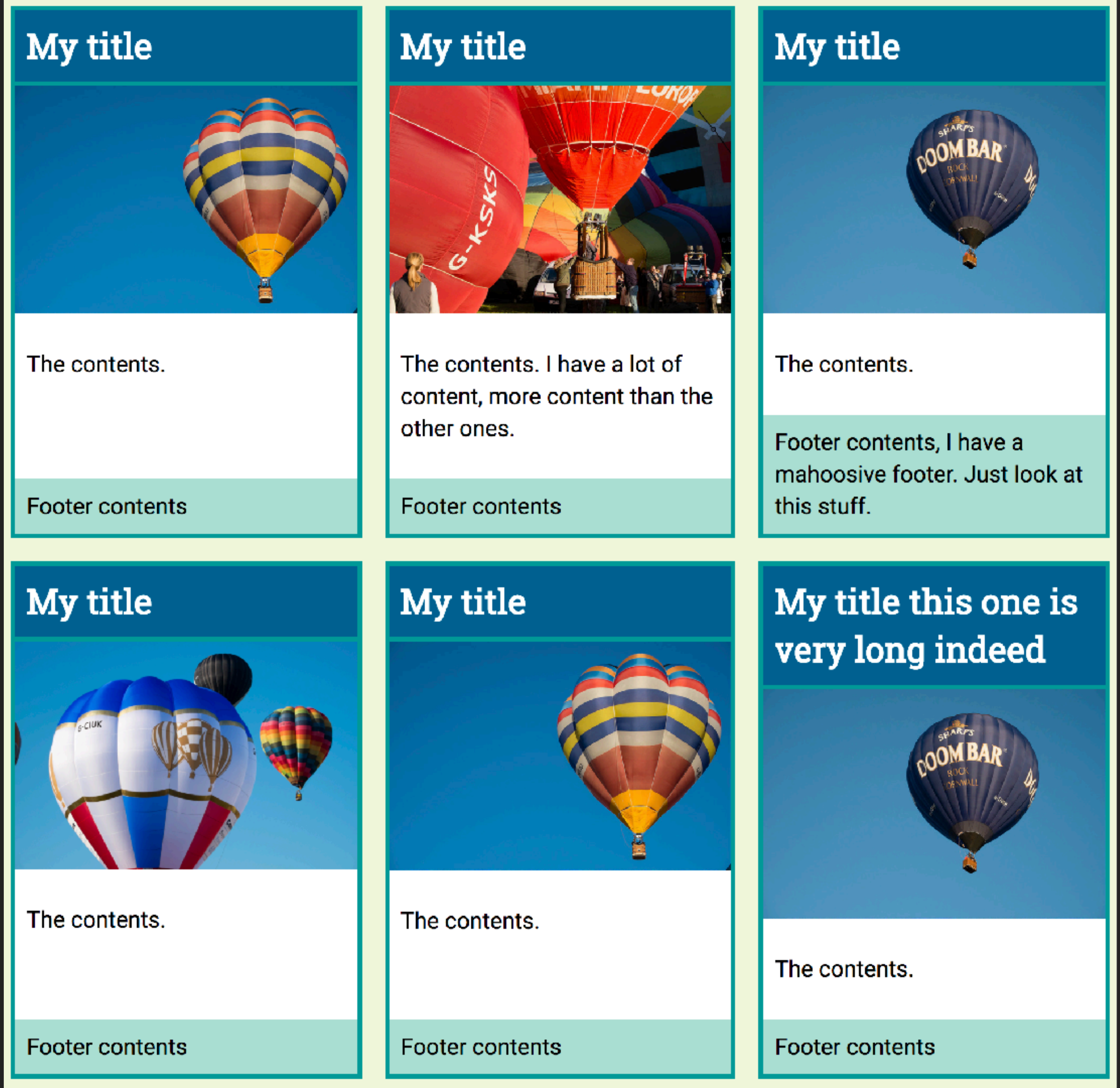
Footer contents

Make the card a flex item

Allow the inner to grow, it pushes the footer down to the bottom of the card.s

```
.card {
  display: flex;
  flex-direction: column;
}

.card .inner {
  flex: 1;
}
```

https://codepen.io/rachelandrew/pen/XgdydE



My title

The contents.

Footer contents

My title

The contents. I have a lot of content, more content than the other ones.

Footer contents

My title

The contents.

Footer contents, I have a mahoosive footer. Just look at this stuff.

My title

The contents.

Footer contents

My title

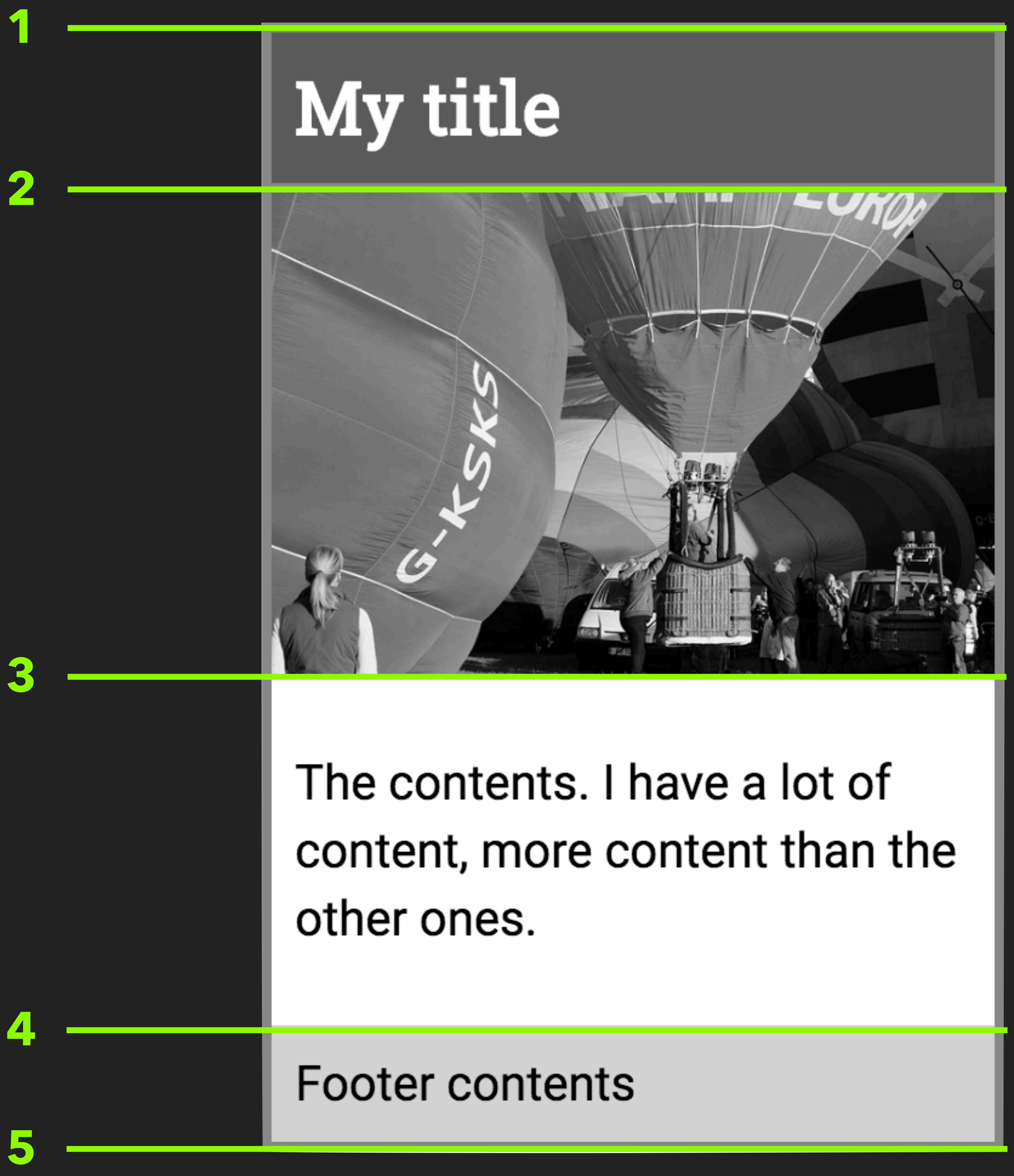
The contents.

Footer contents

My title this one is very long indeed

The contents.

Footer contents


My title

The contents. I have a lot of content, more content than the other ones.

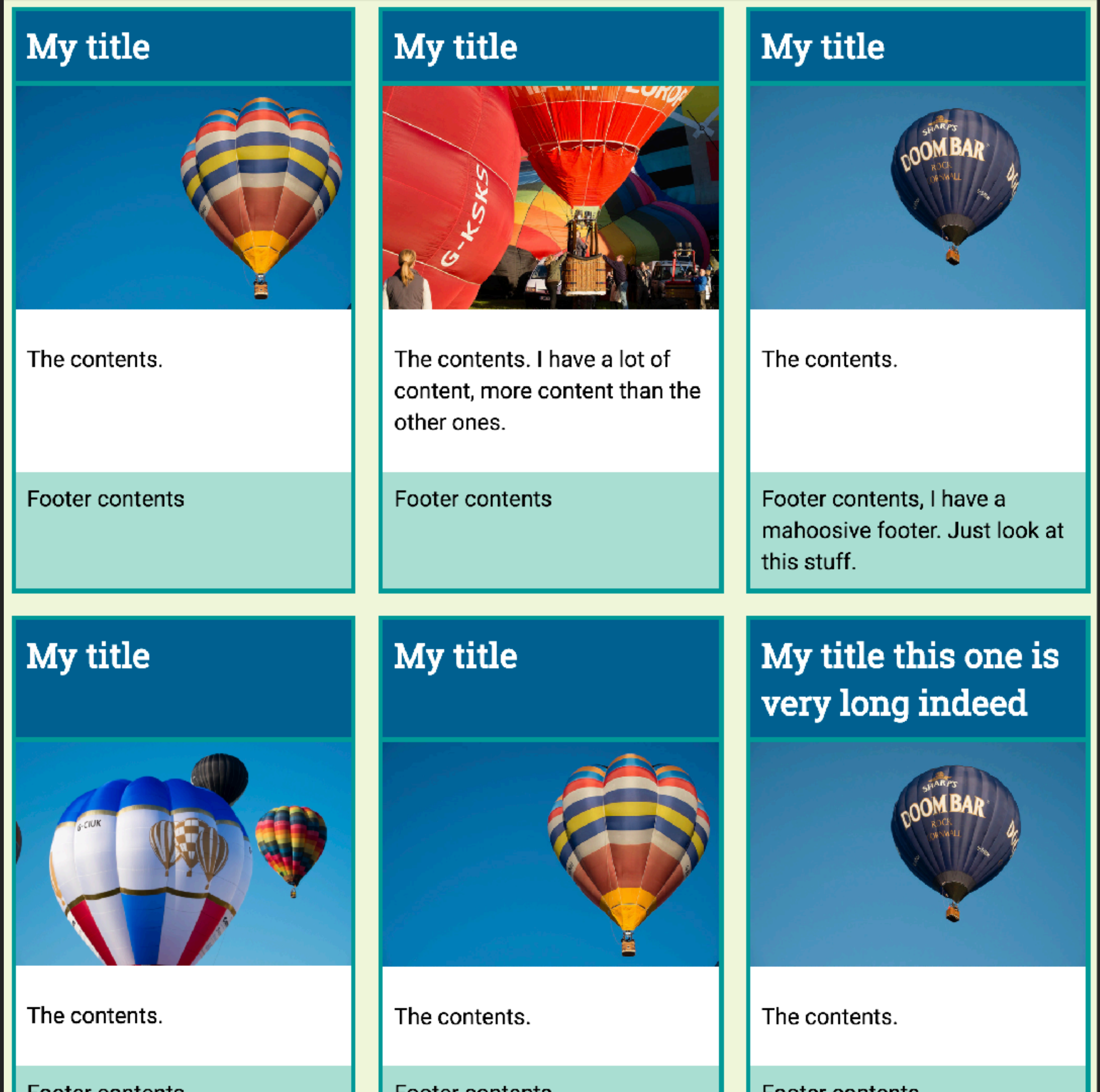
Footer contents

display: subgrid

The card is a direct child of the grid so needs to span four rows of the grid to make room for the four rows in the subgridded internals.

display: subgrid means the card now uses the tracks of the parent grid.

```
.card {
  border: 4px solid rgb(24,154,153);
  background-color: #fff;
  grid-row: auto / span 4;
  display: subgrid;
}
```



My title

The contents.

Footer contents

My title

The contents. I have a lot of content, more content than the other ones.

Footer contents

My title

The contents.

Footer contents, I have a mahoosive footer. Just look at this stuff.

My title

The contents.

Footer contents

My title

The contents.

Footer contents

My title this one is very long indeed

The contents.

