Super-powered Layouts
with
CSS Grid +
CSS Variables

CSS Grid

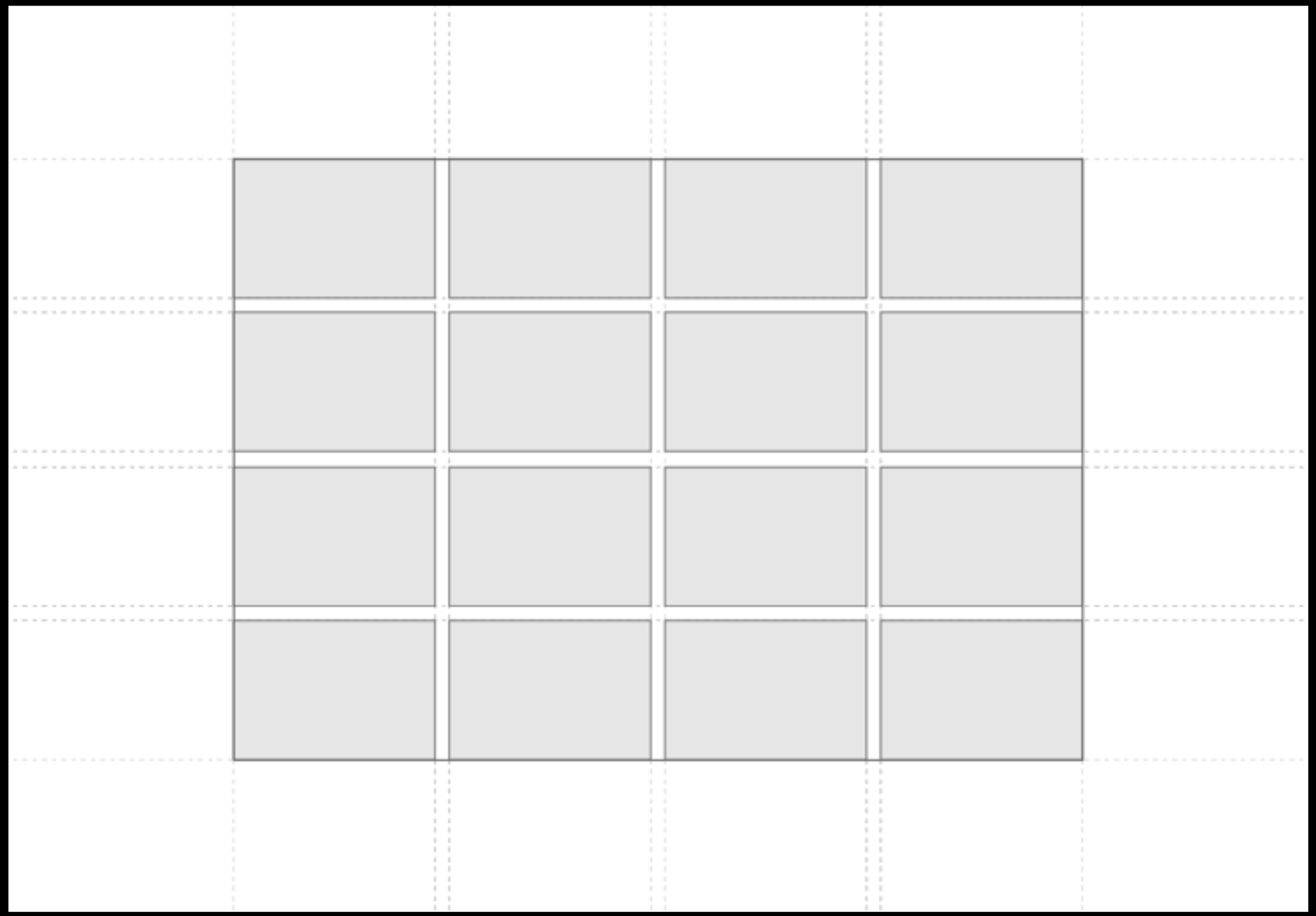
(CSS Grid Layout Module Level 1)

Why not just use flexbox?

