



Super-powered Layouts

with

CSS Grid +
CSS Variables

CSS Grid

(CSS Grid Layout Module Level 1)



Why not just use
flexbox?



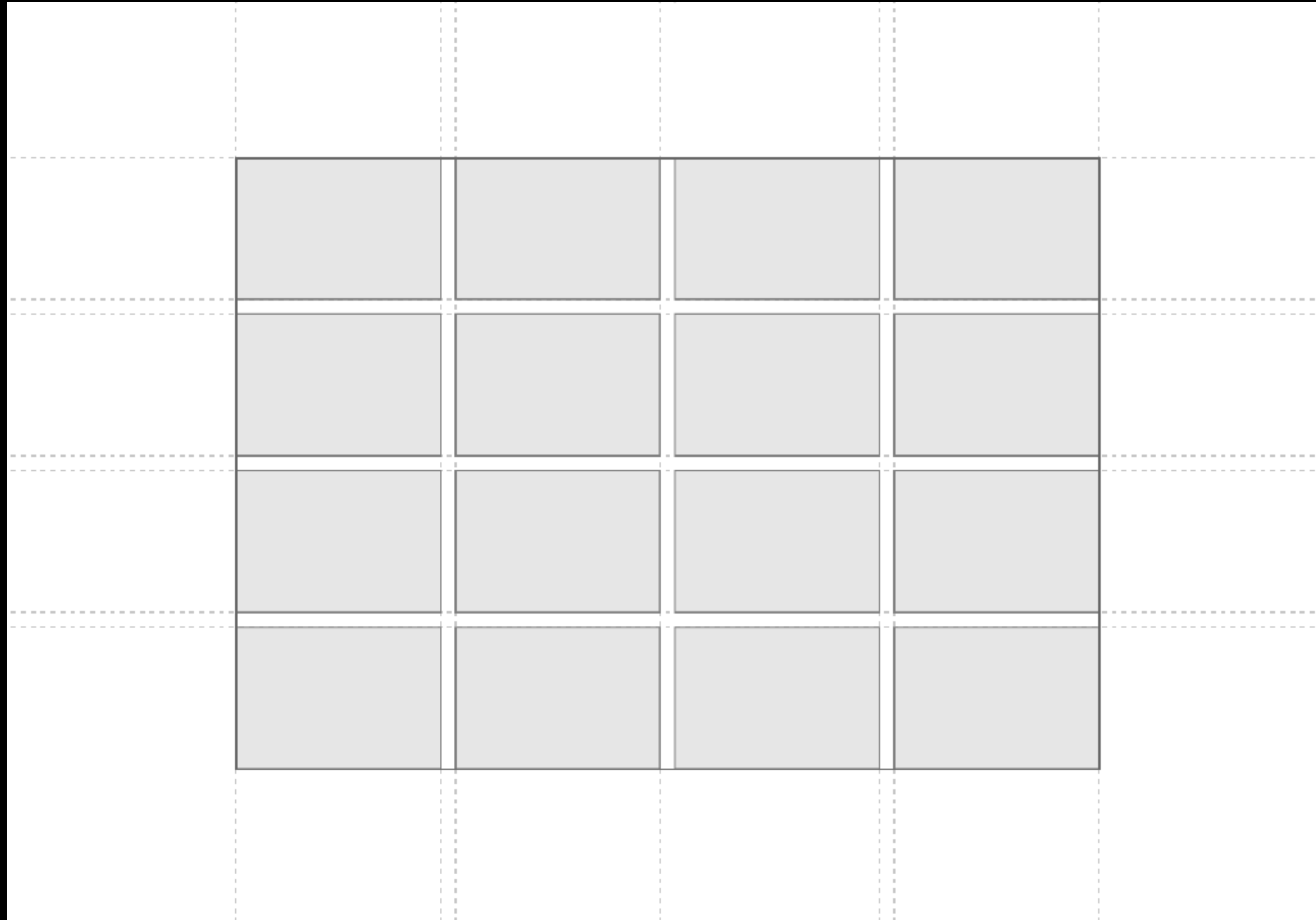
Intrinsic Web Design



Grid terminology

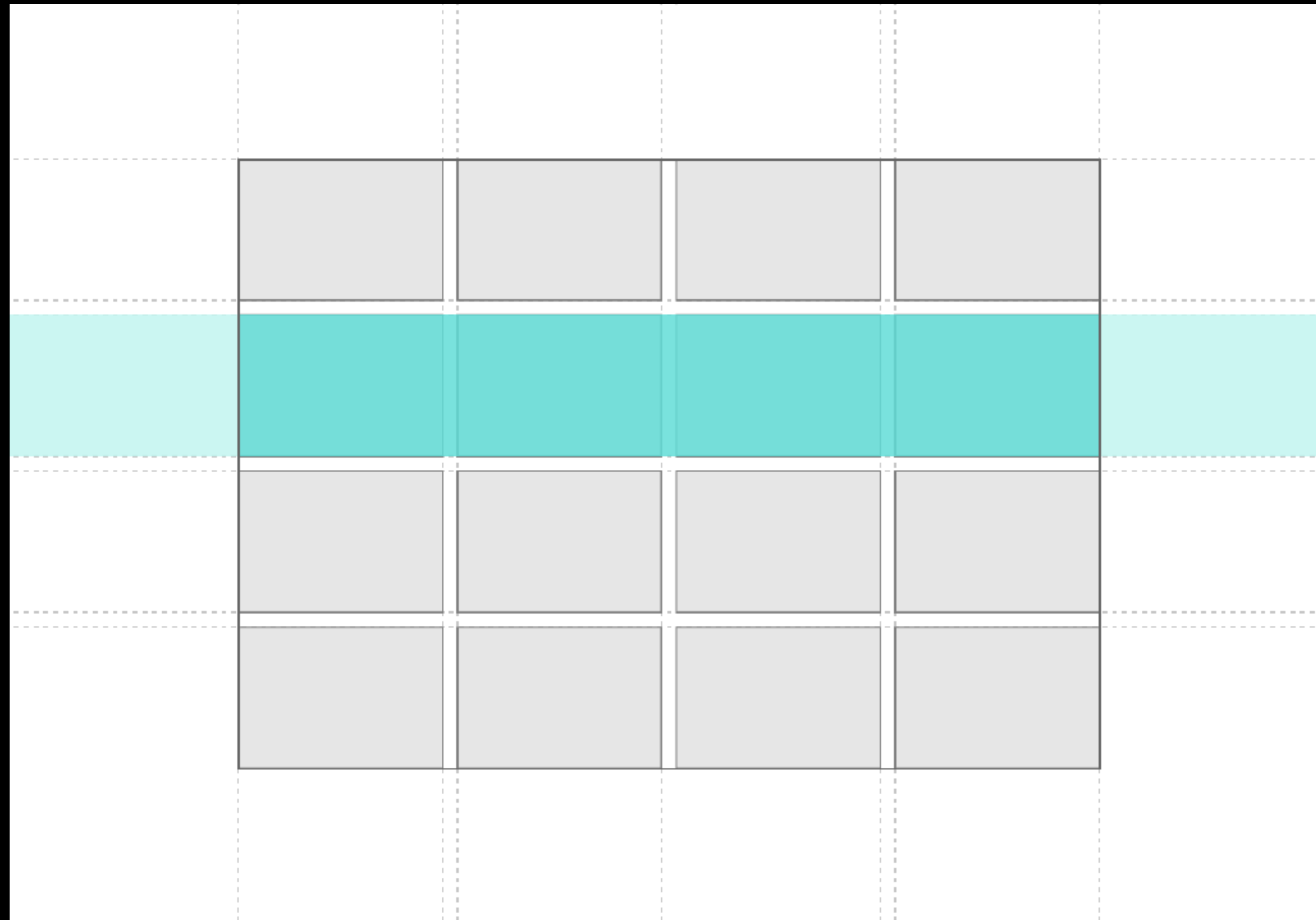
Grid

```
display: grid;
```



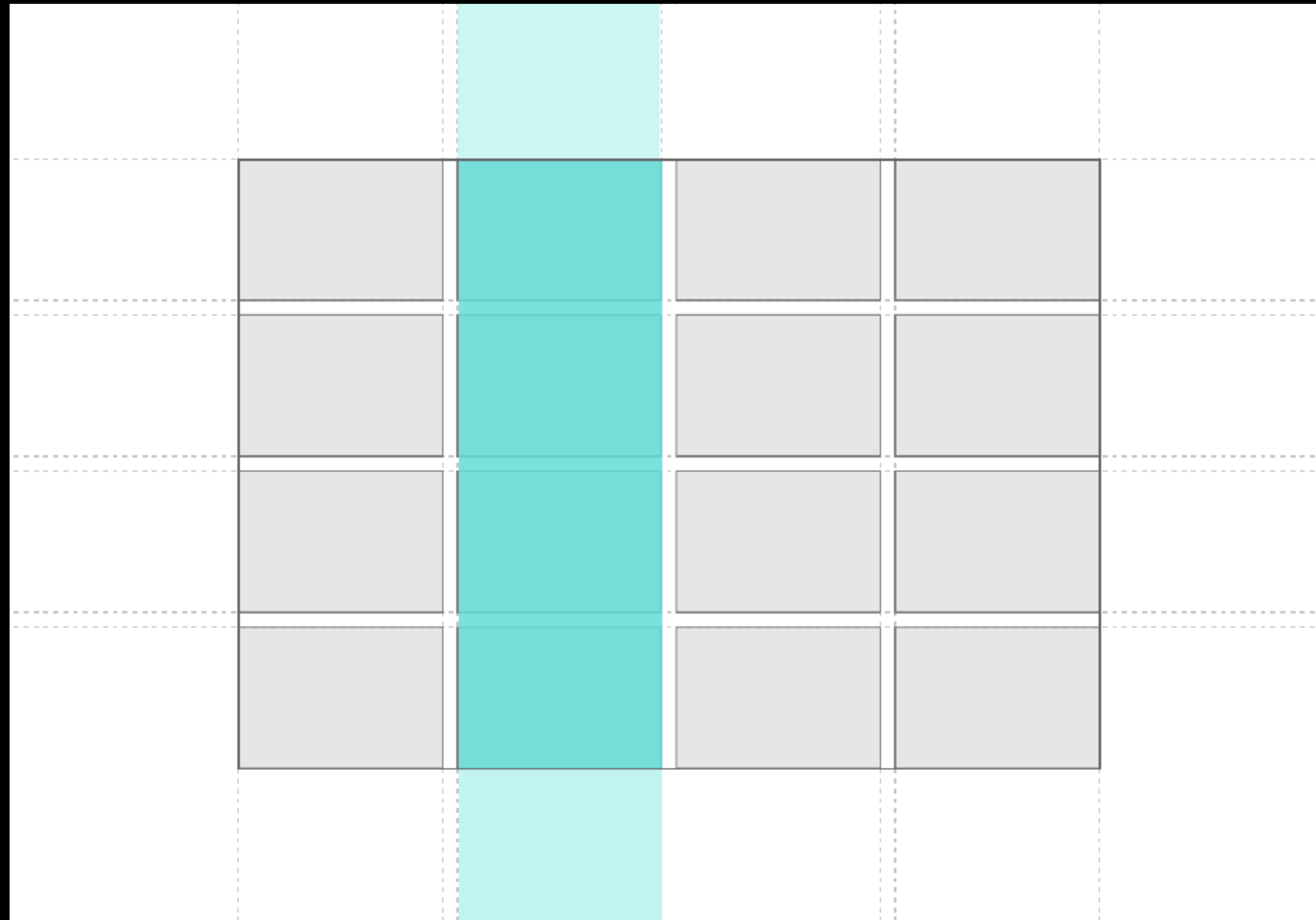
Tracks

grid-template-rows

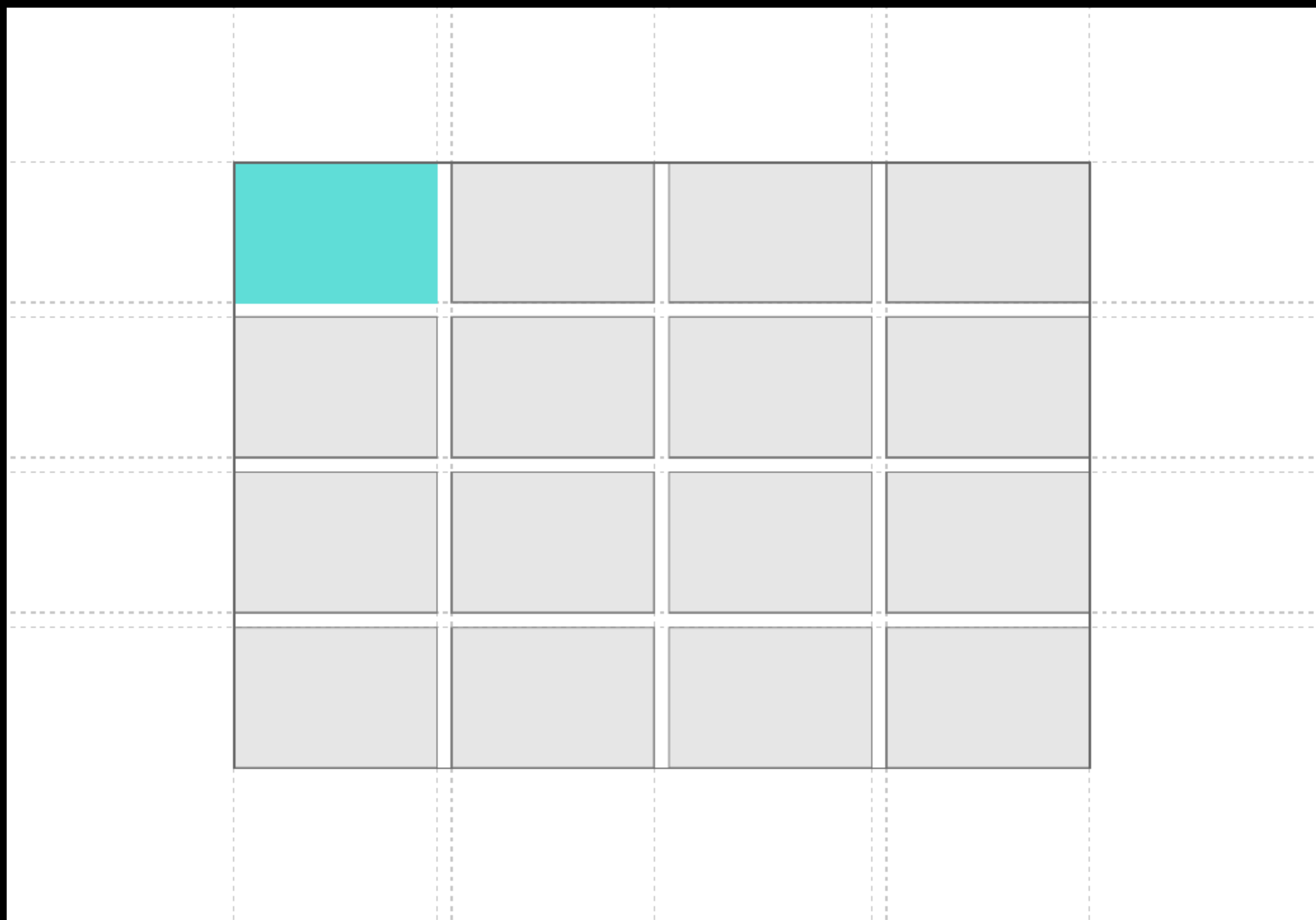


Tracks

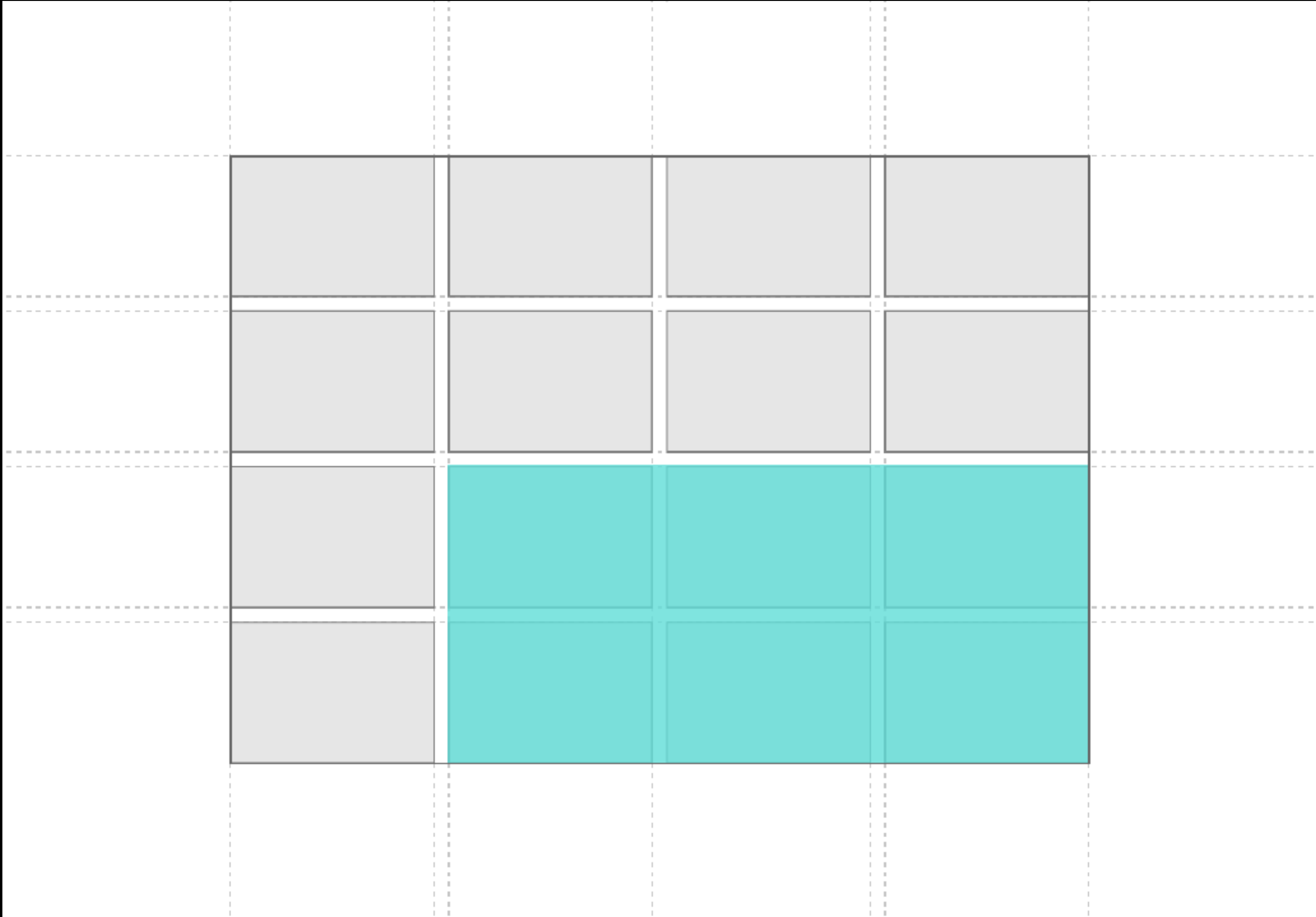
grid-template-columns



Cells

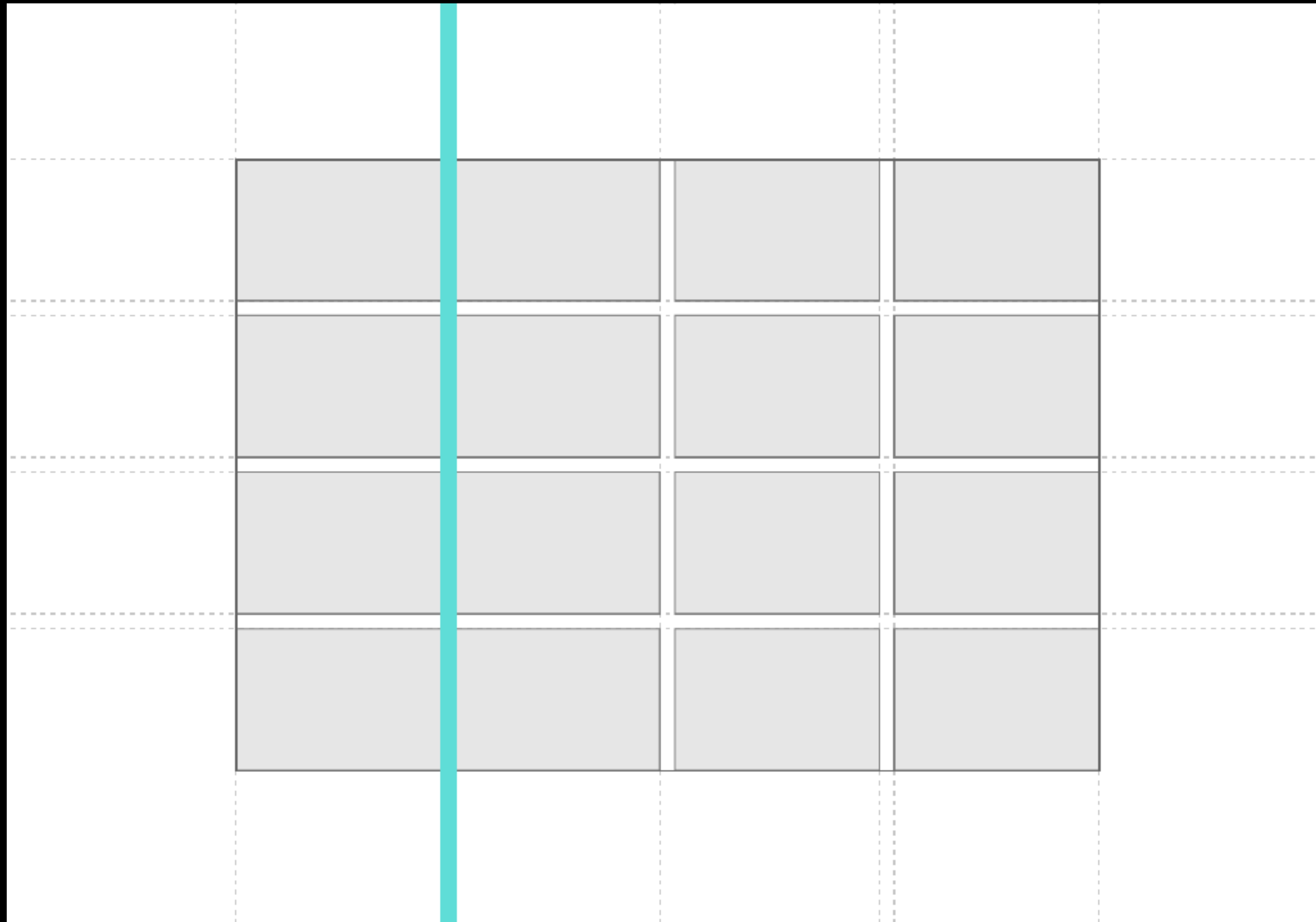


Grid areas



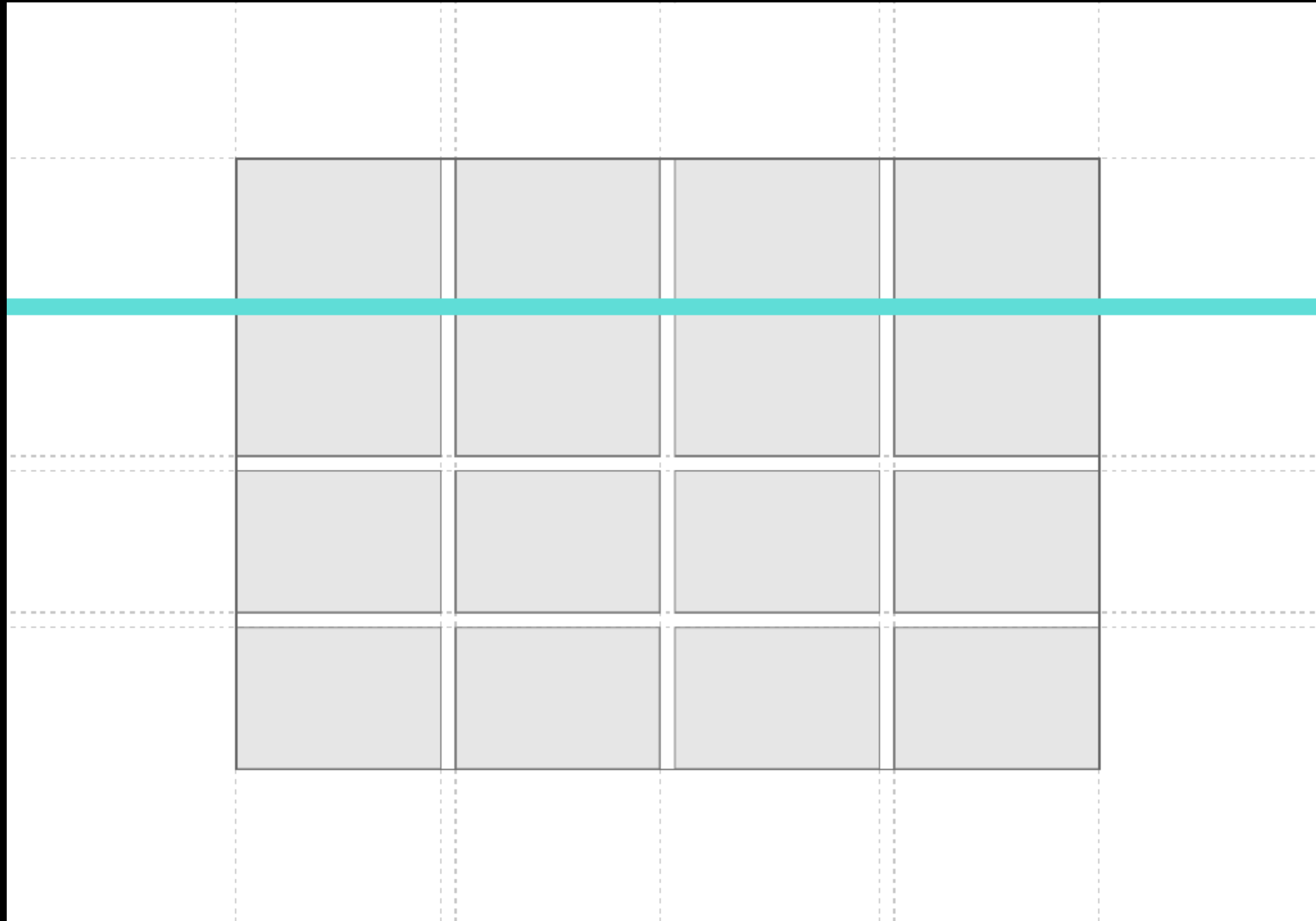
Gutters

`grid-column-gap`



Gutters