Footer contents

Subgrid links and thoughts

- https://rachelandrew.co.uk/archives/2017/03/16/subgrid-moved-to-level-2-of-the-css-grid-specification/
- https://github.com/w3c/csswg-drafts/issues/958
- https://github.com/rachelandrew/cssgrid-ama/issues/13
- http://meyerweb.com/eric/thoughts/2016/01/15/subgrids-considered-essential/

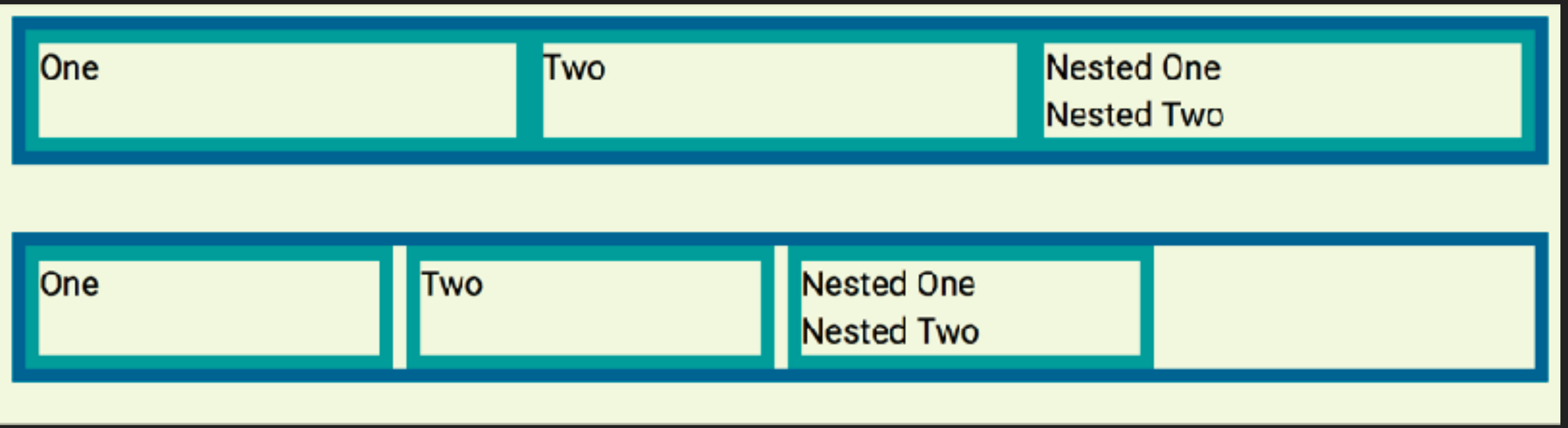
Vanishing boxes with display:contents

display: contents https://drafts.csswg.org/css-display/#box-generation

“The element itself does not generate any boxes, but its children and pseudo-elements still generate boxes as normal. For the purposes of box generation and layout, the element must be treated as if it had been replaced in the element tree by its contents”

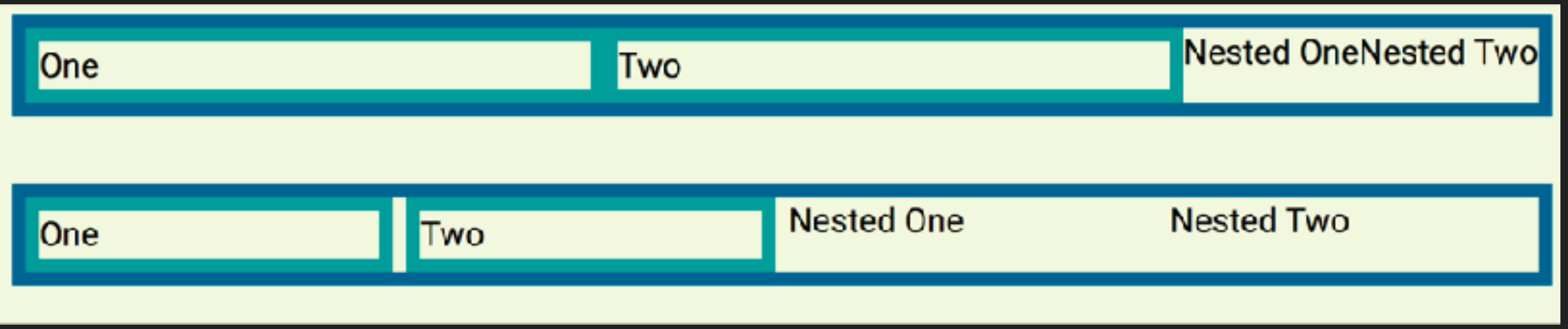
```html
<div class="flex">
  <div>One</div>
  <div>Two</div>
  <div class="nested">
    <div>Nested One</div>
    <div>Nested Two</div>
  </div>
</div>

<div class="grid">
  <div>One</div>
  <div>Two</div>
  <div class="nested">
    <div>Nested One</div>
    <div>Nested Two</div>
  </div>
</div>
```

https://codepen.io/rachelandrew/pen/GEZPex



```
.flex {
  display: flex;
  border: 8px solid rgb(3,99,143);
}
.flex > * {
  flex: 1;
  border: 8px solid rgb(24,154,153);
}

.grid {
  display: grid;
  border: 8px solid rgb(3,99,143);
  grid-template-columns:
repeat(4,minmax(200px, 1fr));
  grid-gap: 8px;
}
.grid > * {
  border: 8px solid rgb(24,154,153);
}
```

https://codepen.io/rachelandrew/pen/GEZPex



```
.nested {
 display: contents;
}
```

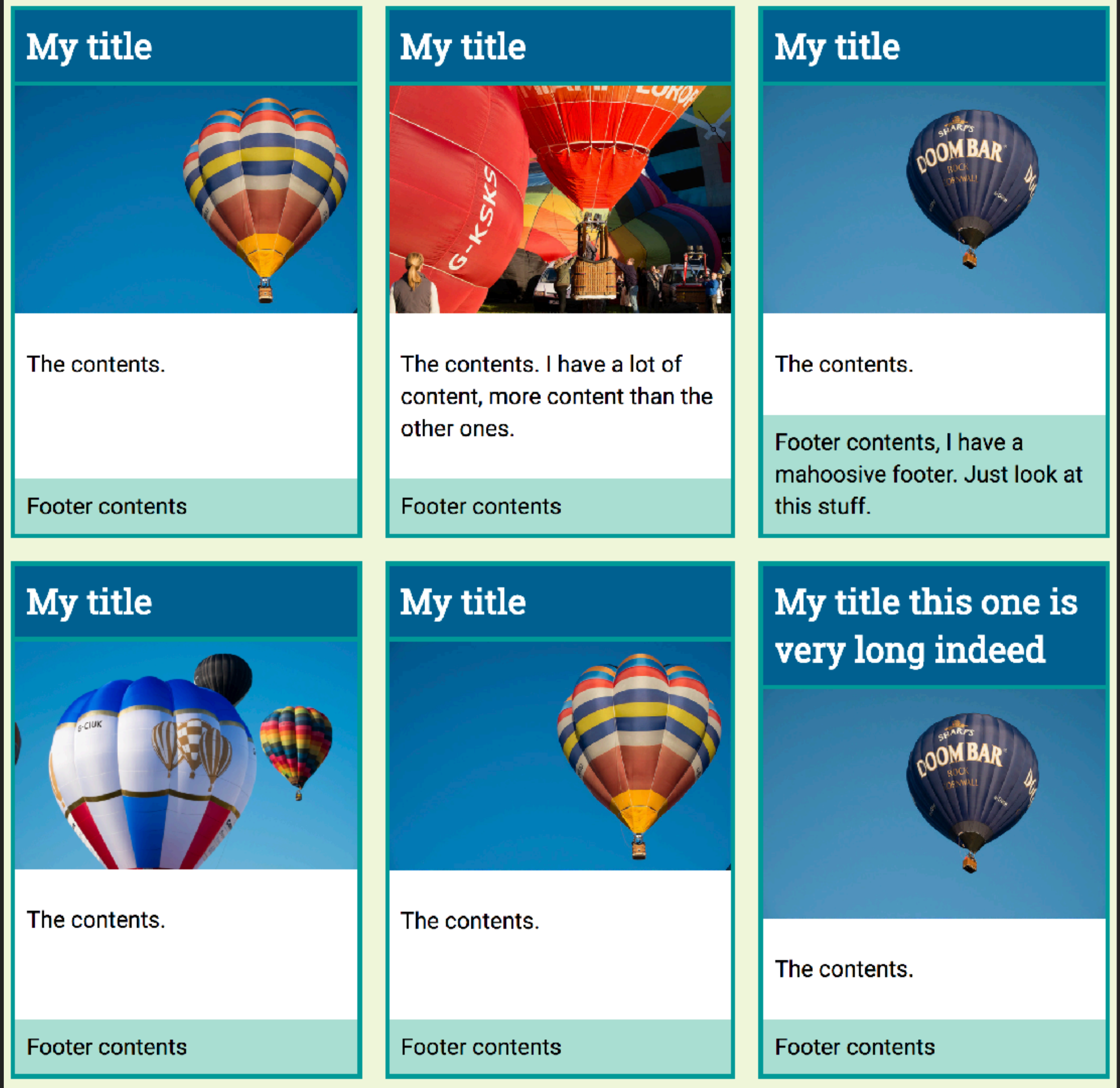
Needs Firefox or Chrome Canary

https://codepen.io/rachelandrew/pen/GEZPex

```
.flex > * {
  flex: 1;
  border: 8px solid rgb(24,154,153);
}
```

One
Two
Nested OneNested Two
One
Two
Nested One
Nested Two

```
.grid > * {
  border: 8px solid rgb(24,154,153);
}
```



My title

The contents.

Footer contents

My title

The contents. I have a lot of content, more content than the other ones.

Footer contents

My title

The contents.

Footer contents, I have a mahoosive footer. Just look at this stuff.

My title

The contents.

Footer contents

My title

The contents.

Footer contents

My title this one is very long indeed

The contents.

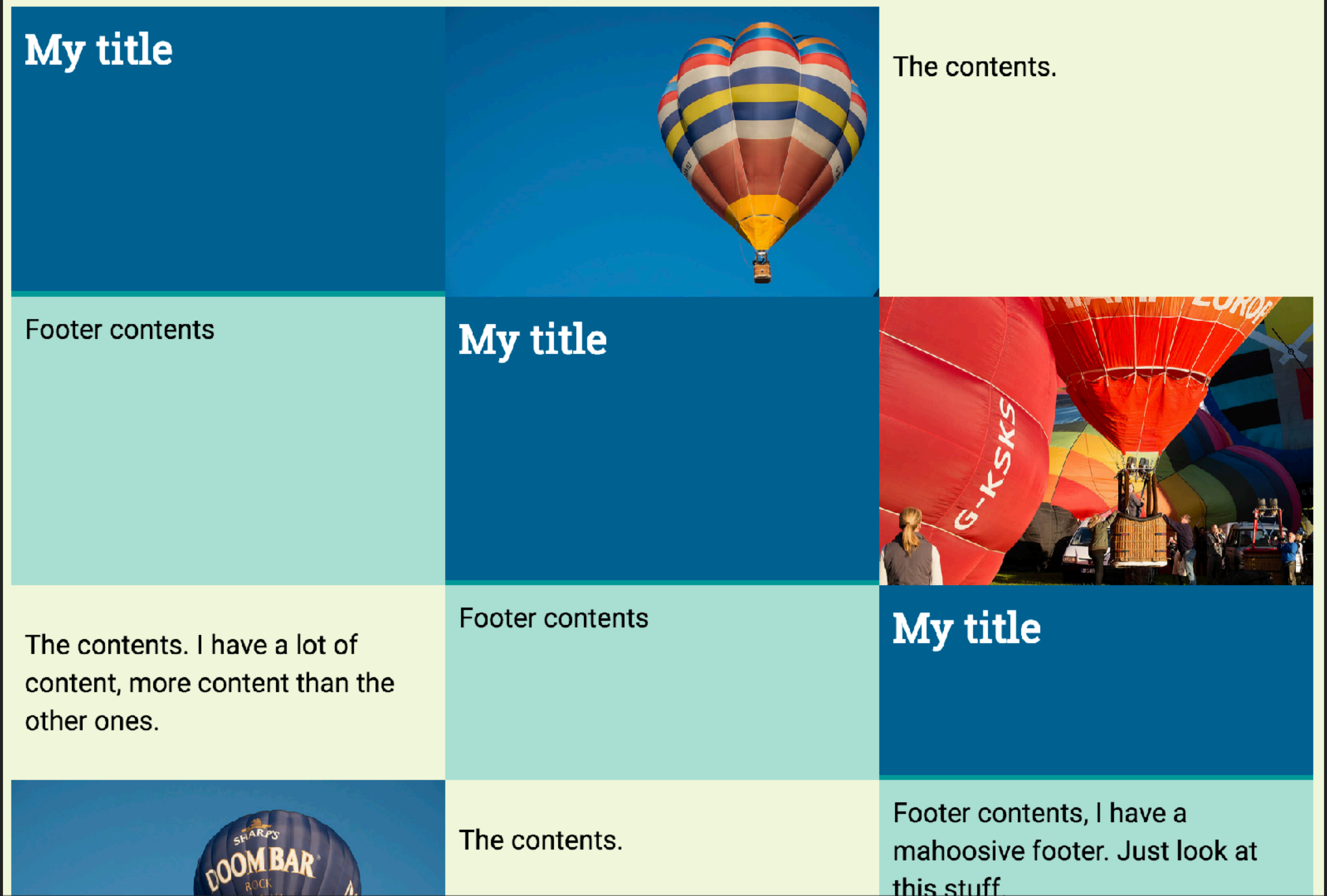
Footer contents

display: contents

We add this to the direct child of the grid container.

```
.card {
  border: 4px solid rgb(24,154,153);
  background-color: #fff;
  display: contents;
}
```

https://codepen.io/rachelandrew/pen/QgNJYa



My title

The contents.