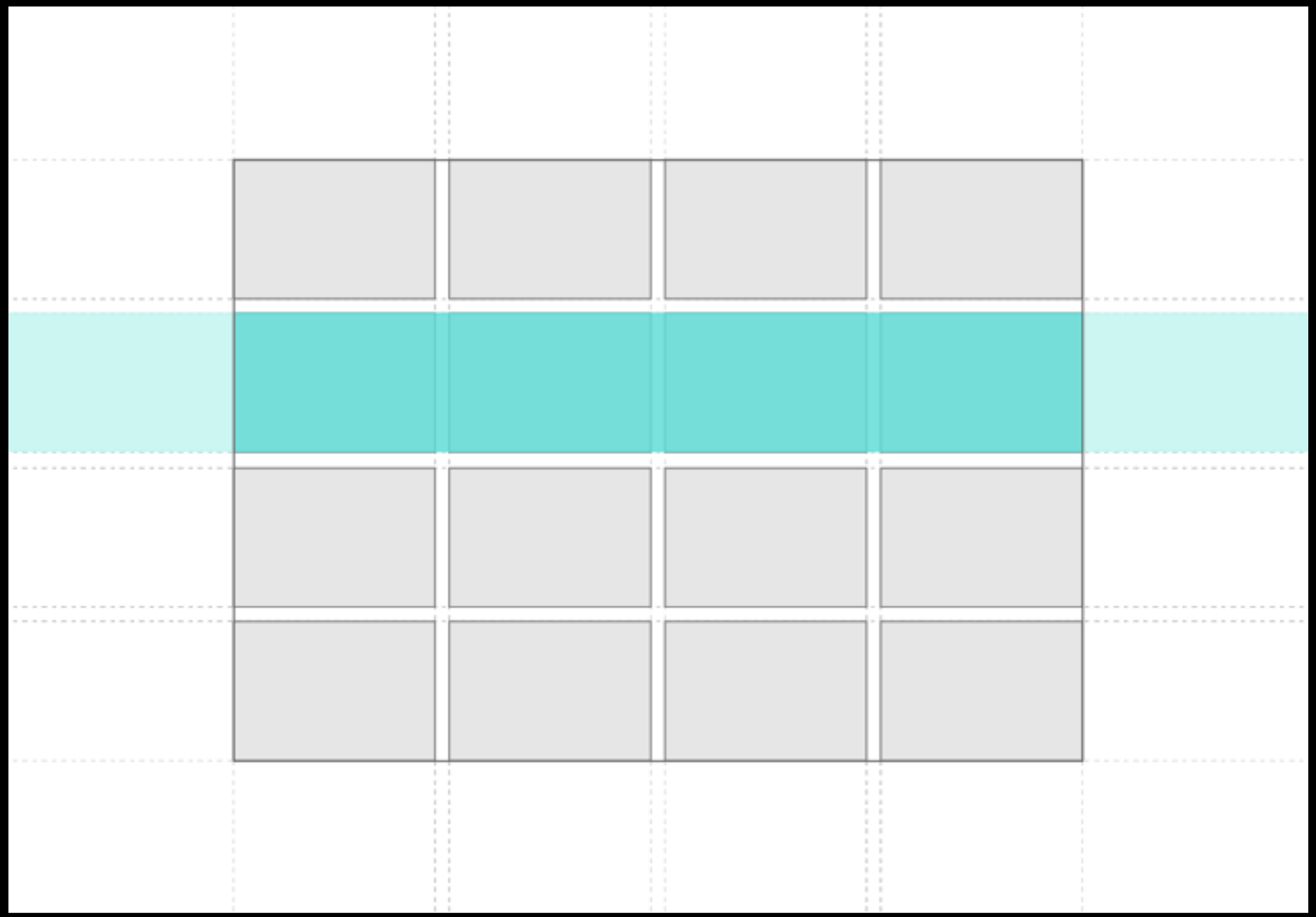
Grid terminology

- Grid