`grid-row-gap`





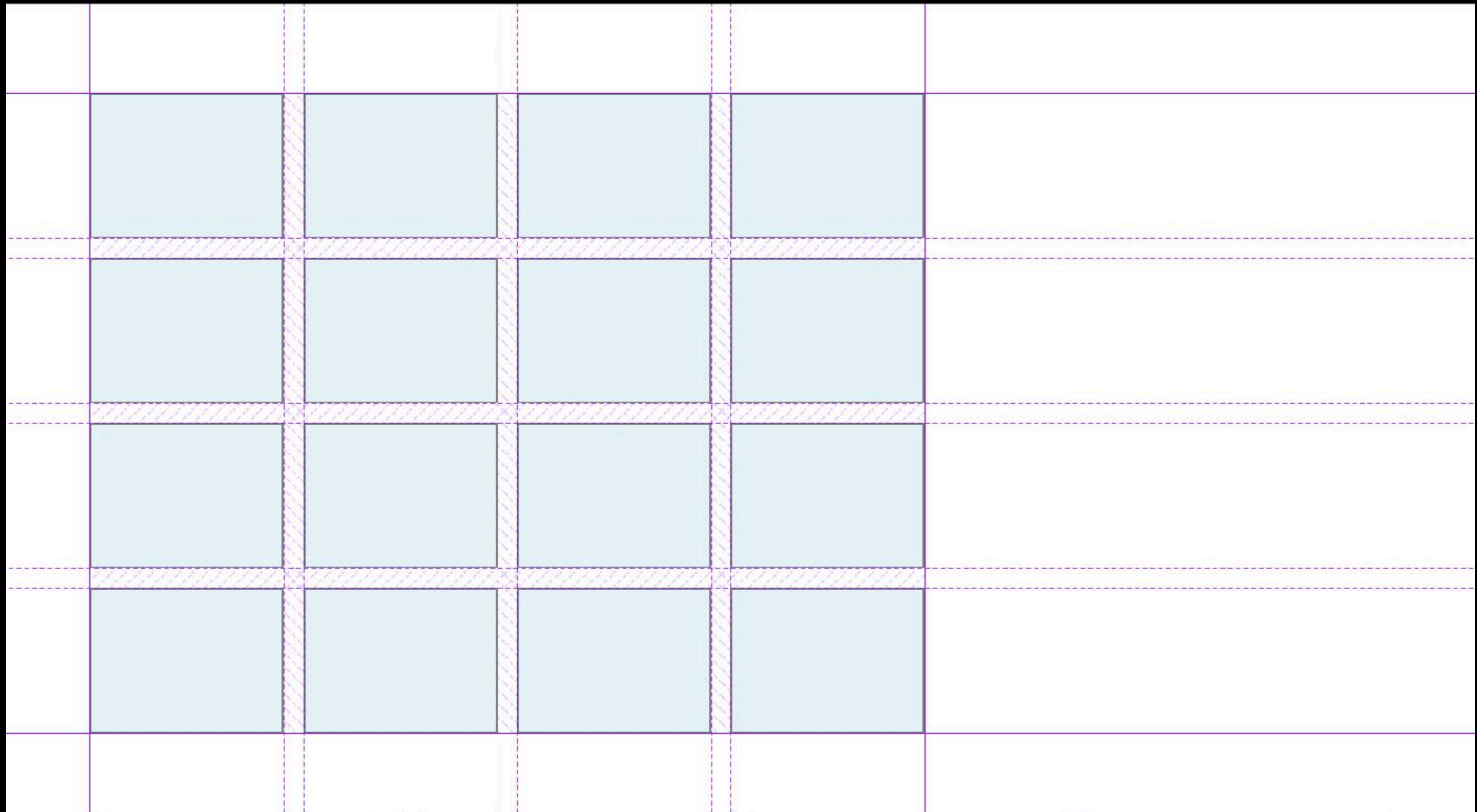
Defining a grid

On the container:

```
.grid {  
  display: grid;  
  grid-template-columns: 200px 200px 200px 200px;  
  grid-template-rows: 150px 150px 150px 150px;  
}
```


On the container:

```
.grid {  
  display: grid;  
  grid-template-columns: 200px 200px 200px 200px;  
  grid-template-rows: 150px 150px 150px 150px;  
  grid-column-gap: 20px;  
  grid-row-gap: 20px;  
}
```



codepen.io/michellebarker/pen/eLZwVg

repeat()

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(4, 200px);  
  grid-template-rows: repeat(4, 150px);  
  grid-gap: 20px;  
}
```

Shorthand

```
.grid {  
  display: grid;  
  grid-template: repeat(4, 150px) / repeat(4, 200px);  
  grid-gap: 20px;  
}
```

Shorthand

```
.grid {  
  display: grid;  
  grid-template: repeat(4, 150px) / repeat(4, 200px);  
  grid-gap: 20px; // grid-row-gap / grid-column-gap  
}
```


Flexible units (fr)

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  grid-template-rows: 150px;  
  grid-gap: 20px;  
}
```

“...represents a fraction of the leftover space in the grid container.

Tracks sized with fr units...respond to leftover space similar to how flex items fill space in a flex container.”