Footer contents

My title

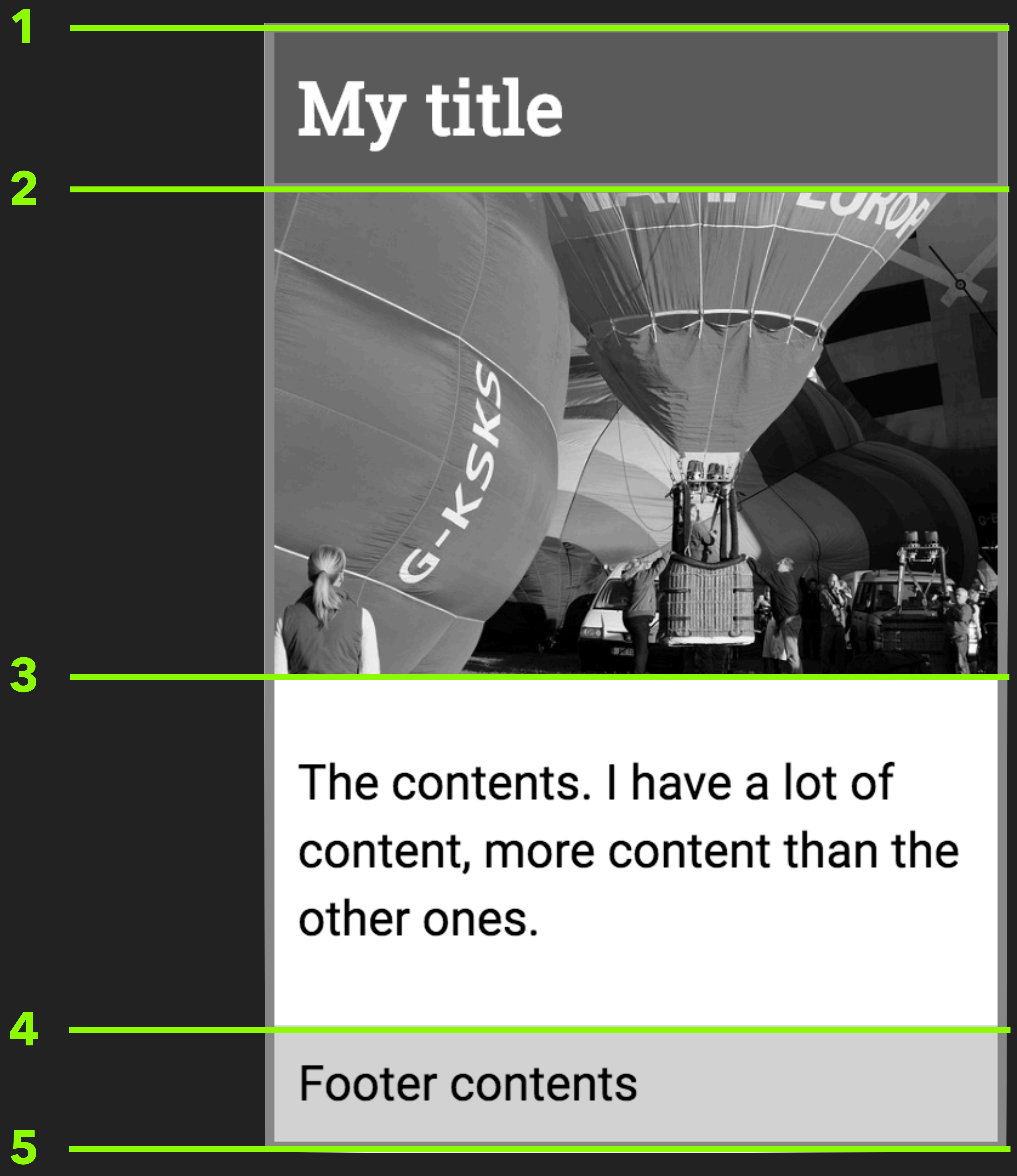
The contents. I have a lot of content, more content than the other ones.

Footer contents

My title

The contents.

Footer contents, I have a mahoosive footer. Just look at this stuff.



My title

The contents. I have a lot of content, more content than the other ones.

Footer contents

Make room for the rows

Each card needs four rows.

```
.card {
  border: 4px solid rgb(24,154,153);
  background-color: #fff;
  grid-row: auto / span 4;
  display: contents;
}
```

Ugh.

Don’t do this.

```
.card:nth-child(1) h2{ grid-column: 1; grid-row: 1; }
.card:nth-child(1) img{ grid-column: 1; grid-row: 2; }
.card:nth-child(1) .inner{ grid-column: 1; grid-row: 3; }
.card:nth-child(1) footer{ grid-column: 1; grid-row: 4; }
.card:nth-child(2) h2{ grid-column: 2; grid-row: 1; }
.card:nth-child(2) img{ grid-column: 2; grid-row: 2; }
.card:nth-child(2) .inner{ grid-column: 2; grid-row: 3; }
.card:nth-child(2) footer{ grid-column: 2; grid-row: 4; }
.card:nth-child(3) h2{ grid-column: 3; grid-row: 1; }
.card:nth-child(3) img{ grid-column: 3; grid-row: 2; }
.card:nth-child(3) .inner{ grid-column: 3; grid-row: 3; }
.card:nth-child(3) footer{ grid-column: 3; grid-row: 4; }
.card:nth-child(4) h2{ grid-column: 1; grid-row: 5; }
.card:nth-child(4) img{ grid-column: 1; grid-row: 6; }
.card:nth-child(4) .inner{ grid-column: 1; grid-row: 7;}
.card:nth-child(4) footer{ grid-column: 1; grid-row: 8; }
.card:nth-child(5) h2{ grid-column: 2; grid-row: 5; }
.card:nth-child(5) img{ grid-column: 2; grid-row: 6; }
.card:nth-child(5) .inner{ grid-column: 2; grid-row: 7; }
.card:nth-child(5) footer{ grid-column: 2; grid-row: 8; }
.card:nth-child(6) h2{ grid-column: 3; grid-row: 5; }
.card:nth-child(6) img{ grid-column: 3; grid-row: 6; }
.card:nth-child(6) .inner{ grid-column: 3; grid-row: 7; }
.card:nth-child(6) footer{ grid-column: 3; grid-row: 8; }
```

https://codepen.io/rachelandrew/pen/QgNJYa

My title

The contents.

Footer contents

My title

The contents. I have a lot of content, more content than the other ones.

Footer contents

My title

The contents.

Footer contents, I have a mahoosive footer. Just look at this stuff.

display: contents

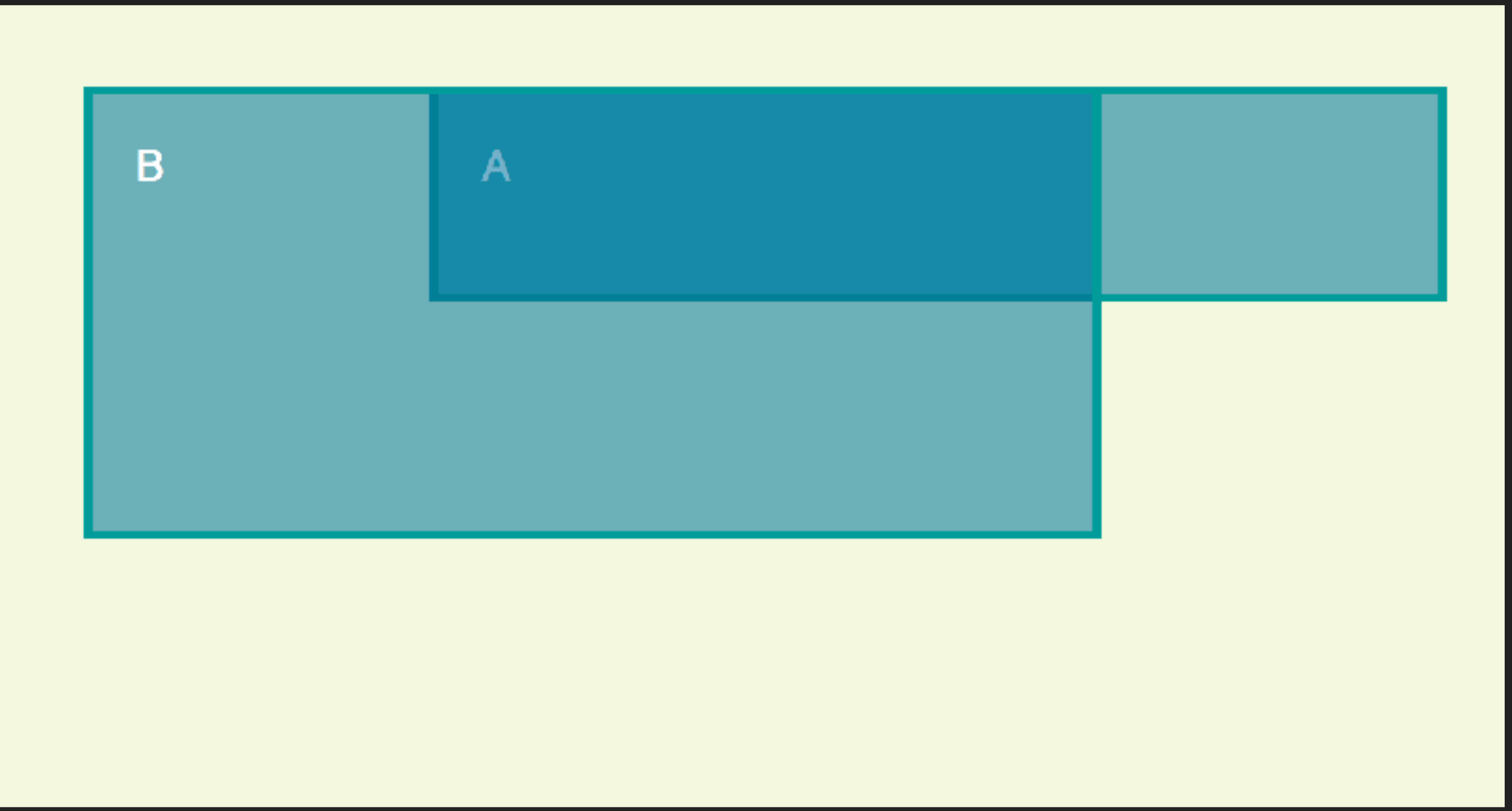
- Use when the element you are removing has no box styling (e.g. backgrounds and borders) attached
- Current browser support Firefox, Chrome Canary

It is all logical.

The order of values in grid-area

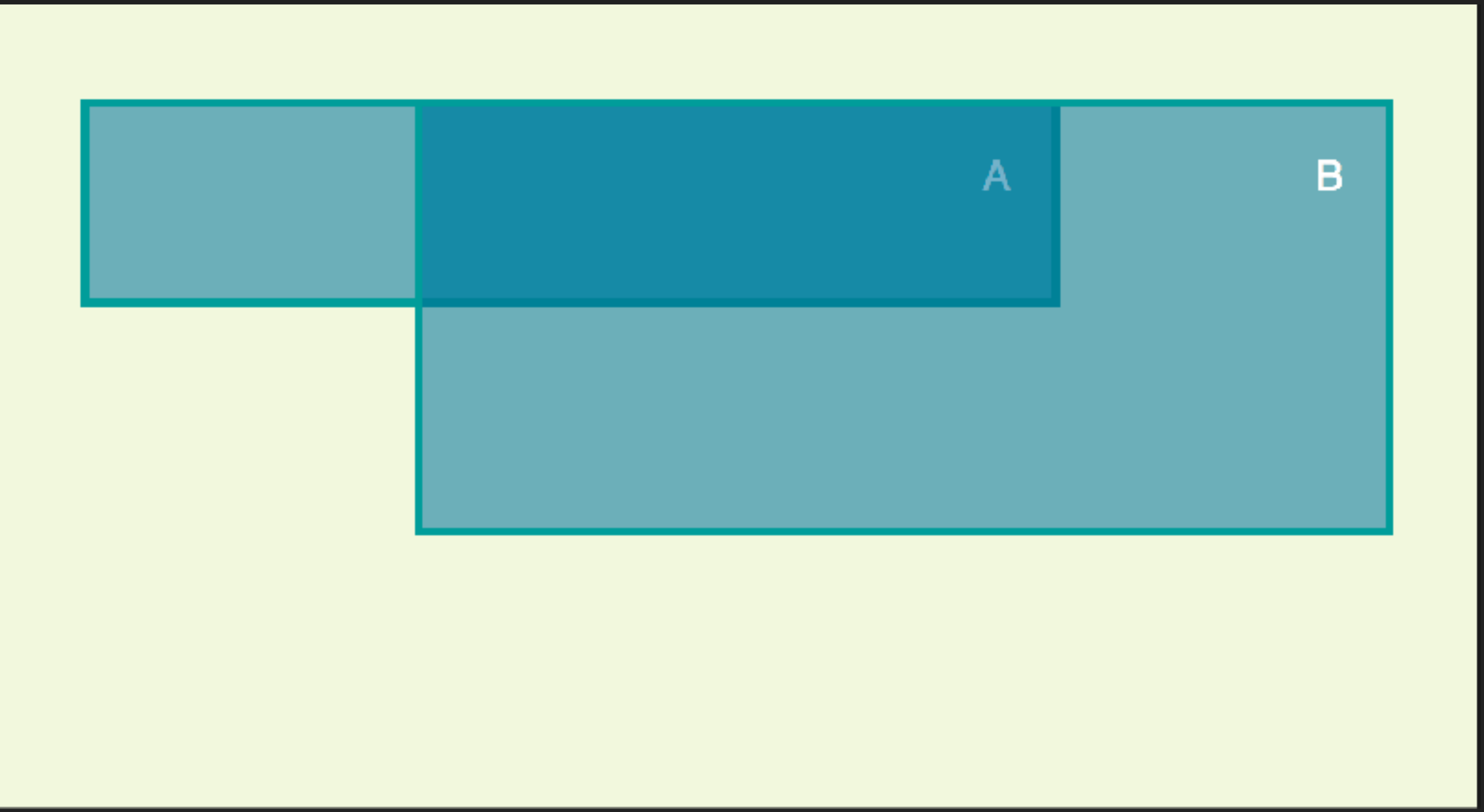
- row-start
- column-start
- row-end
- column-end

```
/* this shorthand */
.a {
  grid-area: 1 / 2 / 2 / 5;
}

/* is the same as this */
.a {
  grid-row-start: 1;
  grid-column-start: 2;
  grid-row-end: 2;
  grid-column-end: 5;
}
```



```
.grid {
  display: grid;
  grid-gap: 10px;
  grid-template-columns: repeat(4, 150px);
  grid-template-rows: repeat(3, 100px);
}

.a {
  grid-area: 1 / 2 / 2 / 5;
}
.b {
  grid-area: 1 / 1 / 3 / 4;
}
```

https://codepen.io/rachelandrew/pen/BZKbaN



```
.grid {
  Direction: rtl;
  display: grid;
  grid-gap: 10px;
  grid-template-columns: repeat(4, 150px);
  grid-template-rows: repeat(3, 100px);
}

.a {
  grid-area: 1 / 2 / 2 / 5;
}
.b {
  grid-area: 1 / 1 / 3 / 4;
}
```