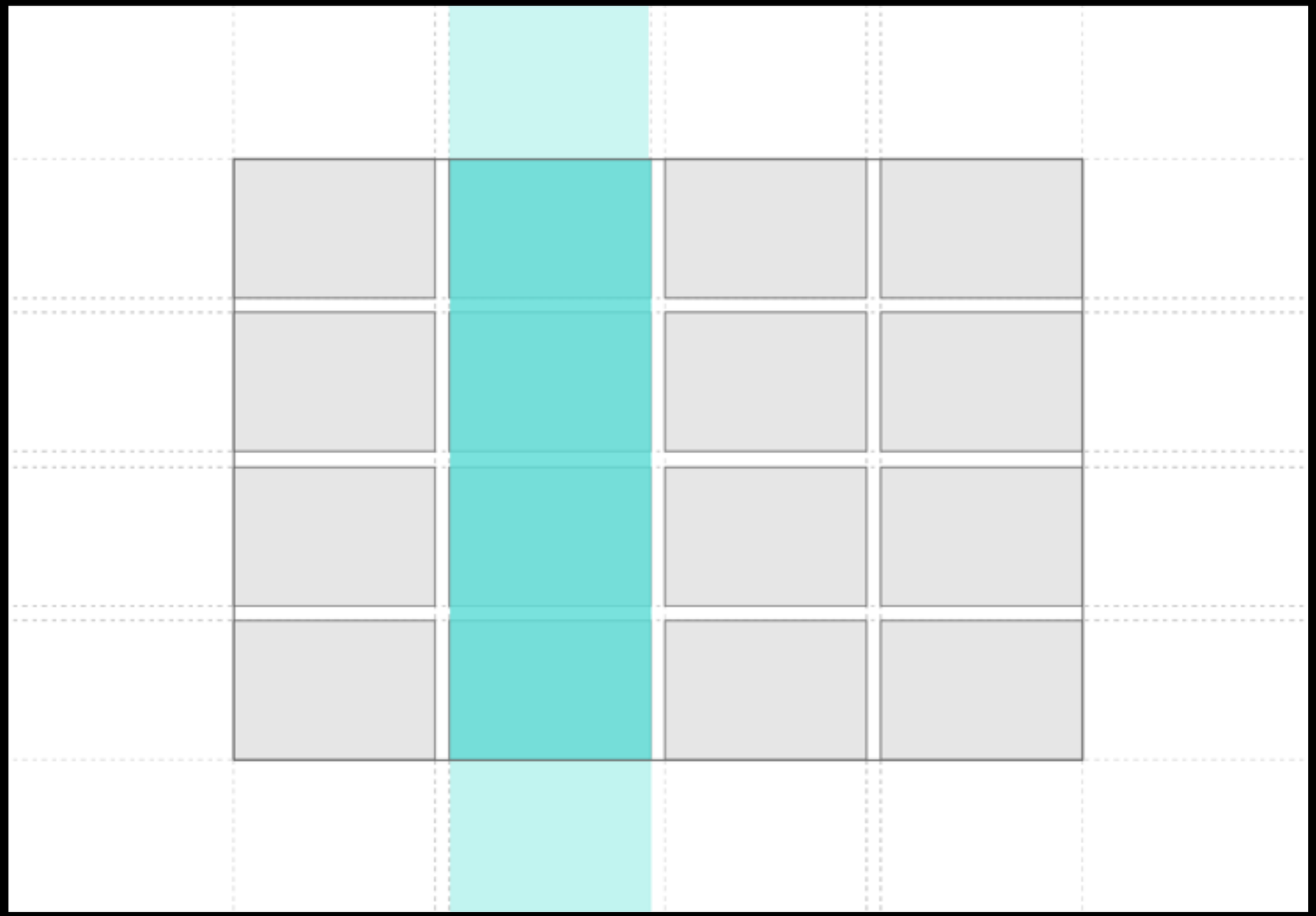
{ display: grid; }