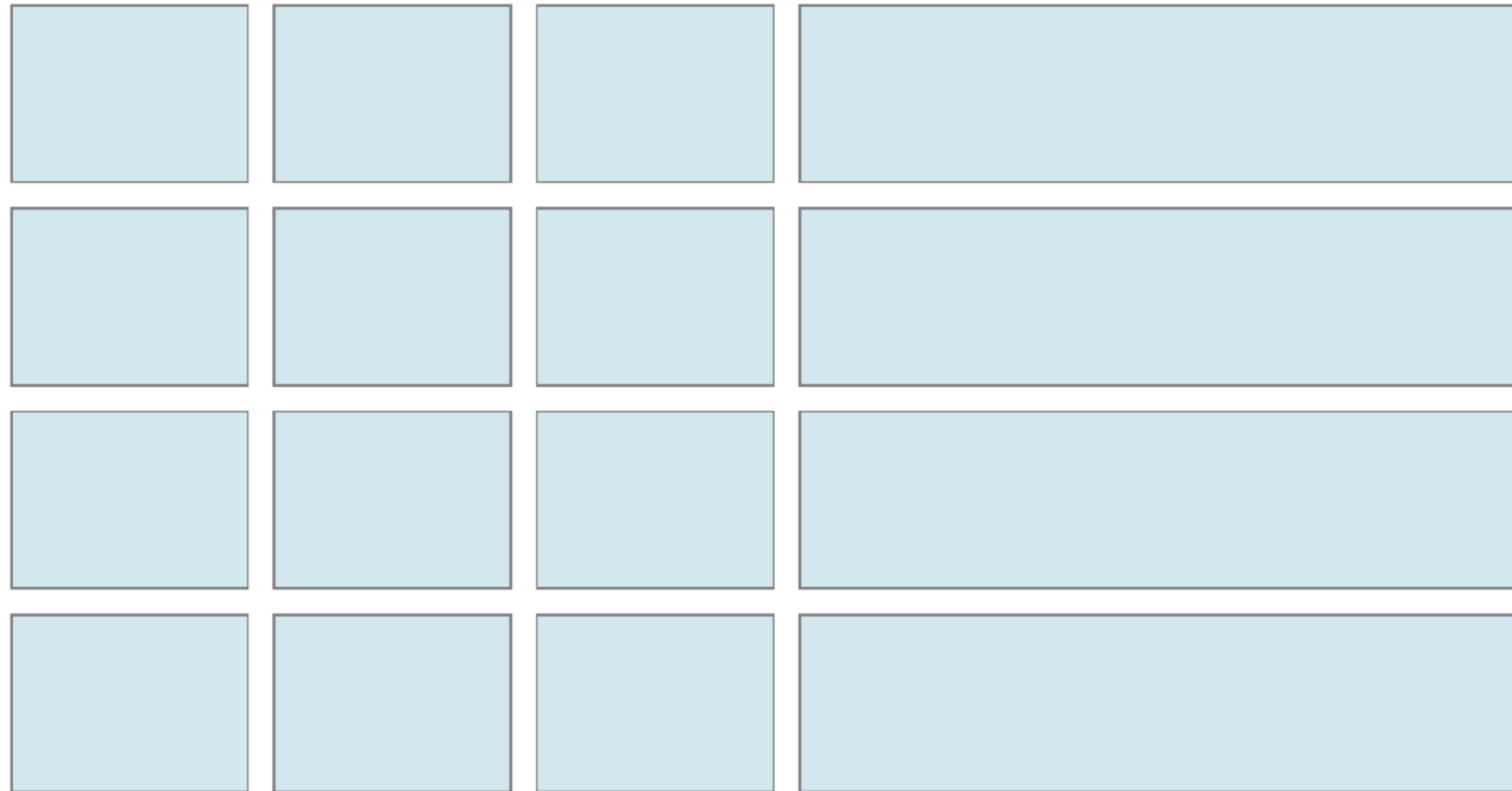
Flexible units (fr)

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  grid-template-rows: repeat(4, 150px);  
  grid-gap: 20px;  
}
```


Flexible units (fr)

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(3, 200px) 1fr;  
  grid-template-rows: 150px;  
  grid-gap: 20px;  
}
```

Flexible units (fr)

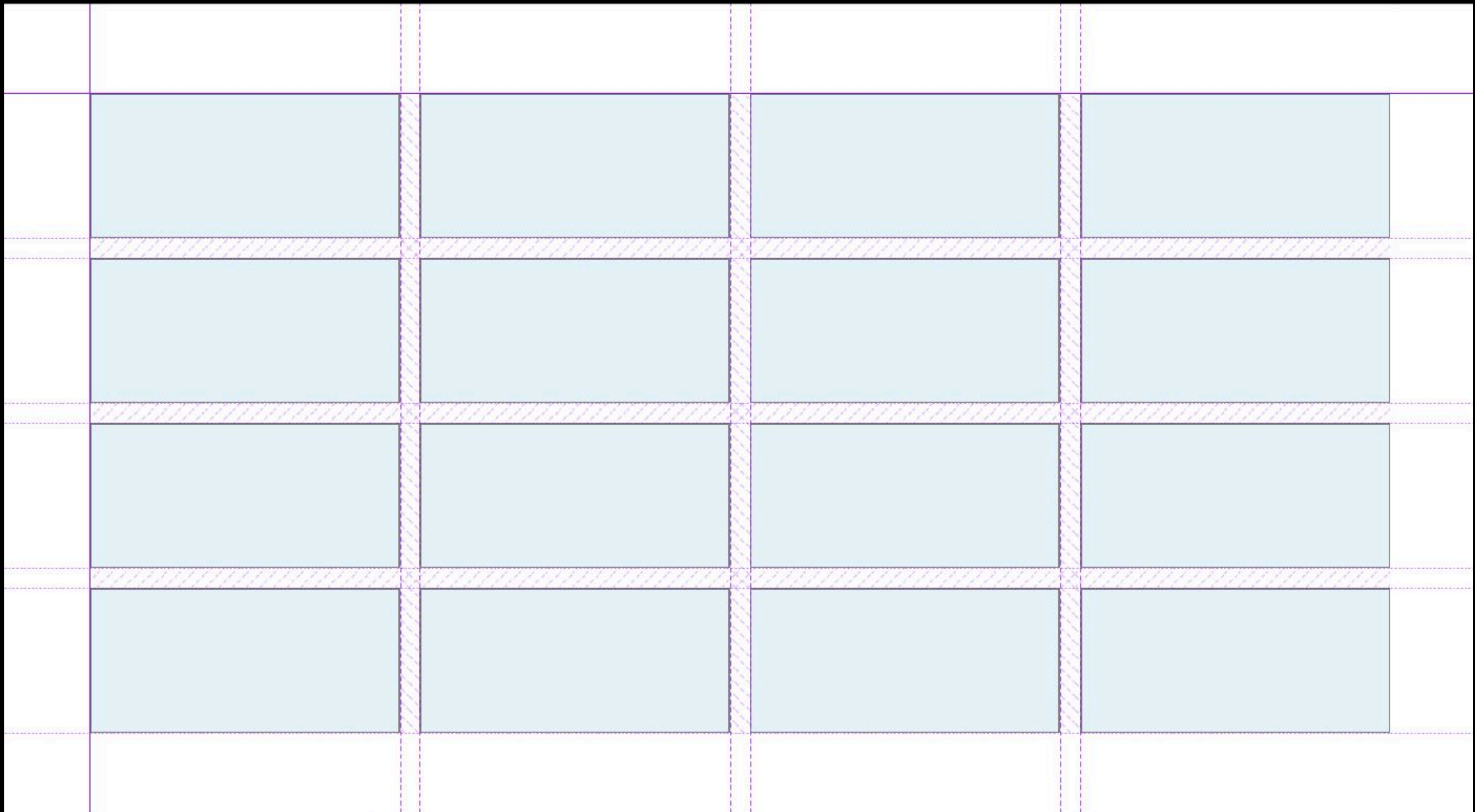


Implicit tracks

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  grid-auto-rows: 150px;  
  grid-gap: 20px;  
}
```

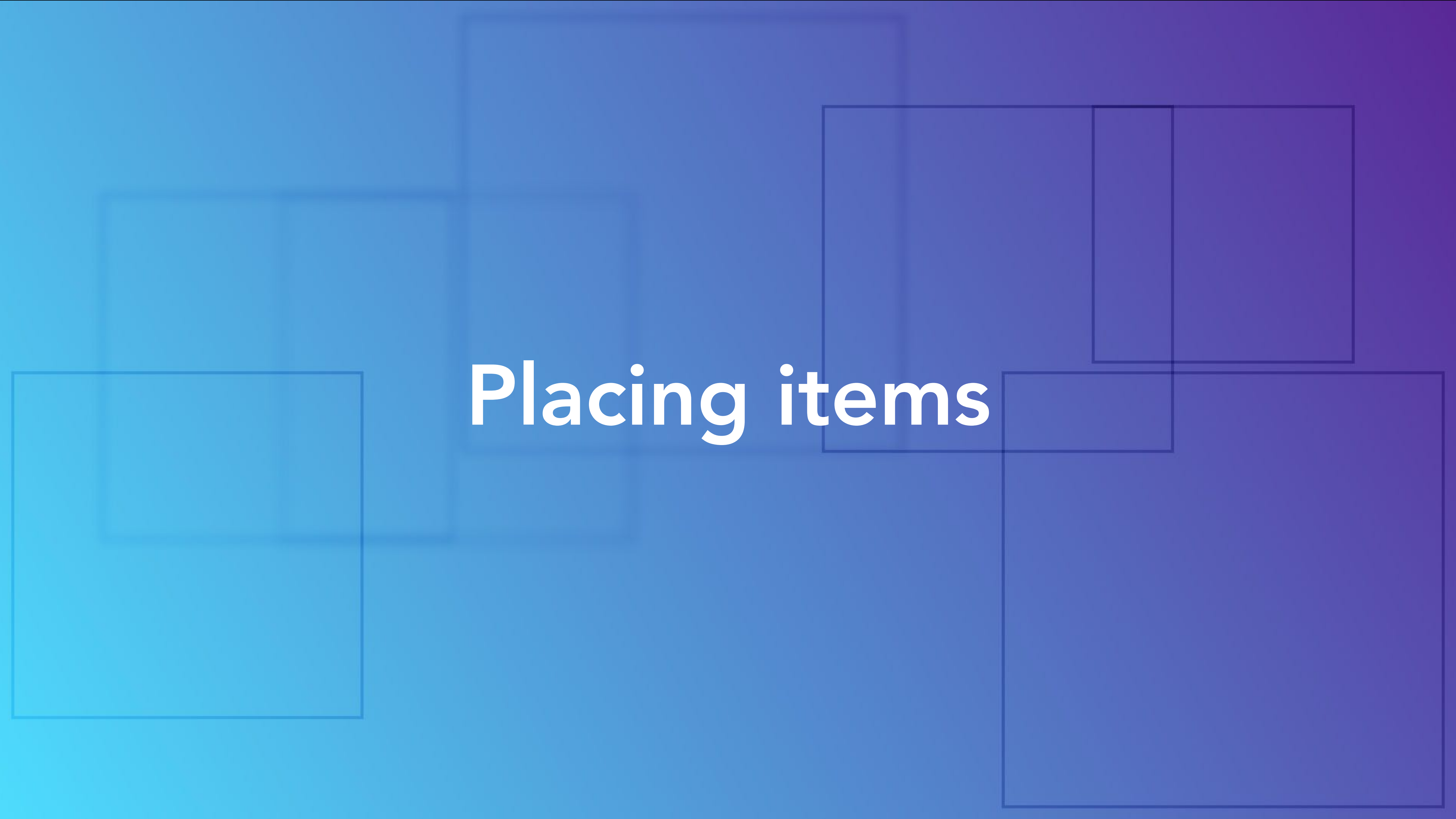
Implicit tracks

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  grid-template-rows: repeat(4, 150px);  
  grid-auto-rows: 150px;  
  grid-gap: 20px;  
}
```