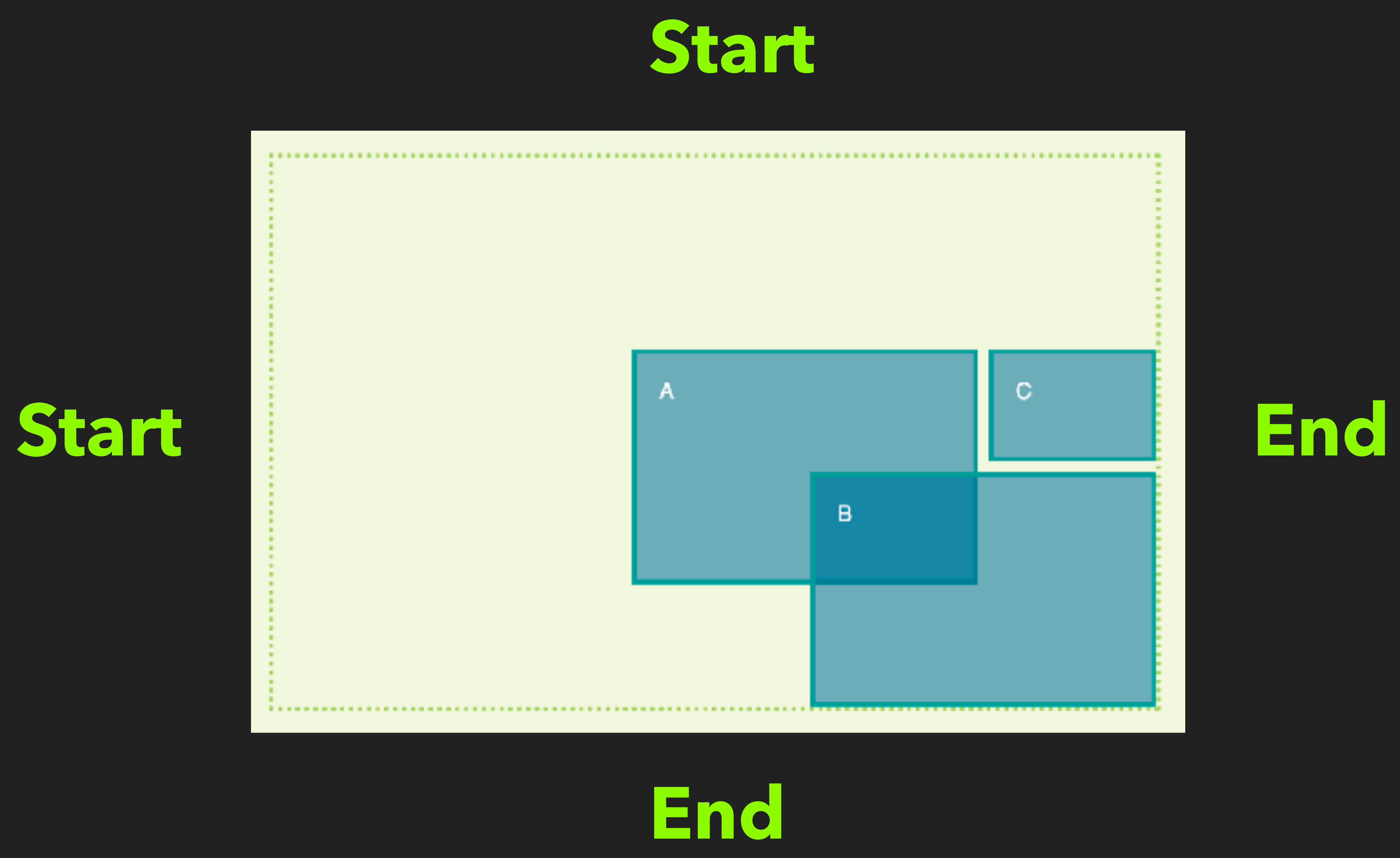
https://codepen.io/rachelandrew/pen/BZKbaN

“This module introduces logical properties and values that provide the author with the ability to control layout through logical, rather than physical, direction and dimension mappings. The module defines logical properties and values for the features defined in CSS2.1. **These properties are writing-mode relative equivalents of their corresponding physical properties.**”

Logical not Physical

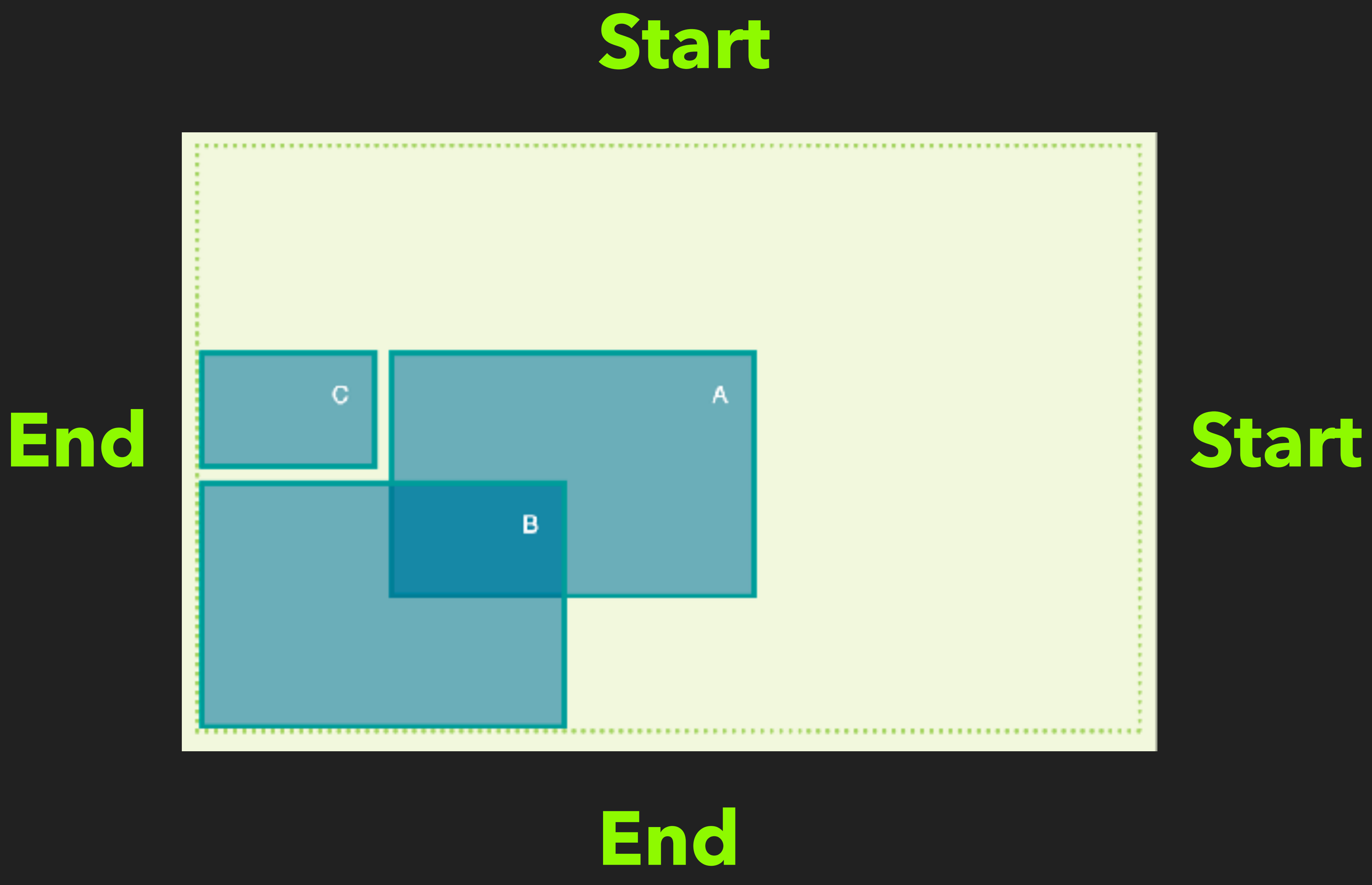
- The **start** of a page rather than the **top**
- The **end** of a block rather than the **right**
- In grid layout we have **start** and **end** for both **columns** and **rows**, rather than referring to the top and bottom of columns and left and right of rows

direction: ltr



```
.grid {
  display: grid;
  grid-gap: 10px;
  grid-template-columns: repeat(3,
150px);
  grid-template-rows: repeat(3, 100px);
  justify-content: end;
  align-content: end;
}
```

https://codepen.io/rachelandrew/pen/pwyBpG direction: rtl



```
.grid {
  display: grid;
  grid-gap: 10px;
  grid-template-columns: repeat(3, 
150px);
  grid-template-rows: repeat(3, 100px);
  justify-content: end;
  align-content: end;
}
```

https://codepen.io/rachelandrew/pen/pwyBpG

rachelandrew.co.uk ./CSS

Follow 23.2K followers

Box Alignment Cheatsheet

The box alignment specification details how items are aligned in the various layout methods. As different layout methods pose different constraints in terms of alignment, some of the behaviour of Box Alignment is layout method dependent. This cheatsheet compares alignment in CSS Grid Layout and Flexbox.

Key concepts of the Box Alignment specification

When you align an item, you do so within an **alignment container.** This is the rectangle you are aligning the item or items inside, usually the containing block. The item you are aligning inside that block is the **alignment subject**.

The specification defines three types of alignment:

- Positional alignment - keywords such as start, end, center
- Baseline alignment - baseline, last baseline, first baseline
- Distributed alignment - includes space-between and space-around

Most of these keyword values work in reference to the writing mode of the document. Therefore `start` may be the left-hand line of a grid container when working in English, but the right-hand line when working in a right-to-left language.

Block and Inline Axis

You will find the Block and Inline Axis referred to in various ways. The Block Axis, is referred to as the **Column Axis** in the Grid specification and in Flexbox as the **Cross Axis** as it runs across the **Main Axis**.

The Inline Axis is referred to as the **Row Axis** in the Grid specification and in Flexbox as the **Main Axis**.

Alignment on these Axes is easier to understand in Grid as we always have strict rows and columns, in flexbox we also

https://rachelandrew.co.uk/css/cheatsheets/box-alignment

What’s in a name?

What We Do Our Approach Our Work Articles Speaking Labs Team

Breaking Out With CSS Grid Layout

Written by Tyler Sticka on May 11, 2017

A while back I shared a simple technique for allowing certain elements to fill the full viewport width from within a fixed-width container:

```
.u-containProse {
  max-width: 40em;
  margin-left: auto;
  margin-right: auto;
}

.u-release {
  margin-left: calc(-50vw + 50%);
  margin-right: calc(-50vw + 50%);
}
```

https://cloudfour.com/thinks/breaking-out-with-css-grid-layout/

Just **do this!**

Magic occurs.

```
.Prose {
  display: grid;
  grid-template-columns:
    [full-start] minmax(1em, 1fr)
    [main-start] minmax(0, 40em) [main-end]
    minmax(1em, 1fr) [full-end];
}

.Prose > * {
  grid-column: main;
}

.Prose-splash {
  grid-column: full;
}
```

Balloons in Bristol

Despite the fact that British weather is not ideal for modes of transport that are unusable during any significant weather, Bristol is known for hot air balloons. These make pleasing images to include in presentation code demos. Here are some.

The content of this demo has been placed using a mixture of named grid laines and auto-placement. The named lines mark out the start and end of our content column and full width grid. We have not specified rows, and are using auto-placement which will place the items in a new row.

My markup

A div containing three direct child elements, one with a class of ‘gallery’. That’s our full width content.