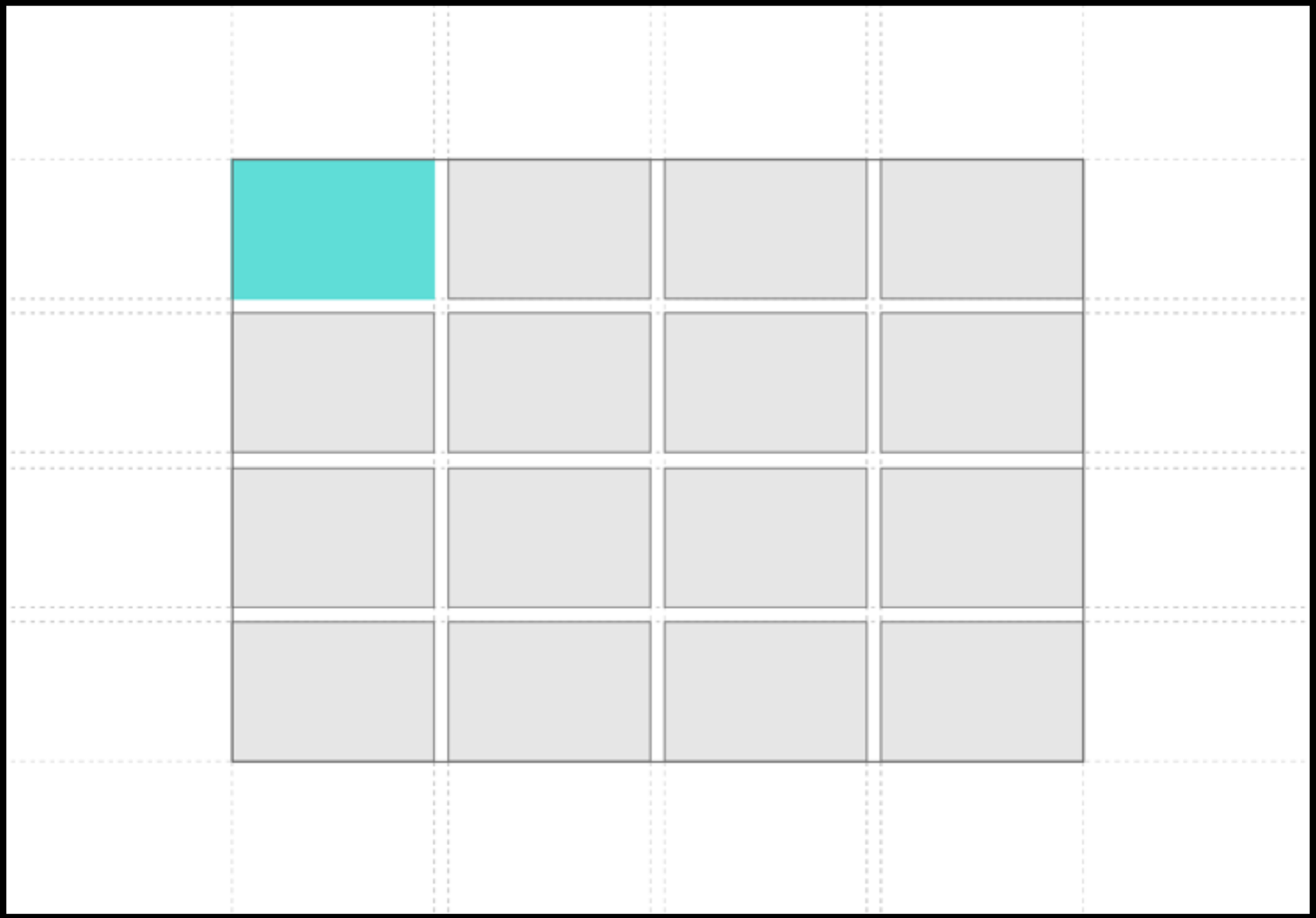
- # Tracks
  (rows and columns)