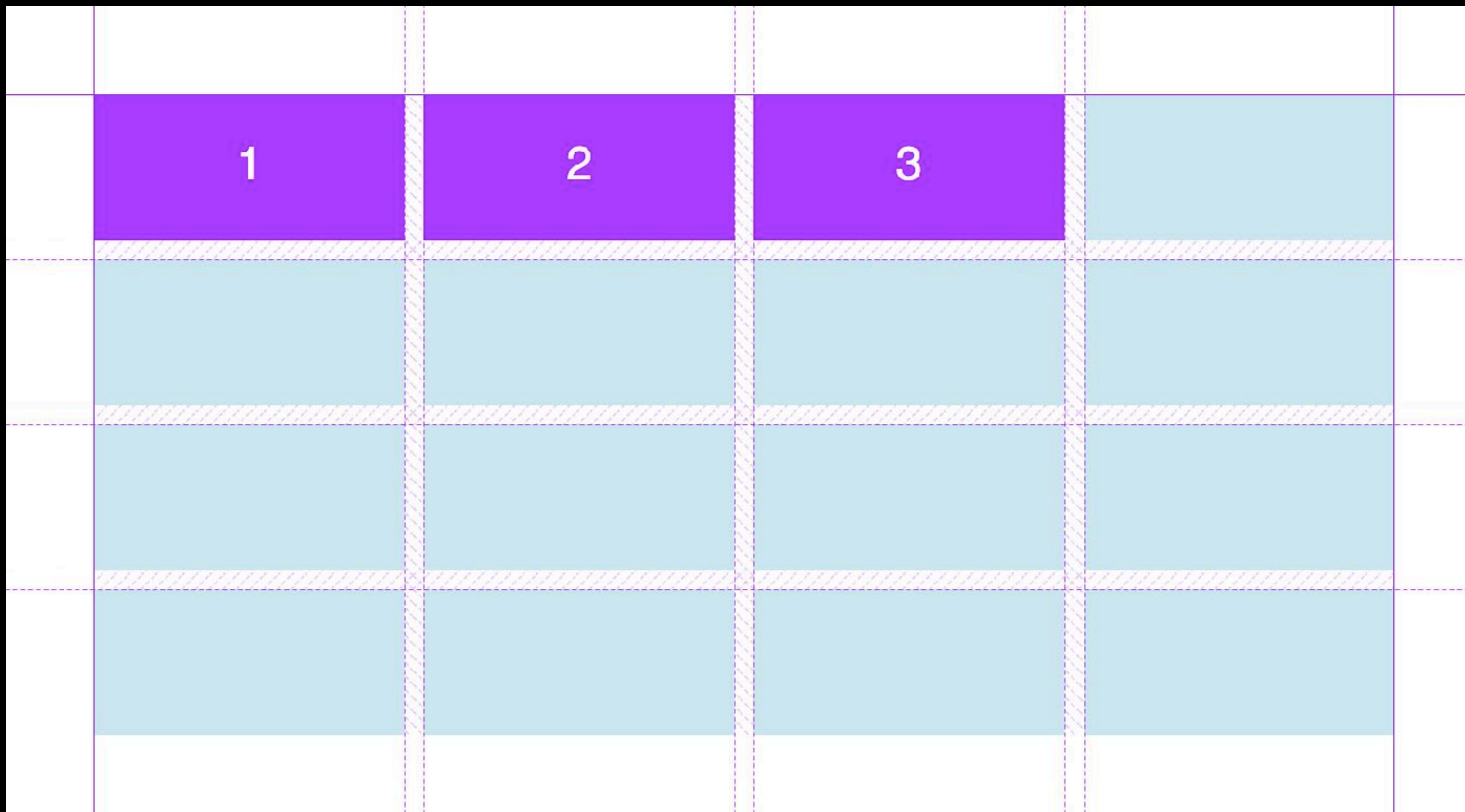
codepen.io/michellebarker/pen/eLZwVg

Placing items



Auto-placement

```
<div class="grid">  
  <div class="grid__item">1</div>  
  <div class="grid__item">2</div>  
  <div class="grid__item">3</div>  
</div>
```

codepen.io/michellebarker/pen/gdrVXP

<header>

<main>

<aside>

Placing items by grid line

```
<div class="grid">  
  <header></header>  
  <main></main>  
  <aside></aside>  
</div>
```


	1	2	3	4	5
1					
2					
3					
4					
5					

Placing items by grid line

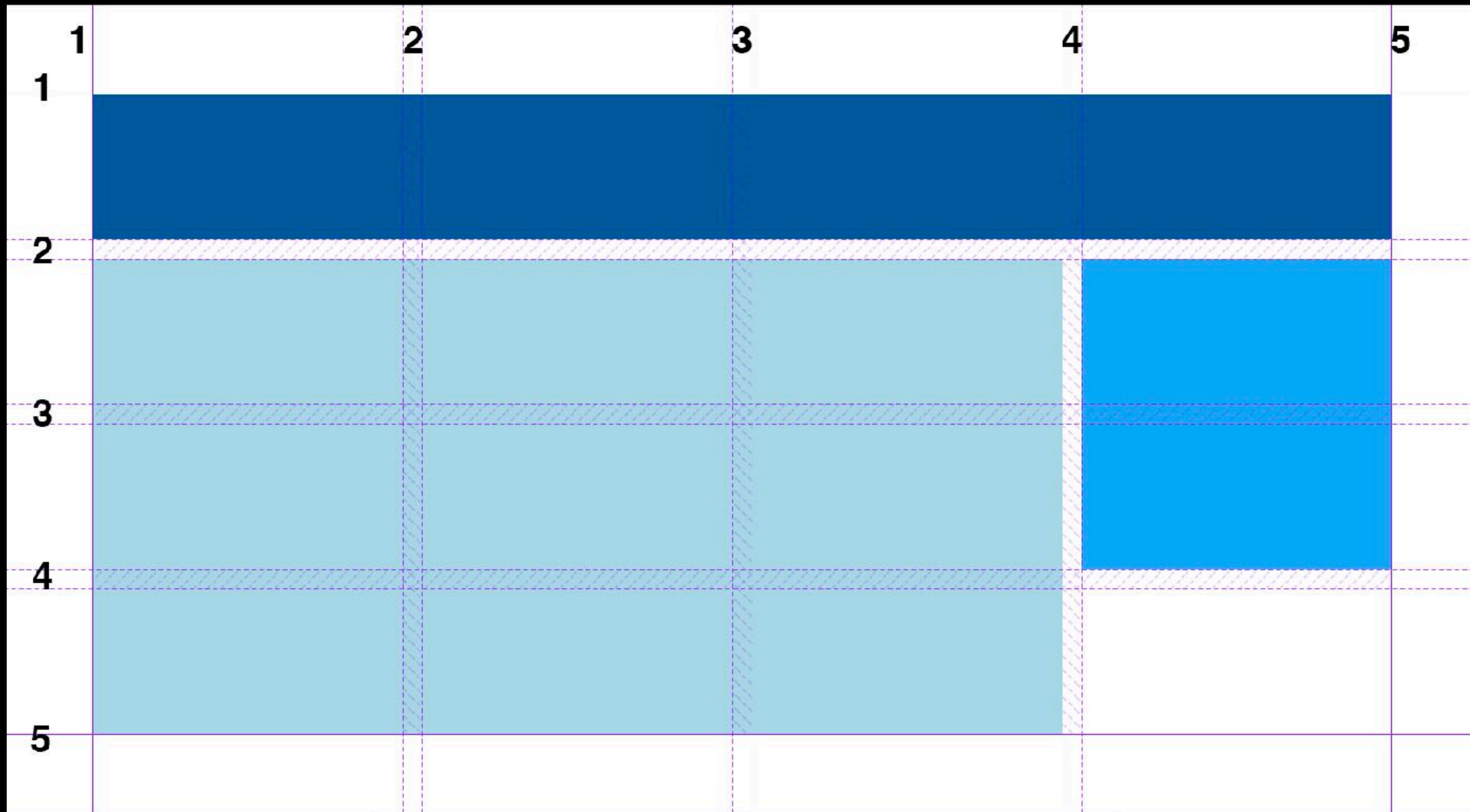
```
header {  
  grid-column-start: 1;  
  grid-column-end: 5;  
}
```

1	2	3	4	5
1				
2				
3				
4				
5				

Placing items by grid line

```
main {  
  grid-column-start: 1;  
  grid-column-end: 4;  
  grid-row-start: 2;  
  grid-row-end: 5;  
}
```

```
aside {  
  grid-column-start: 4;  
  grid-column-end: 5;  
  grid-row-start: 2;  
  grid-row-end: 4;  
}
```