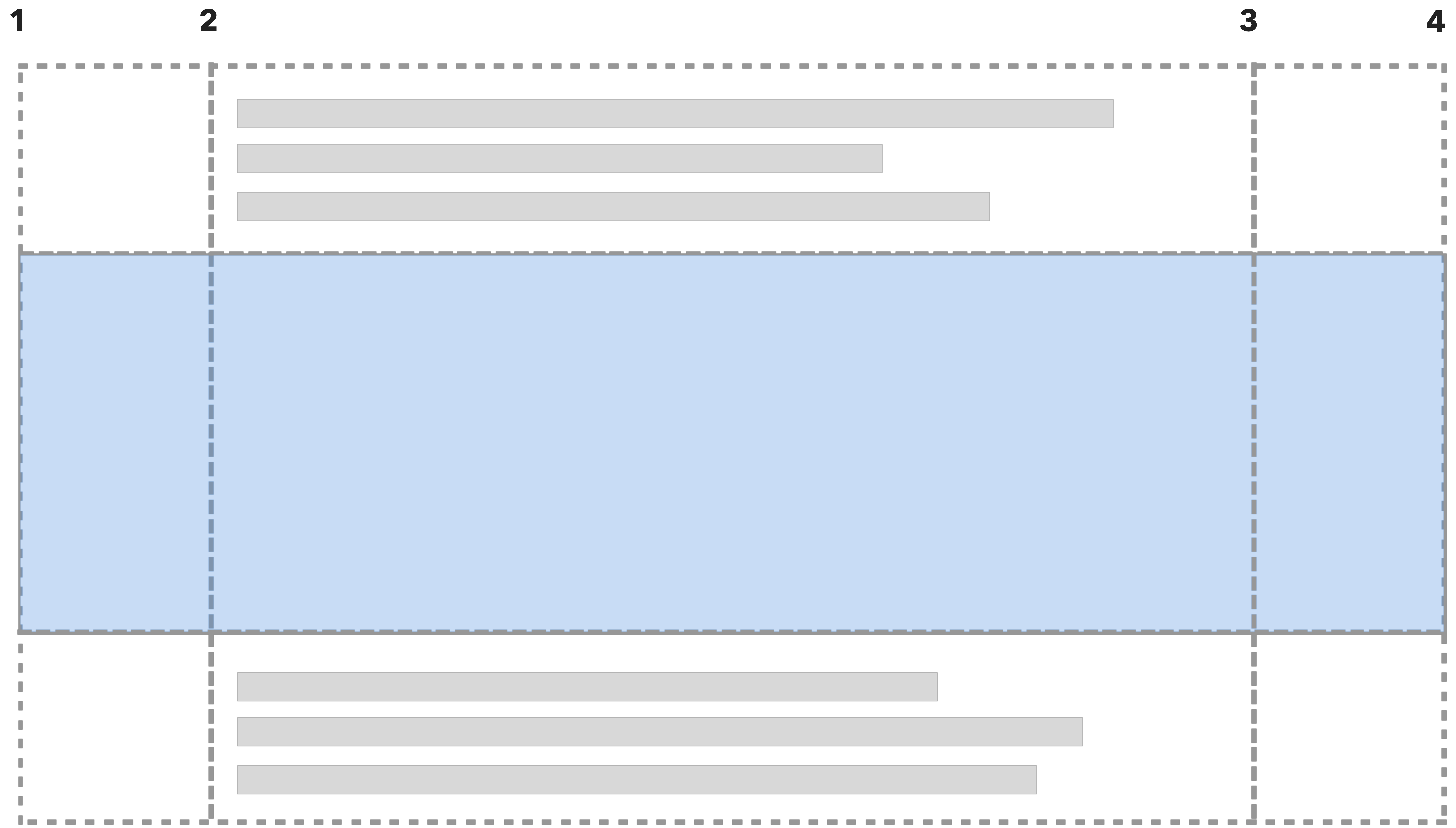
```
<div class="grid">
  <div>Content</div>
  <div class="gallery">Full width
content</div>
  <div>Content</div>
</div>
```

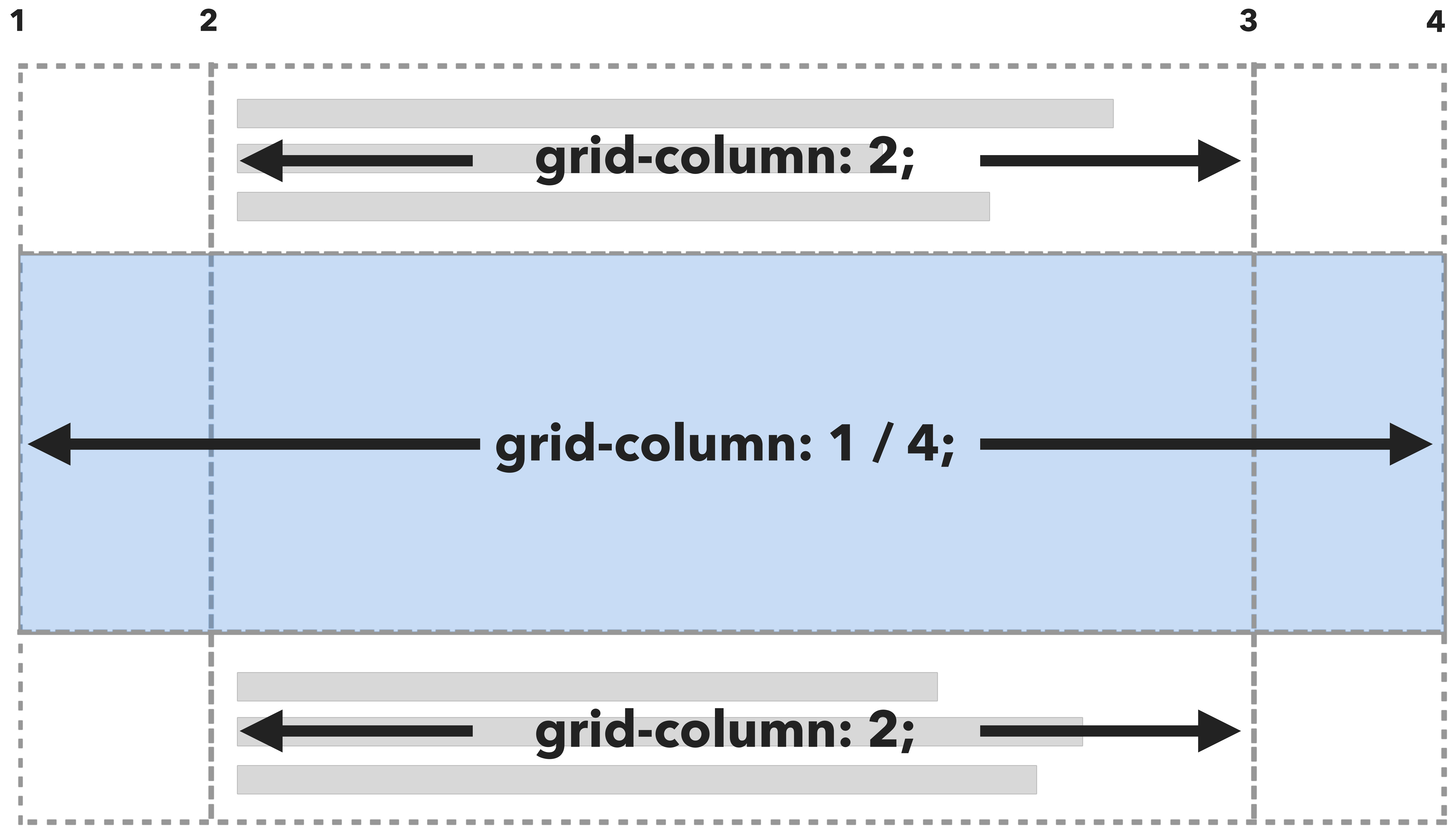
A grid with 3 column tracks

Using the line numbers to place our content and full width items.

```
.grid {
  display: grid;
  grid-template-columns:
    minmax(1em, 1fr)
    minmax(0, 660px)
    minmax(1em, 1fr);
}

.grid > * {
  grid-column: 2 ;
}

.grid > .gallery {
  grid-column: 1 / -1 ;
}
```

https://codepen.io/rachelandrew/pen/mwOmJW


1
2
3
4


1
2
3
4
grid-column: 2;
grid-column: 1 / 4;
grid-column: 2;

Balloons in Bristol

Despite the fact that British weather is not ideal for modes of transport that are unusable during any significant weather, Bristol is known for hot air balloons. These make pleasing images to include in presentation code demos. Here are some.

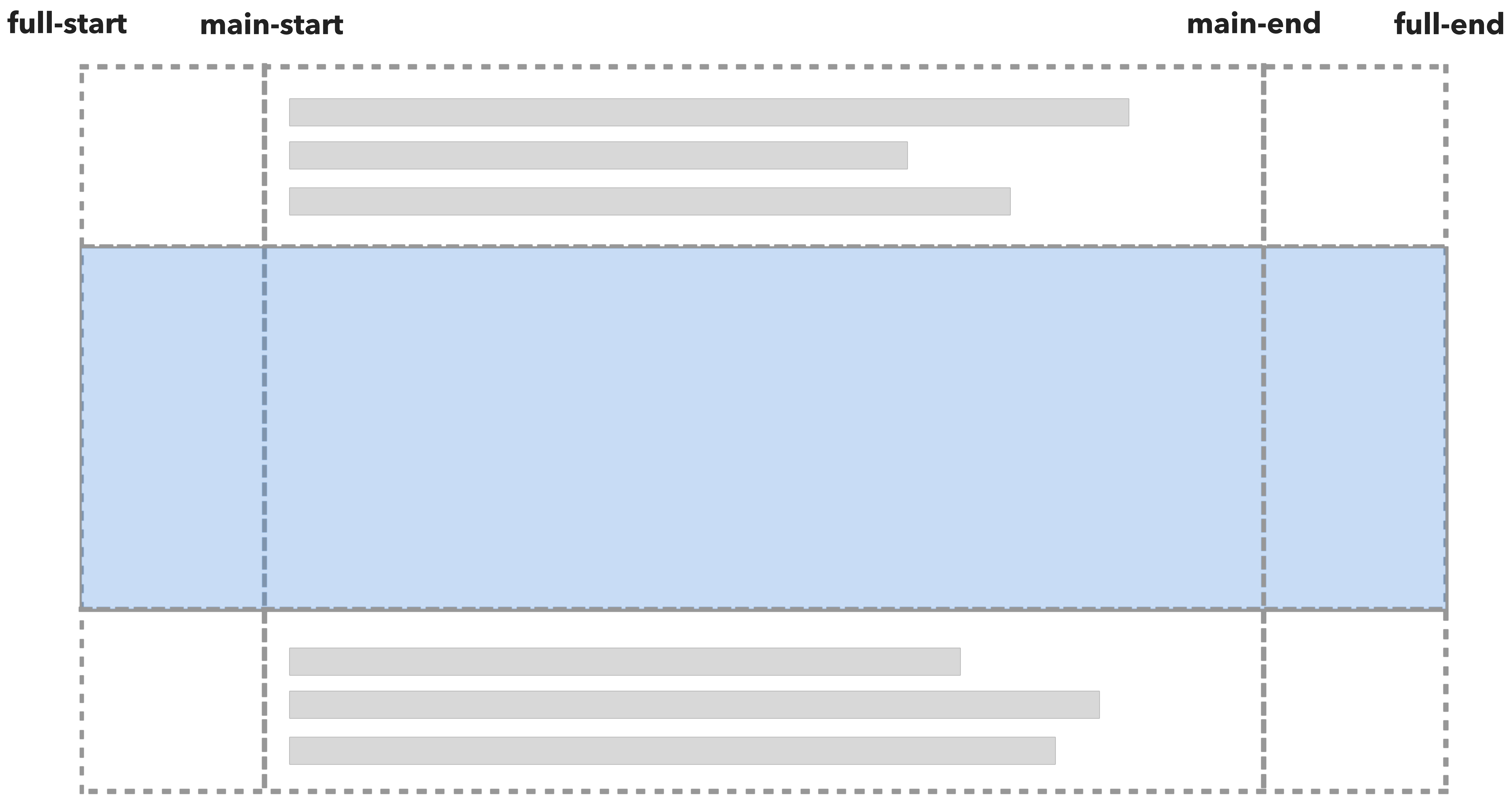
The content of this demo has been placed using a mixture of named grid laines and auto-placement. The named lines mark out the start and end of our content column and full width grid. We have not specified rows, and are using auto-placement which will place the items in a new row.