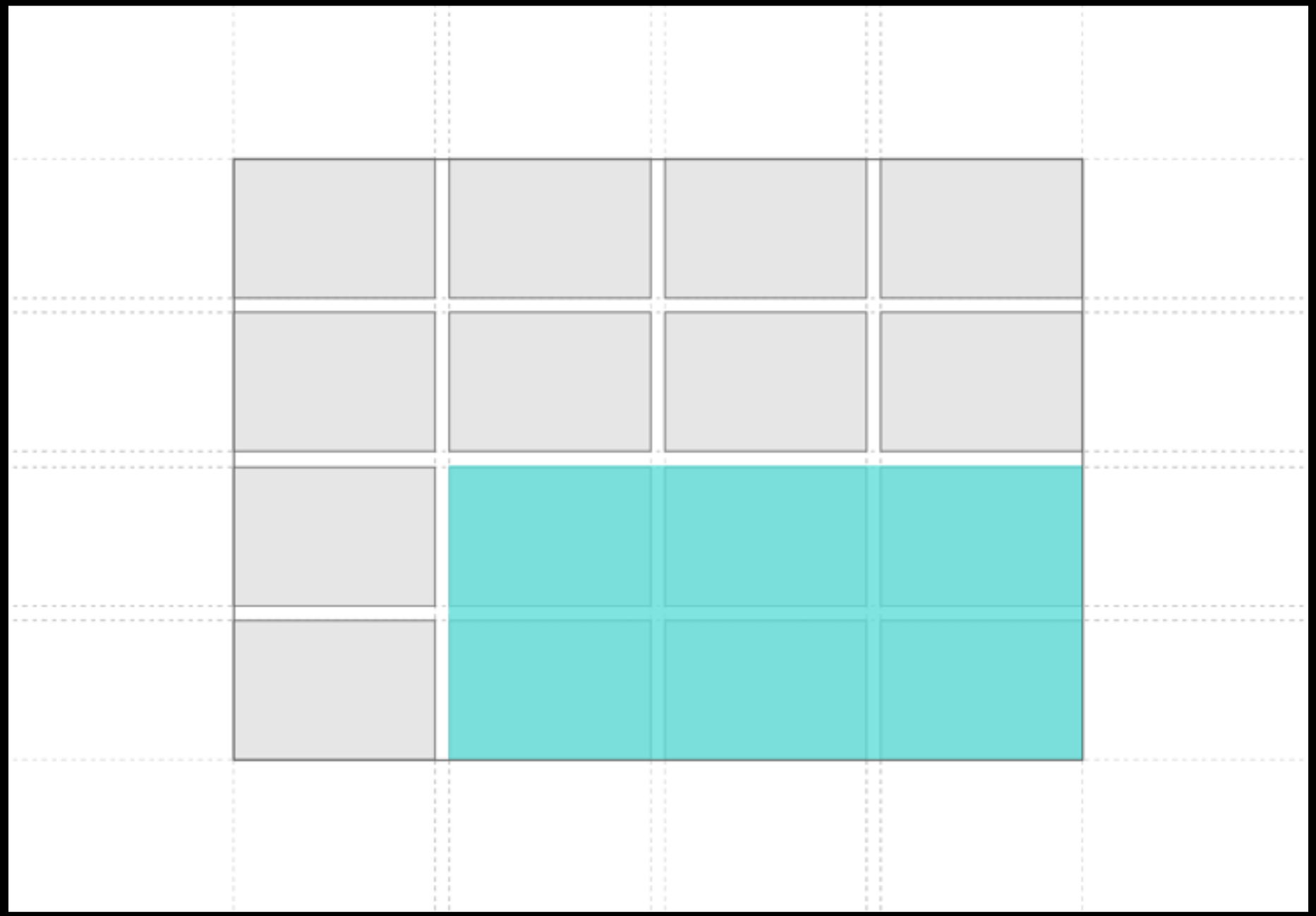
- Tracks