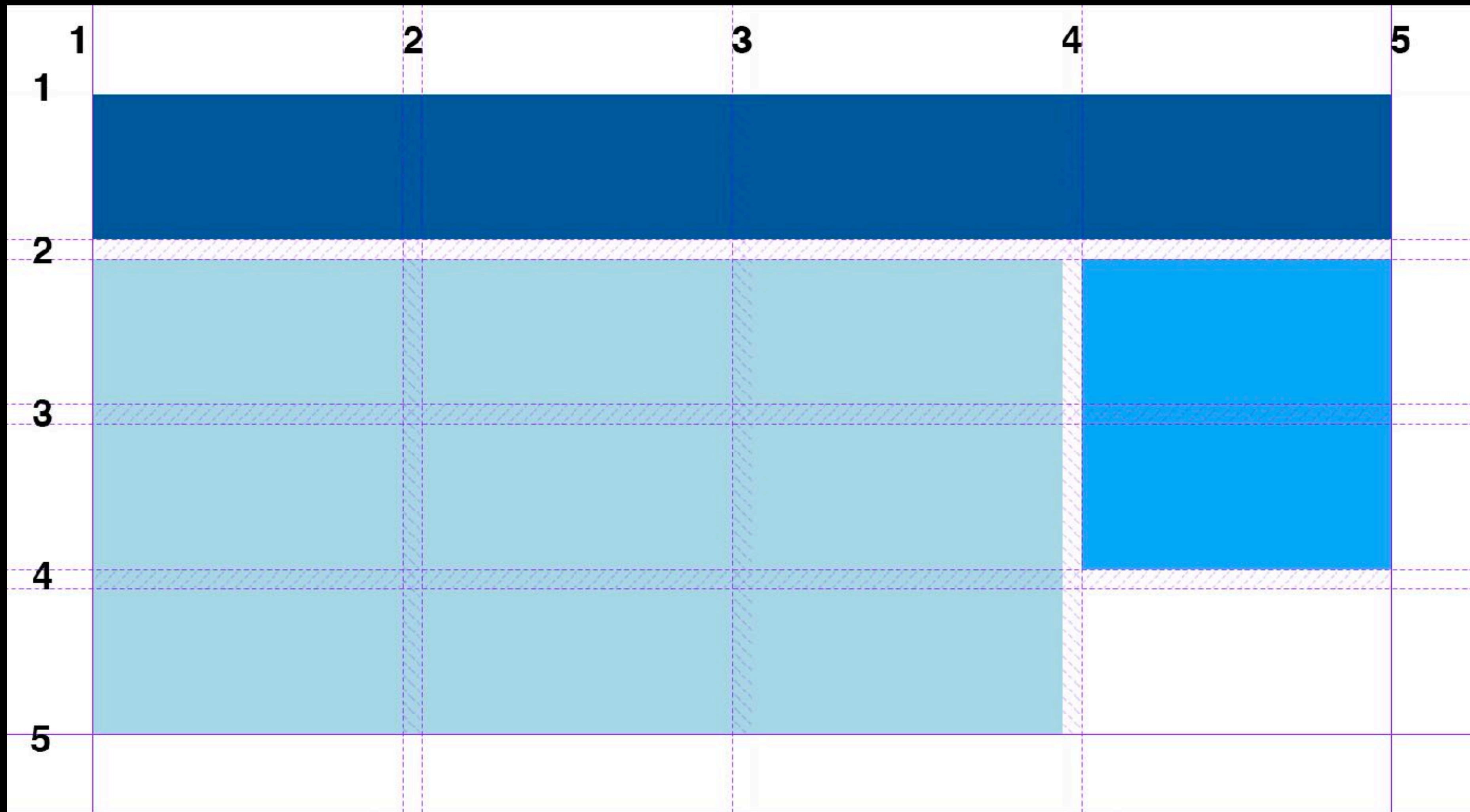



Placing items by grid line

```
header {  
  grid-column: 1 / 5;  
}
```

```
main {  
  grid-column: 1 / 4;  
  grid-row: 2 / 5;  
}
```

```
aside {  
  // grid-column: 4 / 5; – unnecessary as will be  
  auto-placed  
  grid-row: span 2;  
}
```



Placing items by grid line

```
main {  
  grid-column: 1 / 4;  
  grid-row: 2 / 4;  
}
```


Placing items by grid line

```
main {  
  grid-column: span 3;  
  grid-row: 2 / 4;  
}
```

Placing items by grid line

```
main {  
  grid-column: 1 / span 3;  
  grid-row: 2 / 4;  
}
```

Placing items by grid line

```
main {  
  grid-column: span 3 / 4;  
  grid-row: 2 / 4;  
}
```

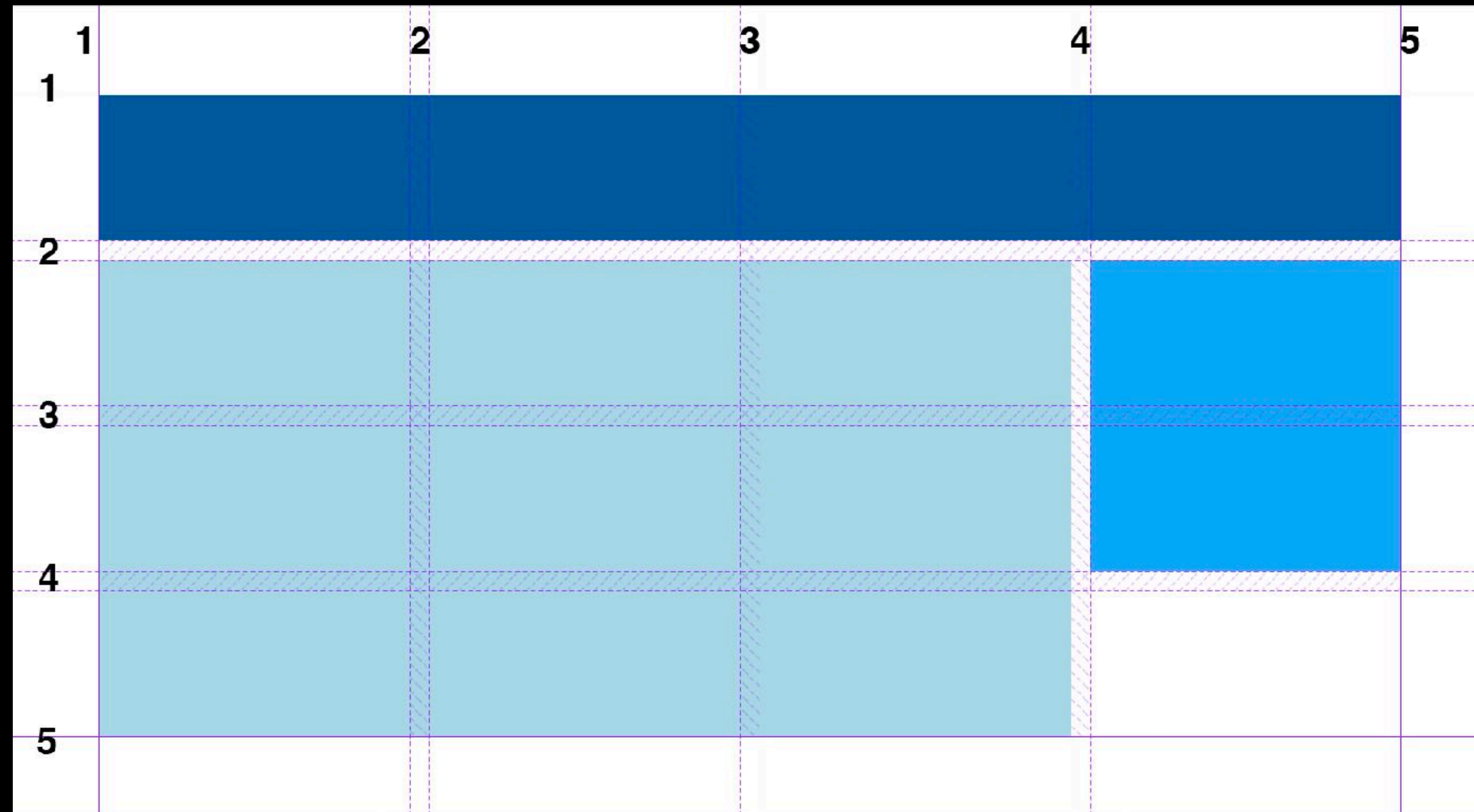

Naming lines

```
.grid {  
  display: grid;  
  grid-template-columns:  
    [header-start main-start]  
    repeat(3, 1fr)  
    [main-end] 1fr [header-end];  
  grid-auto-rows: 150px;  
}
```

header-start
main-start

main-end

header-end



Grid areas

```
header {  
  grid-column: header;  
}
```

```
main {  
  grid-column: main  
  grid-row: 2 / 5;  
}
```

```
aside {  
  grid-row: span 2;  
}
```


grid-template-areas

```
.grid {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  grid-auto-rows: 150px;  
  grid-template-areas: "h h h h"  
                      "m m m a"  
                      "m m m a"  
                      "m m m a"  
                      "m m m .";  
  
  grid-gap: 20px;  
}
```

grid-template-areas

```
header {  
  grid-area: h;  
}
```

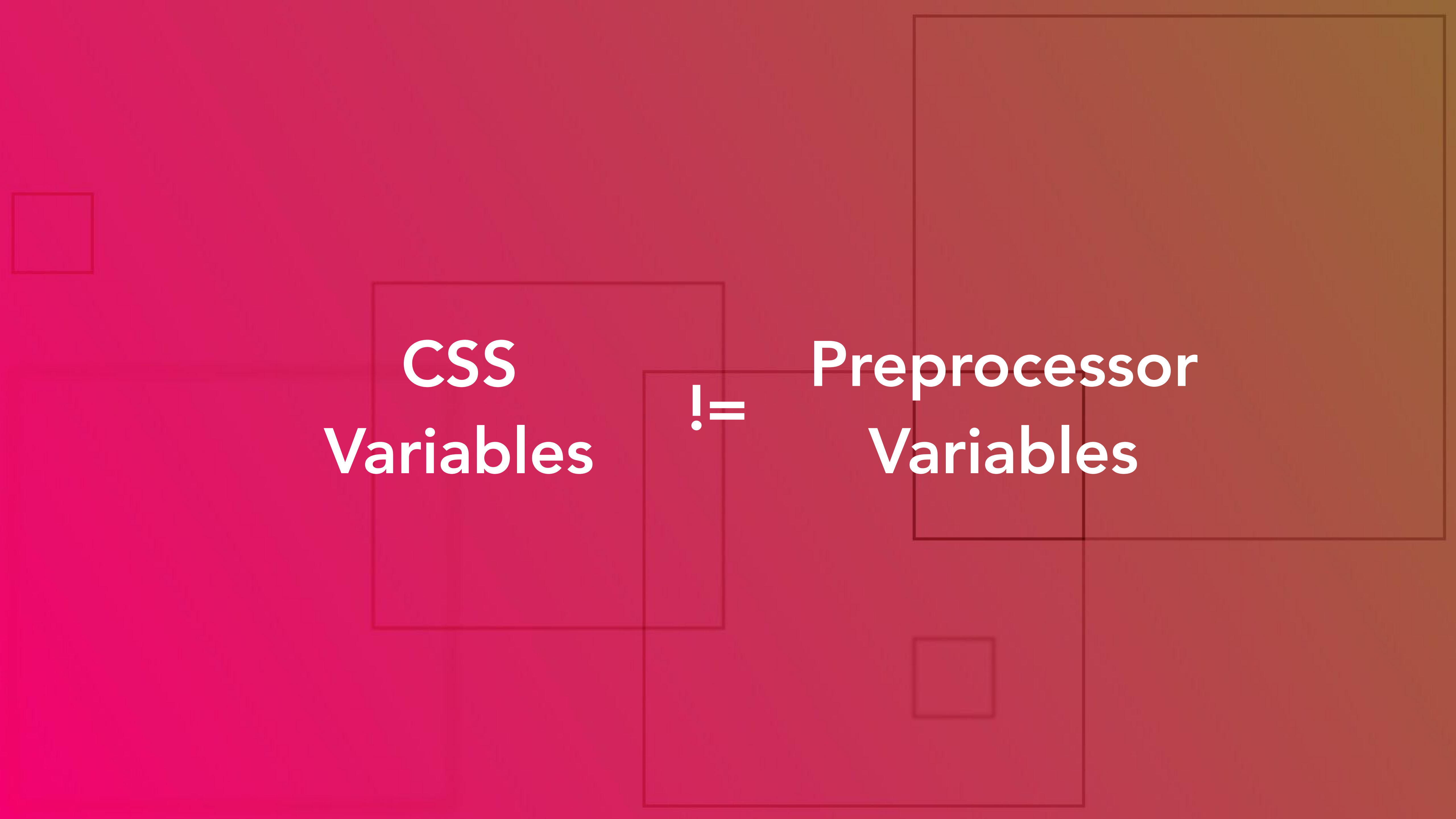
```
main {  
  grid-area: m;  
}
```

```
aside {  
  grid-area: a;  
}
```