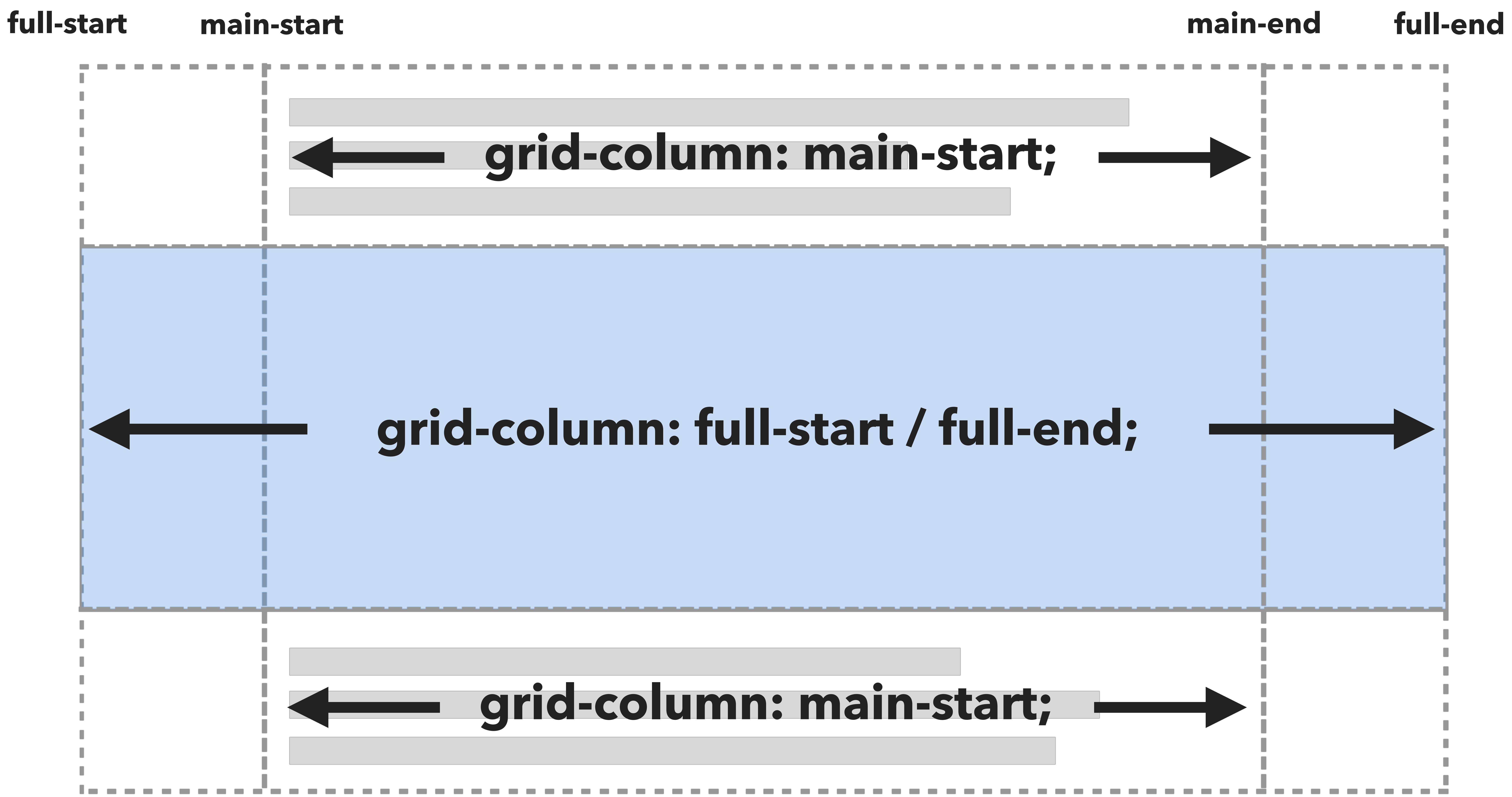
Naming lines on the grid

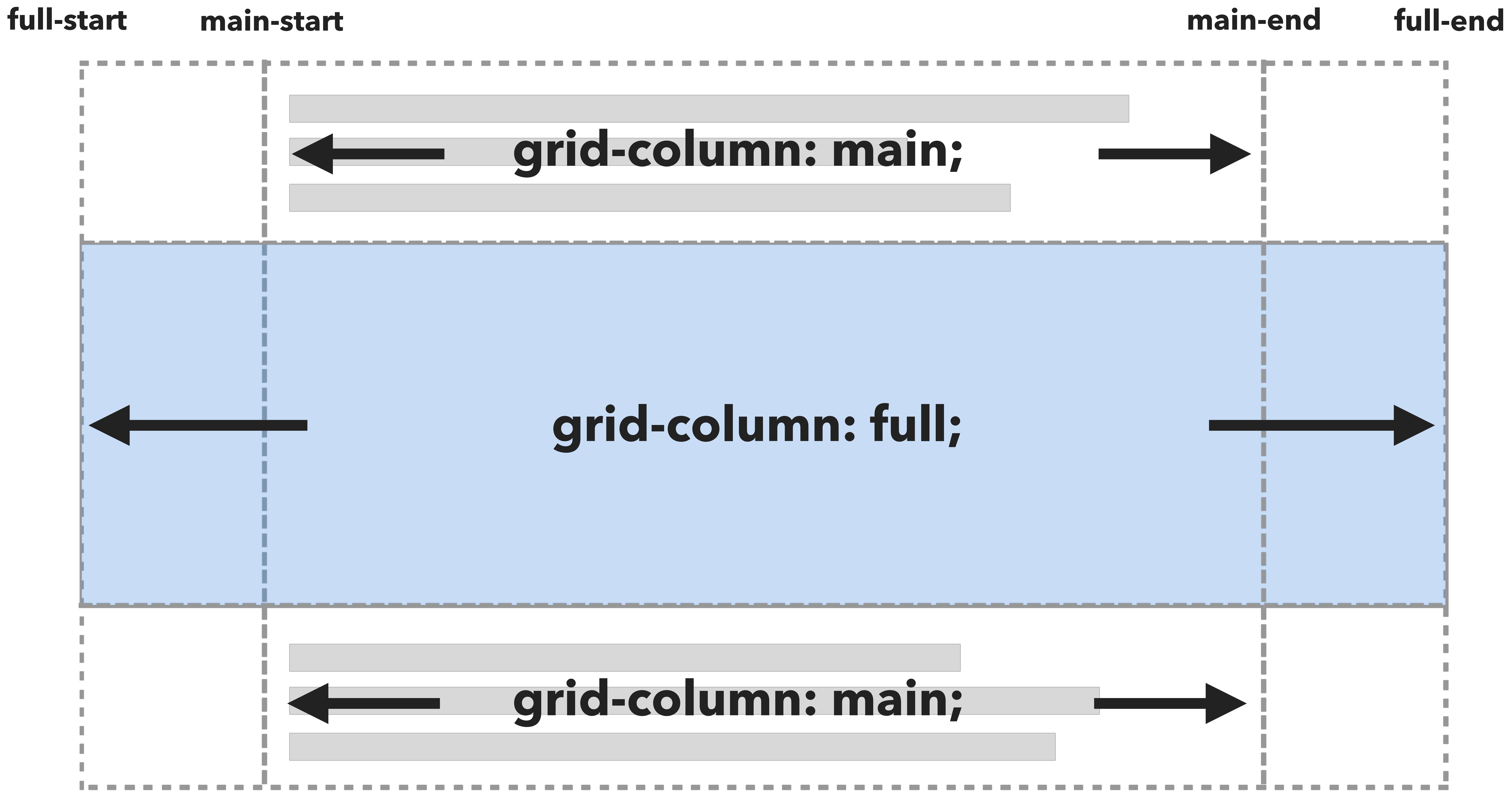
We can now position the items using their line *names*.

```
.grid {
  display: grid;
  grid-template-columns:
    [full-start] minmax(1em, 1fr)
    [main-start] minmax(0, 660px)
    [main-end] minmax(1em, 1fr)
    [full-end];
}

.grid > * {
  grid-column: main-start;
}

.grid > .gallery {
  grid-column: full-start / full-end;
}
```

https://codepen.io/rachelandrew/pen/EXjrJM


full-start
main-start
main-end
full-end


full-start
main-start
main-end
full-end
grid-column: main-start;
grid-column: full-start / full-end;
grid-column: main-start;


full-start
main-start
main-end
full-end
grid-column: main;
grid-column: full;
grid-column: main;

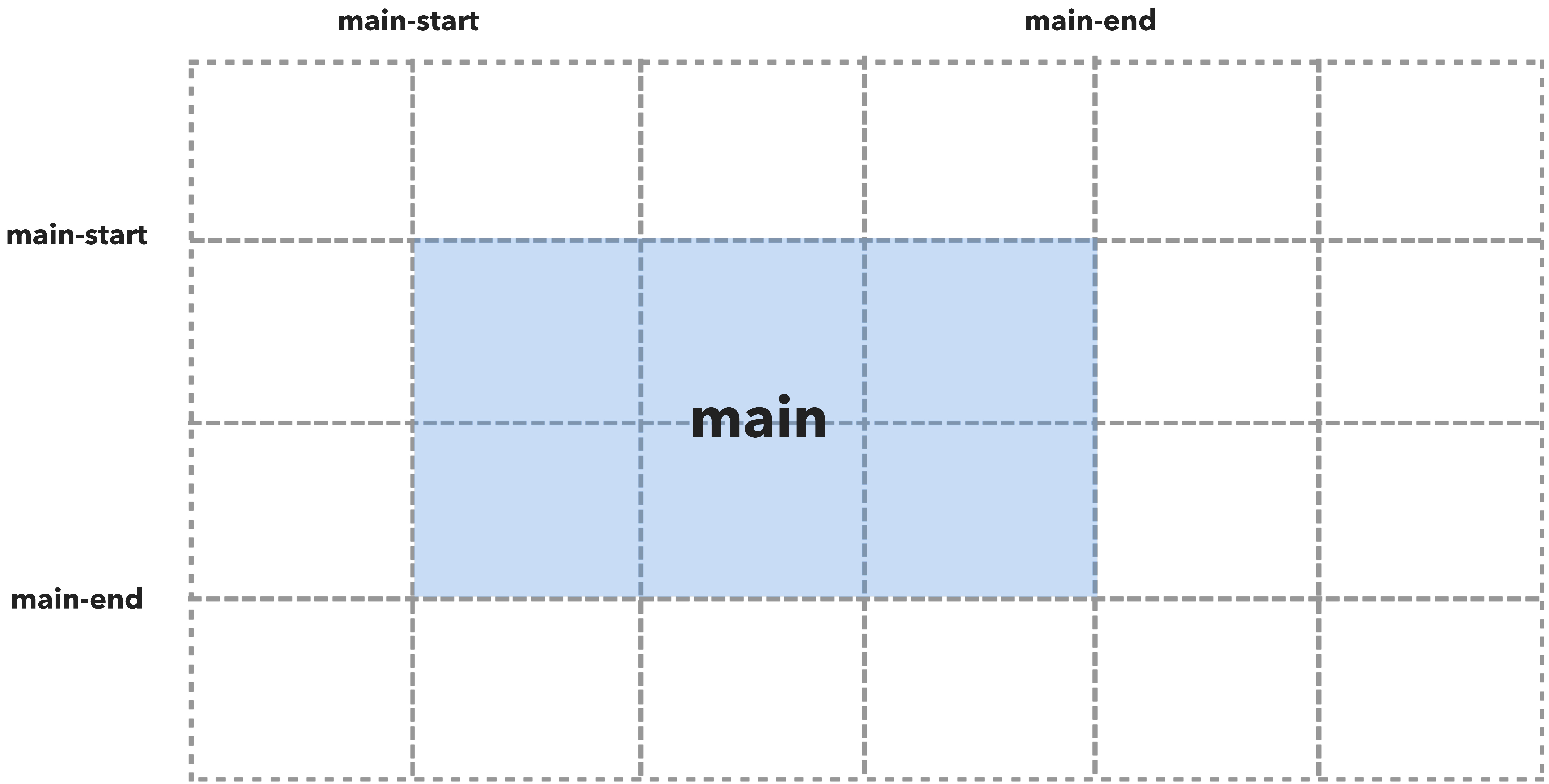
‘main’ and ‘full’

These line names don’t exist anywhere in our grid definition.

```
.grid {
  display: grid;
  max-width: 960px;
  margin: 0 auto;
  grid-template-columns: [full-start]
minmax(1em, 1fr)
    [main-start] minmax(0, 660px) [main-end]
    minmax(1em, 1fr) [full-end];
}

.grid > * {
  grid-column: main;
}

.grid > .gallery {
  grid-column: full;
}
```

https://www.w3.org/TR/css-grid-1/#implicit-named-areas

“Since a named grid area is referenced by the implicit named lines it produces, explicitly adding named lines of the same form (foo-start/foo-end) effectively creates a named grid area. ”


main-start
main-end
main-start
main
main-end

Implicit named areas

Created by having named lines using an ident with *-start and *-end.

```
.grid {
  display: grid;
  grid-gap: 20px;
  grid-template-columns:
    100px [main-start]
    100px 100px 100px [main-end]
    100px 100px;
  grid-template-rows:
    100px [main-start]
    100px 100px [main-end] 100px;
}

.item {
  grid-area: main;
}
```

https://codepen.io/rachelandrew/pen/EXNmvj

Magic named area

We have defined lines named **full-start** and **full-end** for rows and columns so we have an area named **full**.

```
.grid {
  display: grid;
  grid-template-columns: [full-start]
minmax(1em, 1fr)
    [main-start] minmax(0, 660px) [main-
end]
    minmax(1em, 1fr) [full-end];
  grid-template-rows: auto auto [full-
start] auto [full-end];
}

.grid > * {
  grid-column: main-start;
}

.grid > .gallery {
  grid-area: full;
}
```

https://codepen.io/rachelandrew/pen/jwPjWK

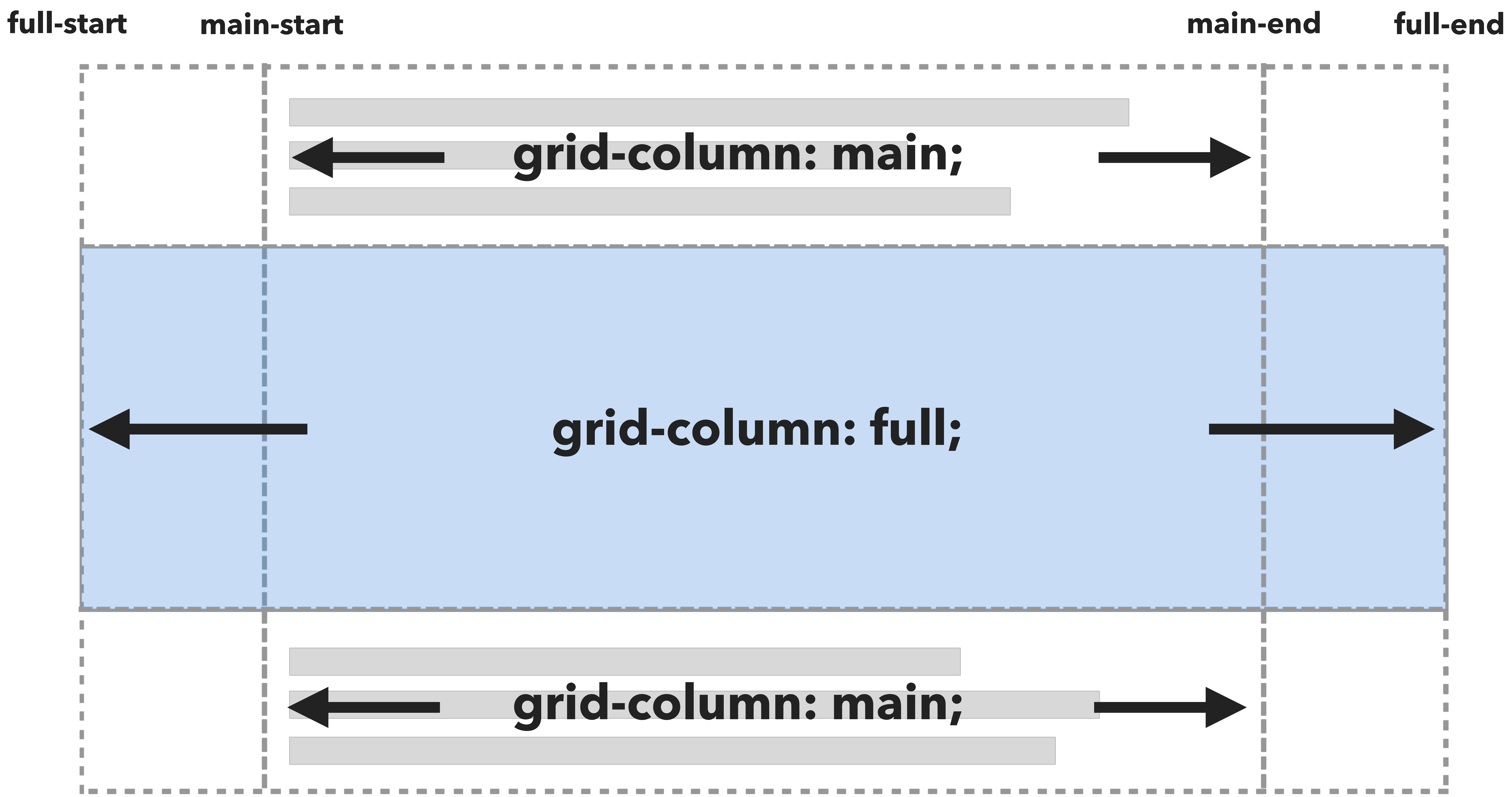
https://www.w3.org/TR/css-grid-1/#line-placement

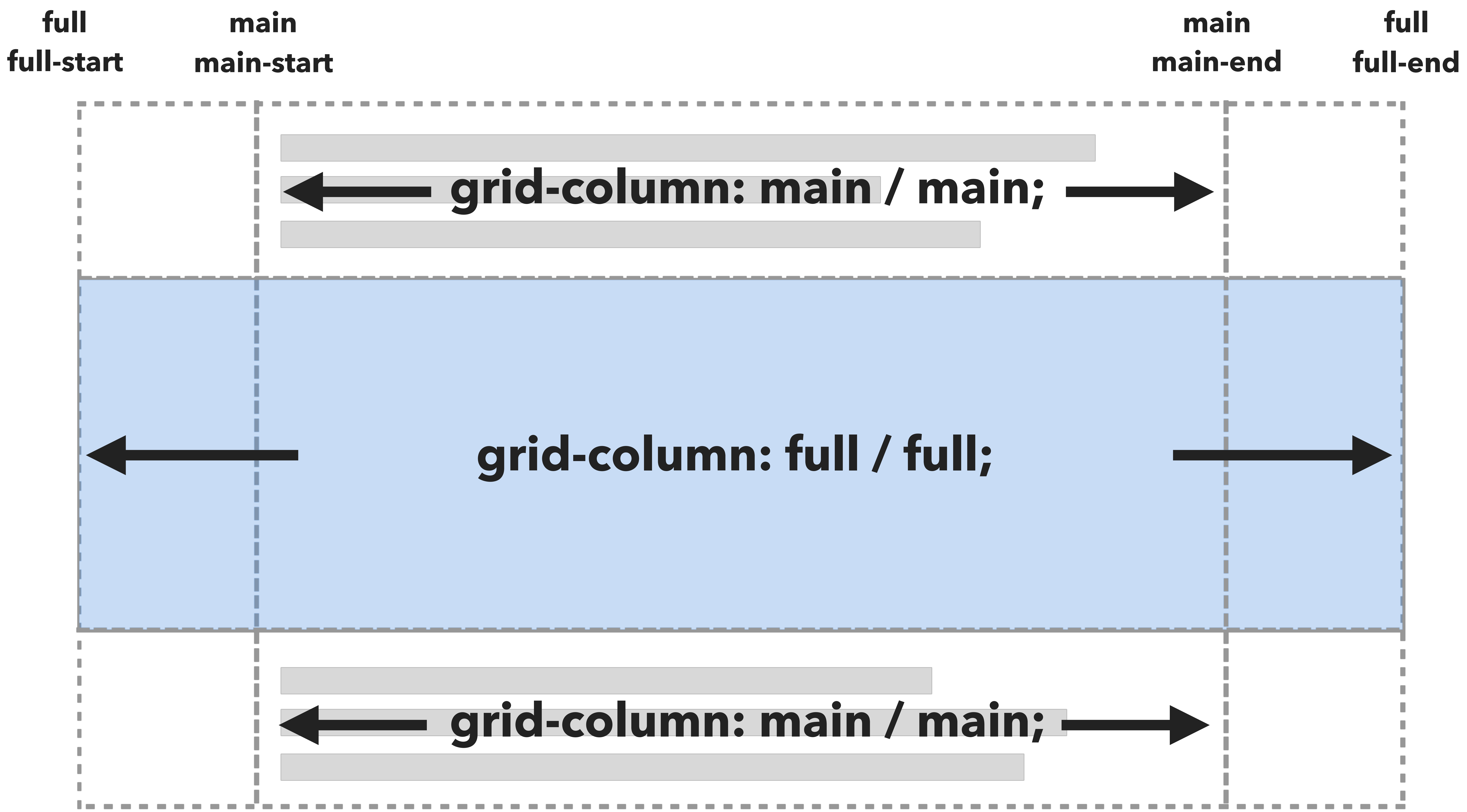
“Note: Named grid areas automatically generate implicit named lines of this form, so specifying grid-row-start: foo will choose the start edge of that named grid area (unless another line named foo-start was explicitly specified before it).”

Named lines create a named area which in turn can be used as named lines.

https://www.w3.org/TR/css-grid-1/#placement-shorthands

“[when using grid-row and grid-column shorthands] … When the second value is omitted, if the first value is a <custom-ident>, the grid-row-end/grid-column-end longhand is also set to that <custom-ident>; otherwise, it is set to auto.”


full-start
main-start
main-end
full-end
grid-column: main;
grid-column: full;
grid-column: main;


full
full-start
main
main-start
main
main-end
full
full-end
grid-column: main / main;
grid-column: full / full;
grid-column: main / main;

Targeting the column track

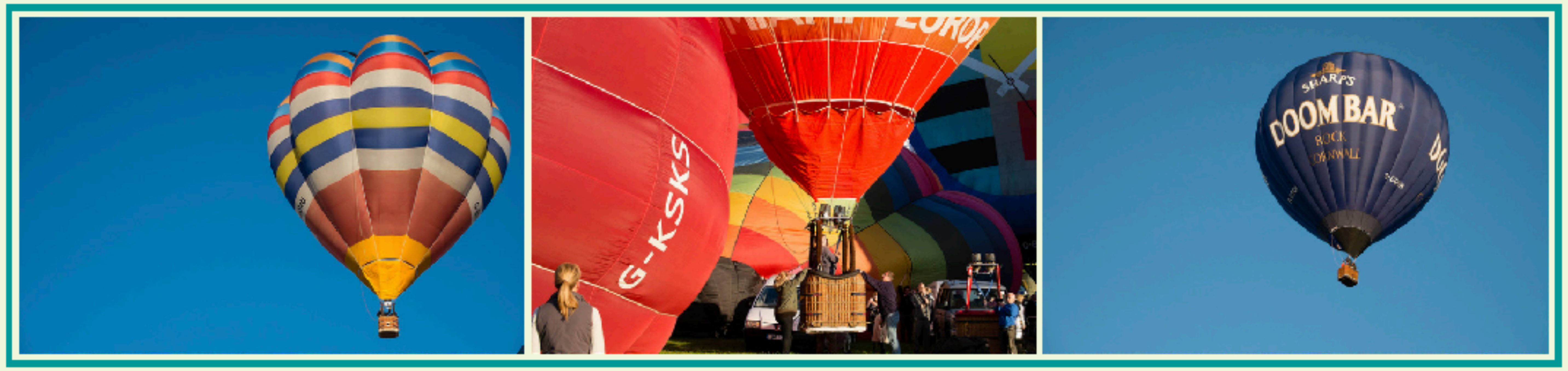
The line name ‘main’ is created from the named area created by our named lines.

```
.grid {
  display: grid;
  max-width: 960px;
  margin: 0 auto;
  grid-template-columns: [full-start]
minmax(1em, 1fr)
    [main-start] minmax(0, 660px) [main-end]
    minmax(1em, 1fr) [full-end];
}

.grid > * {
  grid-column: main;
}

.grid > .gallery {
  grid-column: full;
}
```

https://codepen.io/rachelandrew/pen/owXKMd

Balloons in Bristol

Despite the fact that British weather is not ideal for modes of transport that are unusable during any significant weather, Bristol is known for hot air balloons. These make pleasing images to include in presentation code demos. Here are some.



The content of this demo has been placed using a mixture of named grid laines and auto-placement. The named lines mark out the start and end of our content column and full width grid. We have not specified rows, and are using auto-placement which will place the items in a new row.

Extending the example

Adding named areas panel1 and panel2.

```
.grid {
  display: grid;
  grid-template-columns: [full-start panel1-start]
1fr 1fr [content-start] 1fr  1fr  1fr 1fr [panel1-
end panel2-start ] 1fr 1fr 1fr  1fr [content-end]
1fr 1fr [panel2-end full-end]  ;
}

.grid > * {
  grid-column: content;
}

.grid > .gallery {
  grid-column: full;
}

.grid > .panel1 {
  grid-column: panel1;
  padding: 4px;
}

.grid > .panel2 {
  grid-column: panel2;
  padding: 4px;
}
```

https://codepen.io/rachelandrew/pen/YQXmJJ/

Magic Grid Lines

If you have a named area you get grid lines named *-start and *-end for rows and columns.

```
.grid {
  display: grid;
  grid-template-columns: minmax(1em,
1fr) minmax(0, 660px) minmax(1em, 1fr);
  grid-template-areas:
    ". title ."
    ". content-top ."
    "full-width full-width full-width"
    ". content-bottom ."
}

h1 { grid-area: title; }

.content1 { grid-area: content-top; }

.content2 { grid-area: content-bottom; }

.gallery { grid-area: full-width; }
```

https://codepen.io/rachelandrew/pen/qjdzwR

Balloons in Bristol

Despite the fact that British weather is not ideal for modes of transport that are unusable during any significant weather, Bristol is known for hot air balloons. These make pleasing images to include in presentation code demos. Here are some.

The content of this demo has been placed using a mixture of named grid laines and auto-placement. The named lines mark out the start and end of our content column and full width grid. We have not specified rows, and are using auto-placement which will place the items in a new row.

Magic Grid Lines

Each grid-area creates a set of lines for the start and end of the area - rows and columns.