CSS Variables

(a.k.a. Custom Properties)



**CSS
Variables**

!=

**Preprocessor
Variables**

CSS
Variables

==

Dynamic
variables

Defining variables

```
:root {  
  --bgColor: red;  
}
```

Using variables

```
.my-component {  
  background-color: var(--bgColor);  
}
```

Defaults

```
.my-component {  
  background-color: var(--bgColor, orange);  
}  
  
.my-component:last-child {  
  --bgColor: red;  
}
```


Defaults

```
.box {  
  background-color: var(--bgColor, orange);  
}  
  
.purple {  
  --bgColor: rebeccaPurple;  
}
```

Defaults

```
<div class="box"></div>
```

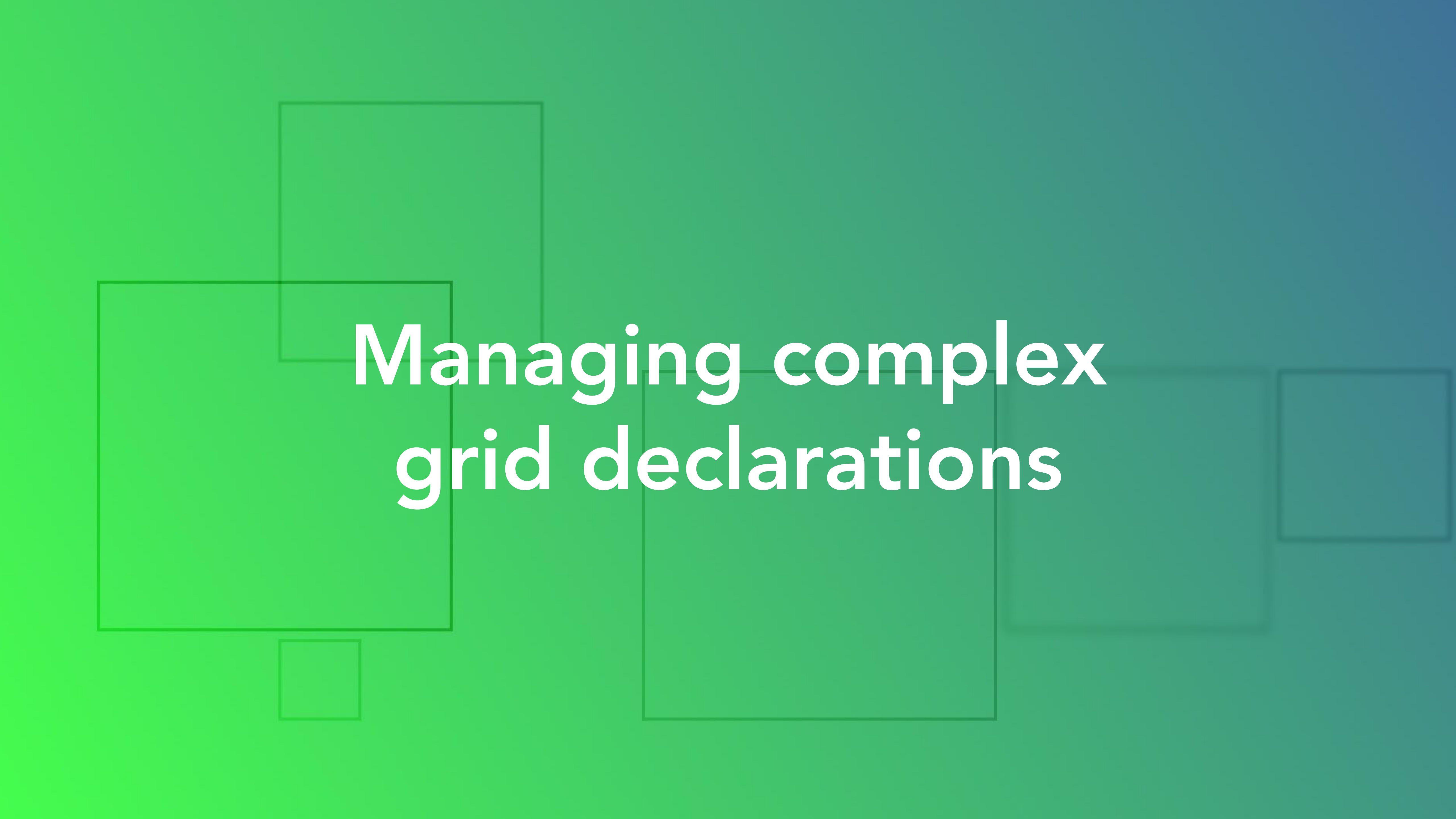
```
.box {  
  background-color:  
    orange;  
}
```

```
<div class="purple">  
  <div class="box"></div>  
</div>
```

```
.box {  
  background-color:  
    rebeccaPurple;  
}
```




CSS Variables + CSS Grid



Managing complex grid declarations

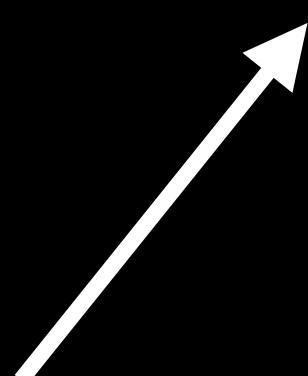
Managing complex grid declarations

```
.grid {  
  display: grid;  
  grid-template-columns:  
    [outer-start] minmax(20px, 1fr) [wrapper-start]  
    repeat(24, minmax(0, 30px))  
    [wrapper-end] minmax(20px, 1fr) [outer-end];  
  grid-template-rows:  
    [outer-start] 1fr [text-top-end] 40px  
    [heading-start] auto [heading-end]  
    40px [text-bottom-start] 1fr [outer-end];  
  grid-gap: 0 20px;  
}
```

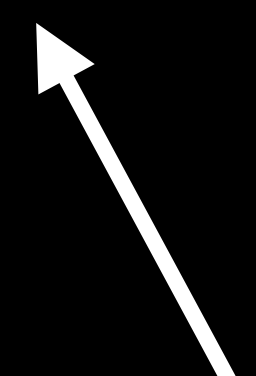

minmax()

minmax(20px, 1fr)

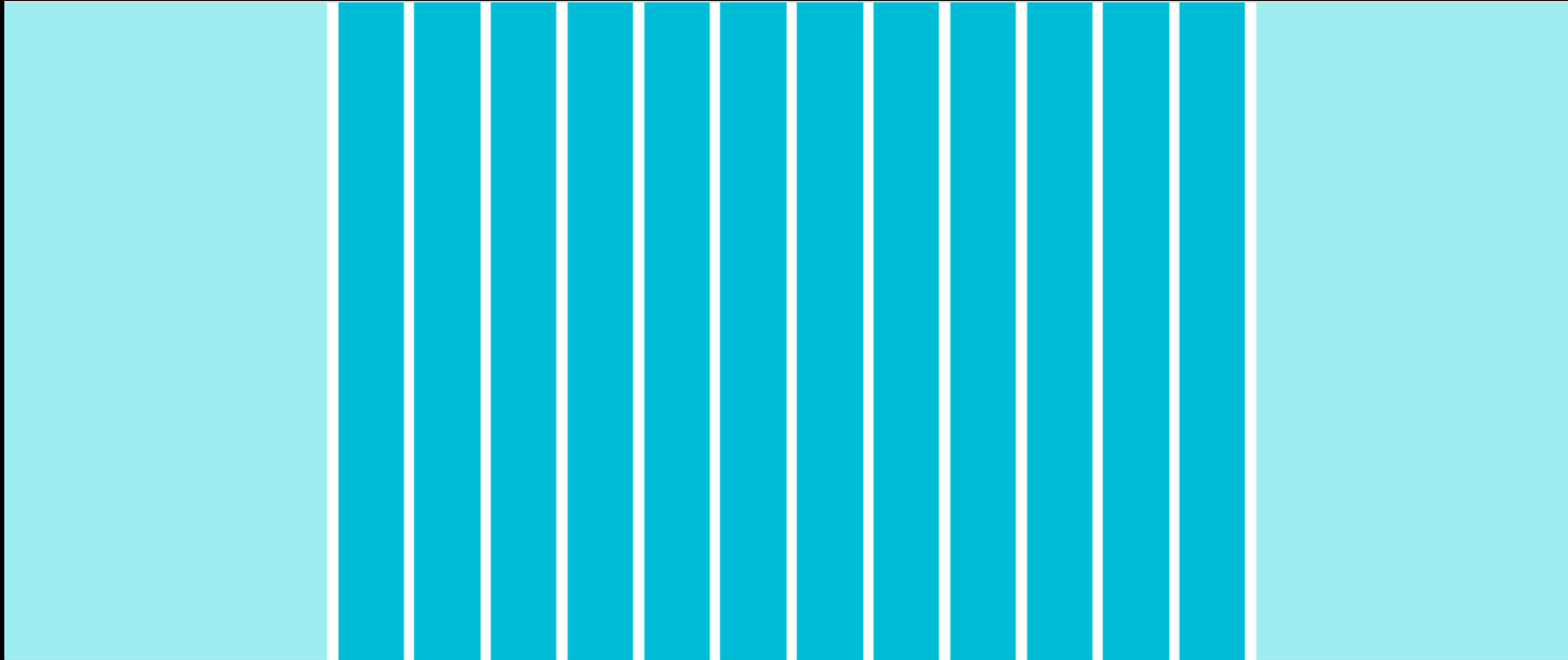
Minimum size

A white arrow pointing from the text 'Minimum size' to the '20px' argument of the minmax function.