For **title**, we have **title-start** and **title-end** for rows and columns.

```
.grid {
  display: grid;
  grid-template-columns: minmax(1em,
1fr) minmax(0, 660px) minmax(1em, 1fr);
  grid-template-areas:
    ". title ."
    ". content-top ."
    "full-width full-width full-width"
    ". content-bottom ."
}

h1 { grid-area: title; }

.content1 { grid-area: content-top; }

.content2 { grid-area: content-bottom; }

.gallery { grid-area: full-width; }
```

https://codepen.io/rachelandrew/pen/qjdzwR

Magic Lines

Positioning some generated content using the magical lines.

```
.grid::after {
  content: "";
  background-color: #fff;
  border: 4px solid rgb(182,222,211);
  grid-column:
    content-top-start / content-top-end;
  grid-row:
    title-start / content-bottom-end;
  z-index: -1;
}
```

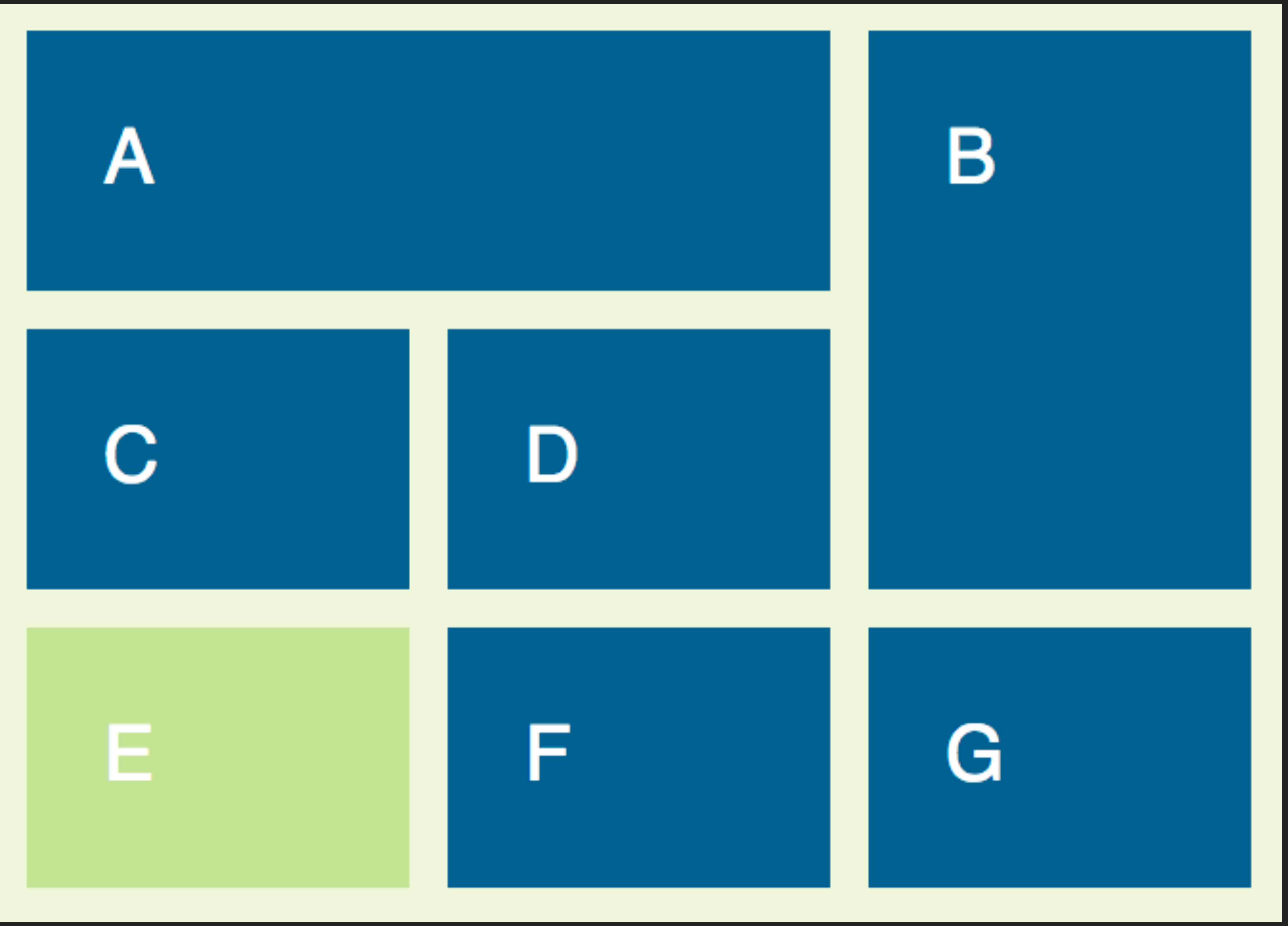
https://codepen.io/rachelandrew/pen/qjdzwR

Balloons in Bristol

Despite the fact that British weather is not ideal for modes of transport that are unusable during any significant weather, Bristol is known for hot air balloons. These make pleasing images to include in presentation code demos. Here are some.

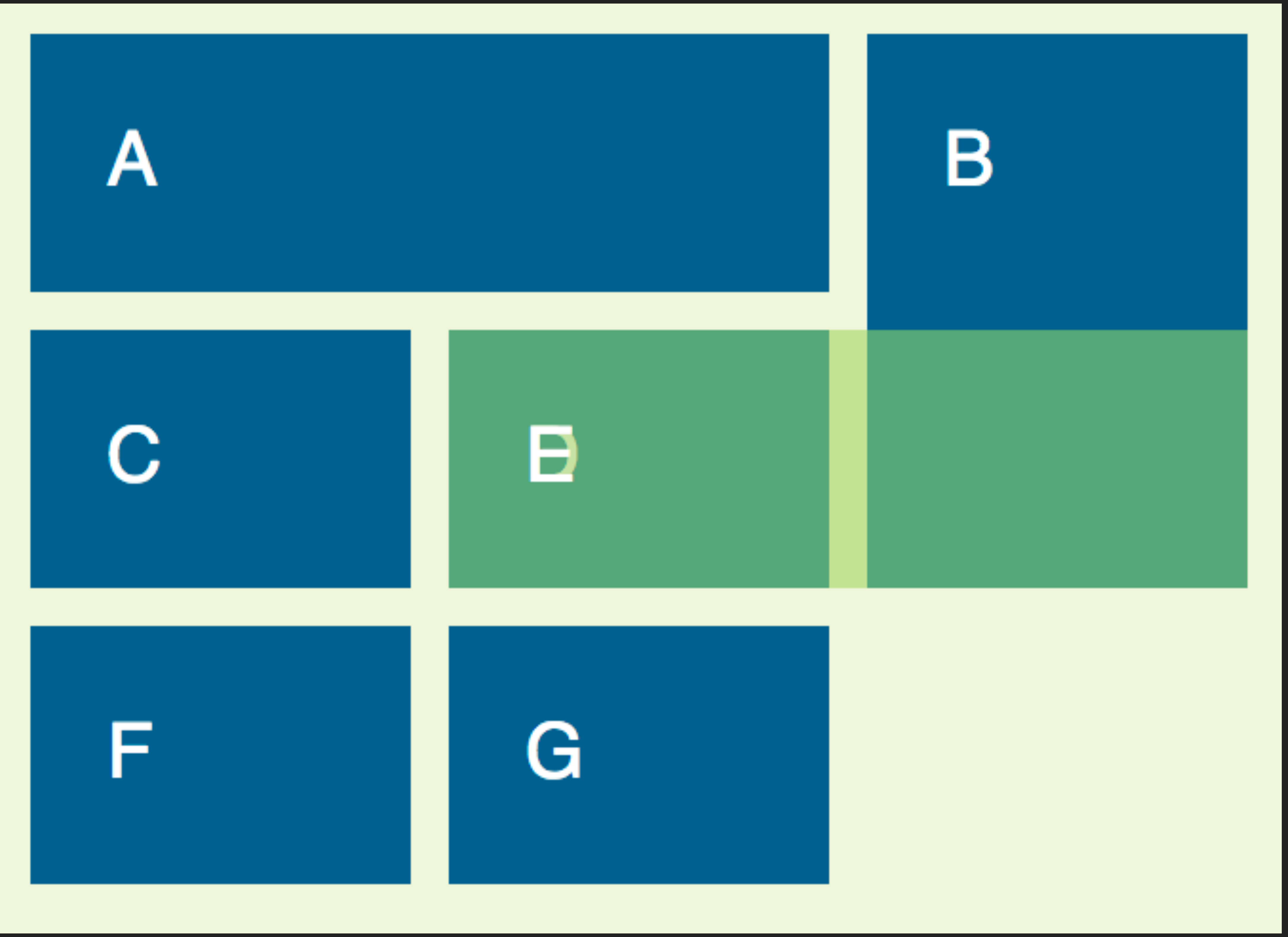
The content of this demo has been placed using a mixture of named grid laines and auto-placement. The named lines mark out the start and end of our content column and full width grid. We have not specified rows, and are using auto-placement which will place the items in a new row.

Things appearing in unexpected places.



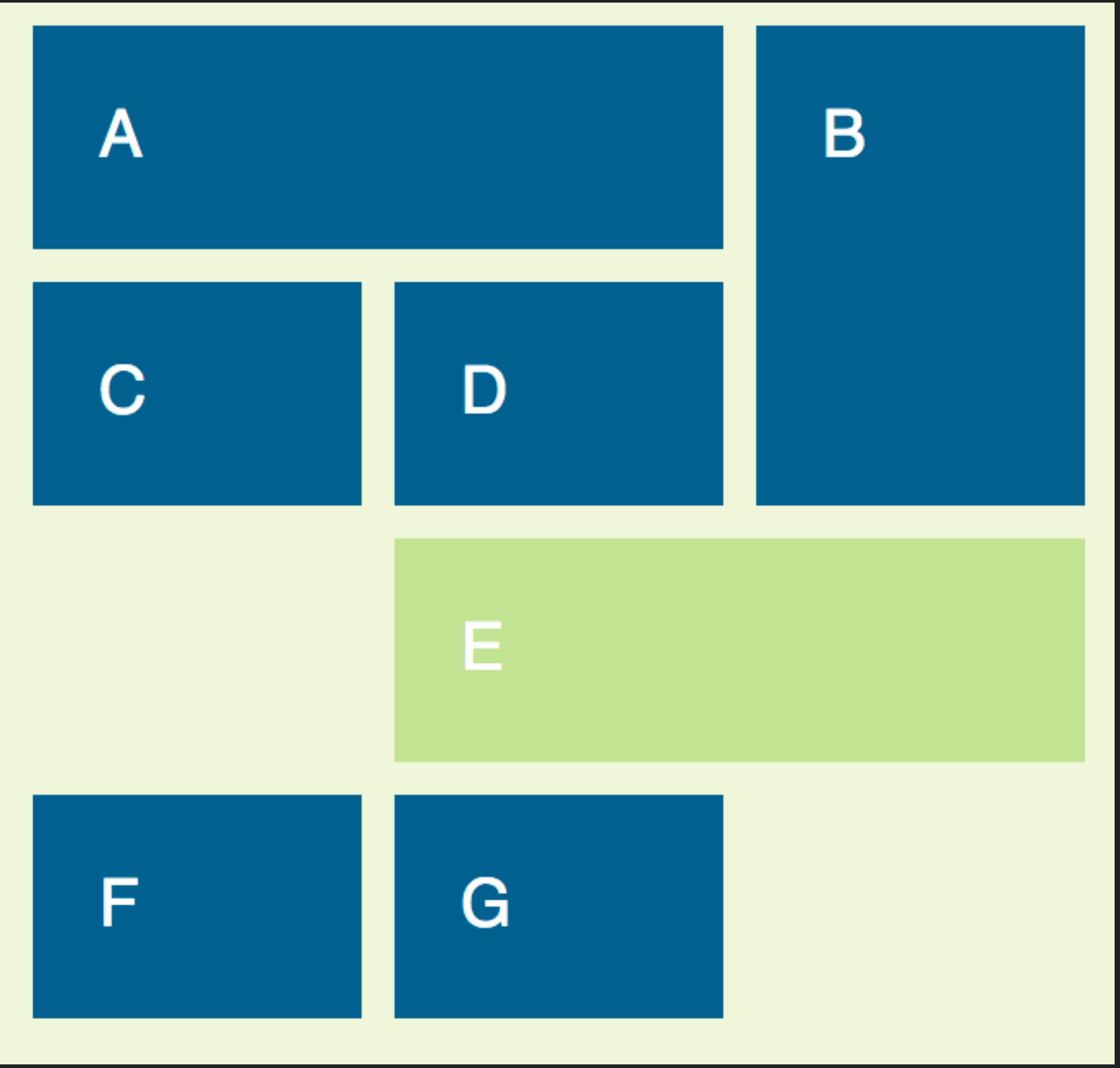
```
.grid {
  display: grid;
  grid-gap: 10px;
  grid-template-columns: 100px [main-start]
100px 100px [main-end];
}
.a {
  grid-column: 1 / 3;
  grid-row: 1;
}
.b {
  grid-column: 3;
  grid-row: 1 / 3;
}
.c {
  grid-column: 1;
}
.d {
  grid-column: 2;
  grid-row: 2;
}
```

https://codepen.io/rachelandrew/pen/JJYxve/



```
.grid {
  display: grid;
  grid-gap: 10px;
  grid-template-columns: 100px [main-
start] 100px 100px [main-end];
}

.e {
  grid-area: main;
}
```

https://codepen.io/rachelandrew/pen/JJYxve/



```
.grid {
  display: grid;
  grid-gap: 10px;
  grid-template-columns: 100px [main-
start] 100px 100px [main-end];
}

.e {
  grid-area: auto/ main;
}
```

https://codepen.io/rachelandrew/pen/JJYxve/

So many possibilities.

I made you some resources

Visit Grid by Example for worked examples, patterns with fallbacks, and a free video tutorial:
gridbyexample.com

I created a huge set of guides for MDN:
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout

Over 4 years of grid thoughts on my site at:
https://rachelandrew.co.uk/archives/tag/cssgrid

CSS Grid AMA:
https://github.com/rachelandrew/cssgrid-ama

@rachelandrew

Resources & code: https://rachelandrew.co.uk/speaking/event/cssday-nl-2017

THANK YOU!