Maximum size

A white arrow pointing from the text 'Maximum size' to the '1fr' argument of the minmax function.

minmax()



Managing complex grid declarations

```
.grid {  
  display: grid;  
  grid-template-columns:  
    [outer-start] minmax(20px, 1fr) [wrapper-start]  
    repeat(24, minmax(0, 30px))  
    [wrapper-end] minmax(20px, 1fr) [outer-end];  
  grid-template-rows:  
    [outer-start] 1fr [text-top-end] 40px  
    [heading-start] auto [heading-end]  
    40px [text-bottom-start] 1fr [outer-end];  
  grid-gap: 0 20px;  
}
```


Managing complex grid declarations

```
:root {  
  --colWidth: 30px;  
  --gutter: 20px;  
  
  @media (min-width: 1600px) {  
    --colWidth: 50px;  
    --gutter: 30px;  
  }  
}
```

Managing complex grid declarations

```
.grid {  
  display: grid;  
  grid-template-columns:  
    [outer-start] minmax(20px, 1fr) [wrapper-start]  
    repeat(24, minmax(0, var(--colWidth)))  
    [wrapper-end] minmax(20px, 1fr) [outer-end];  
  grid-template-rows:  
    [outer-start] 1fr [text-top-end] 40px  
    [heading-start] auto [heading-end]  
    40px [text-bottom-start] 1fr [outer-end];  
  grid-gap: 0 var(--gutter);  
}
```

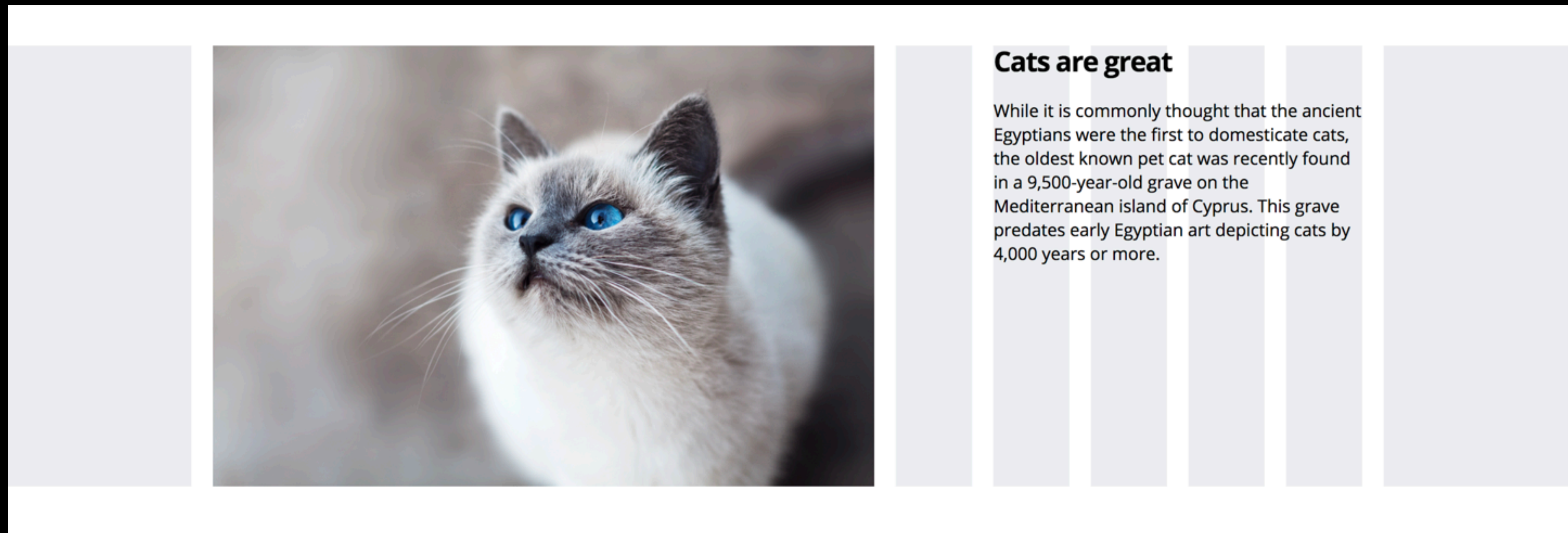



Managing component variants

Managing component variants

```
<article class="grid">  
  <figure class="grid__img"></figure>  
  <div class="grid__text"></div>  
</article>
```

Managing component variants



Managing component variants

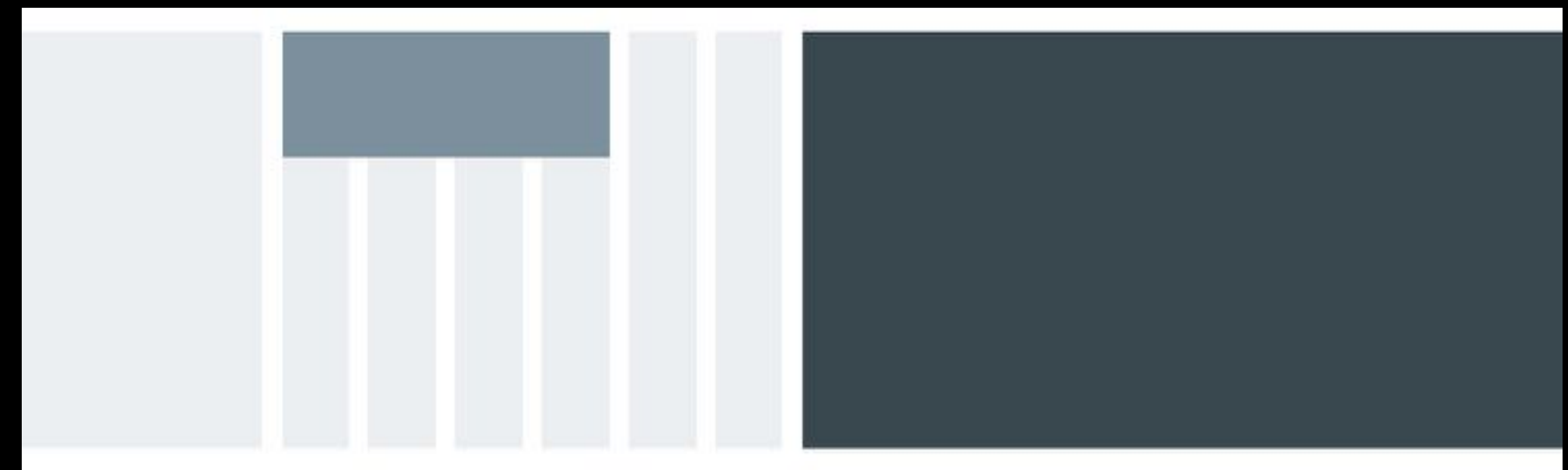
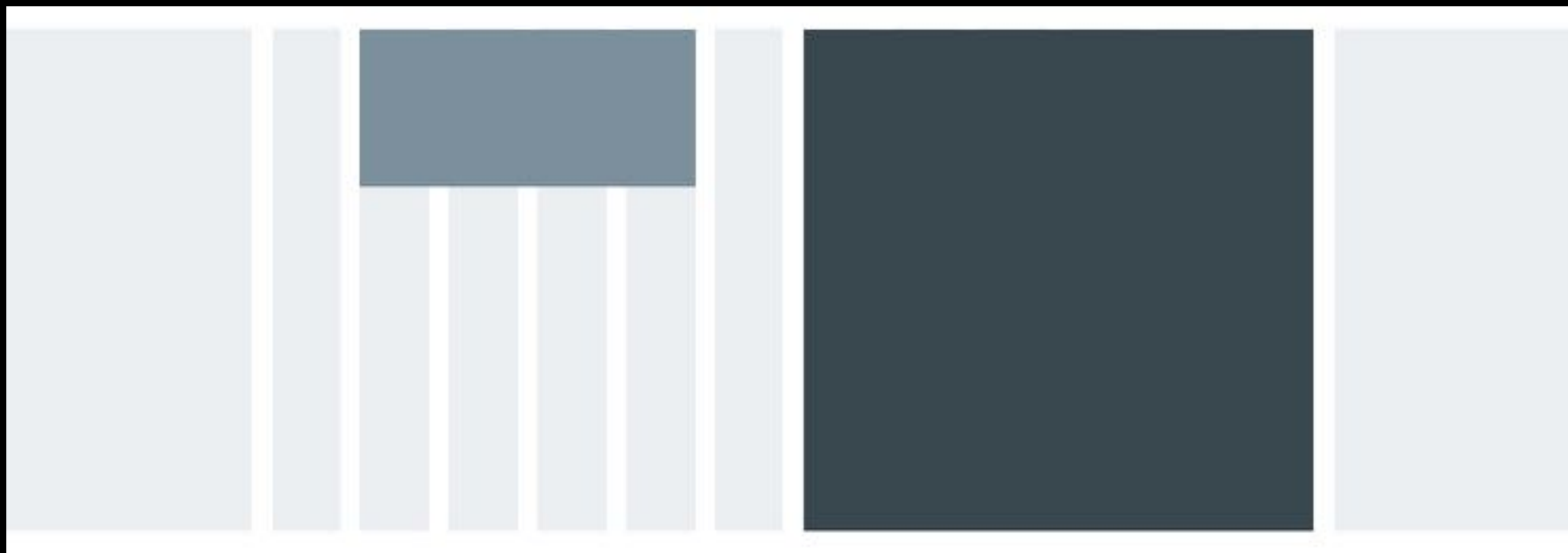
```
.grid {  
  display: grid;  
  grid-template-columns:  
    [start] minmax(20px, 1fr) [wrapper-start]  
    repeat(12, minmax(0, 70px))  
    [wrapper-end] minmax(20px, 1fr) [end];  
  grid-gap: 20px;  
}
```

Managing component variants

```
.grid__img {  
  grid-column: wrapper-start / 9;  
}
```

```
.grid__text {  
  grid-column: 10 / span 4;  
}
```

Managing component variants



codepen.io/michellebarker/pen/XBPMZZ

Managing component variants

```
.grid__img {  
  grid-column:  
    var(--imgStart, wrapper-start) / var(--imgEnd, 6);  
  grid-row: 1;  
}
```

```
.grid__text {  
  grid-column:  
    var(--textStart, 10) / span 4;  
  grid-row: 1;  
}
```

Managing component variants

```
.grid--a {  
  --textStart: 9;  
  --imgStart: wrapper-start;  
  --imgEnd: end;  
}
```

```
.grid--b {  
  --textStart: 9;  
  --imgStart: start;  
}
```

```
.grid--c {  
  --imgEnd: 15;  
}
```




CSS Variables + Javascript

Get property value:

```
getComputedStyle(element).getPropertyValue("--myVariable")
```

(Returns a string)

Set property value:

```
element.style.setProperty("--myVariable", 4)
```

Demo

CSS Grid background generator

codepen.io/michellebarker/pen/xabrLv

Resources

Grid by Example (Rachel Andrew)

gridbyexample.com

Layout Land (Jen Simmons)

layout.land

CSS Grid Playground (MDN)

mozilladevelopers.github.io/playground/css-grid

CSS { In Real Life }

css-irl.info



Is Grid the silver bullet?



Thanks for listening

@mbarker_84 / @CSSInRealLife