  (rows and columns)



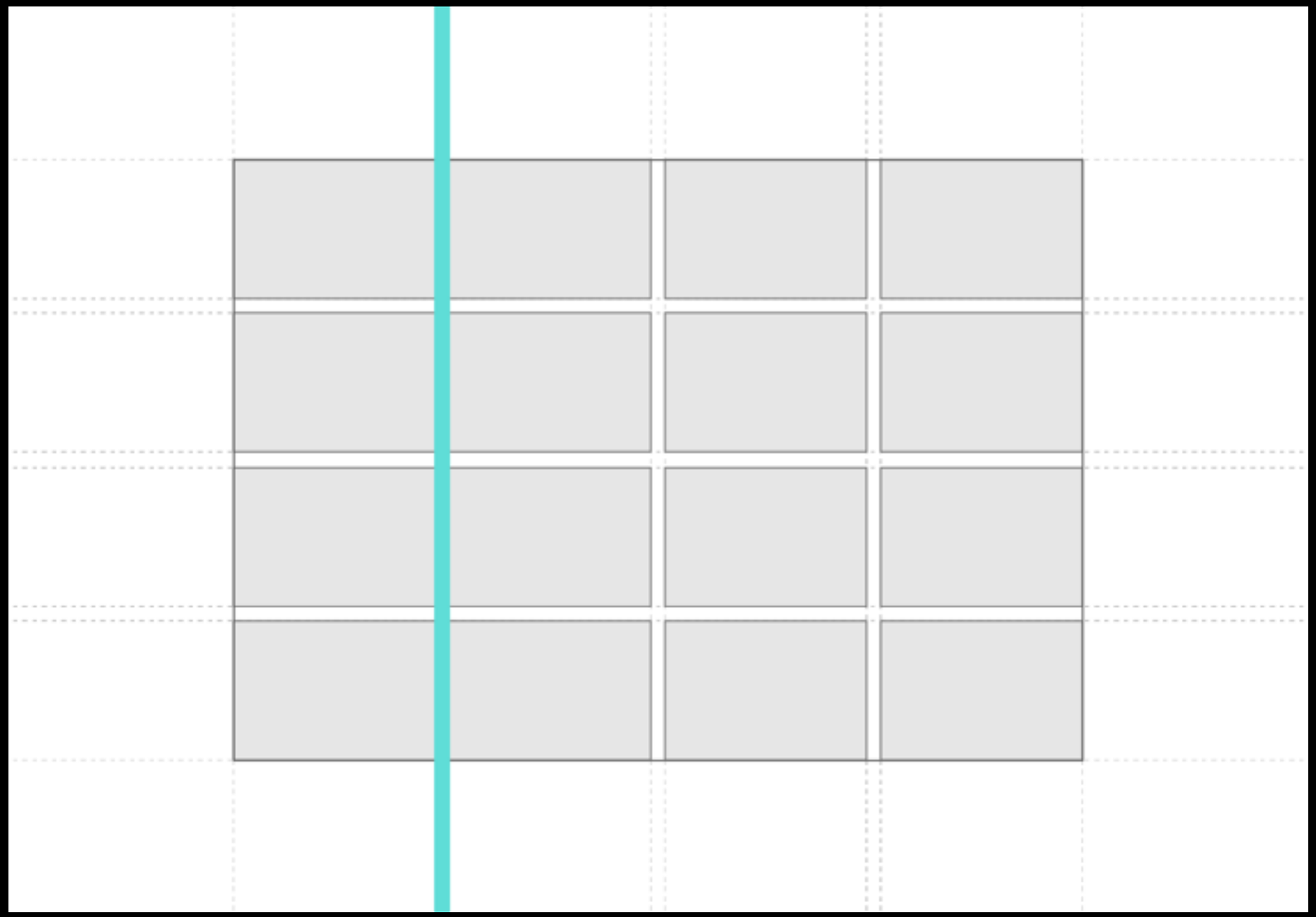
- Cells



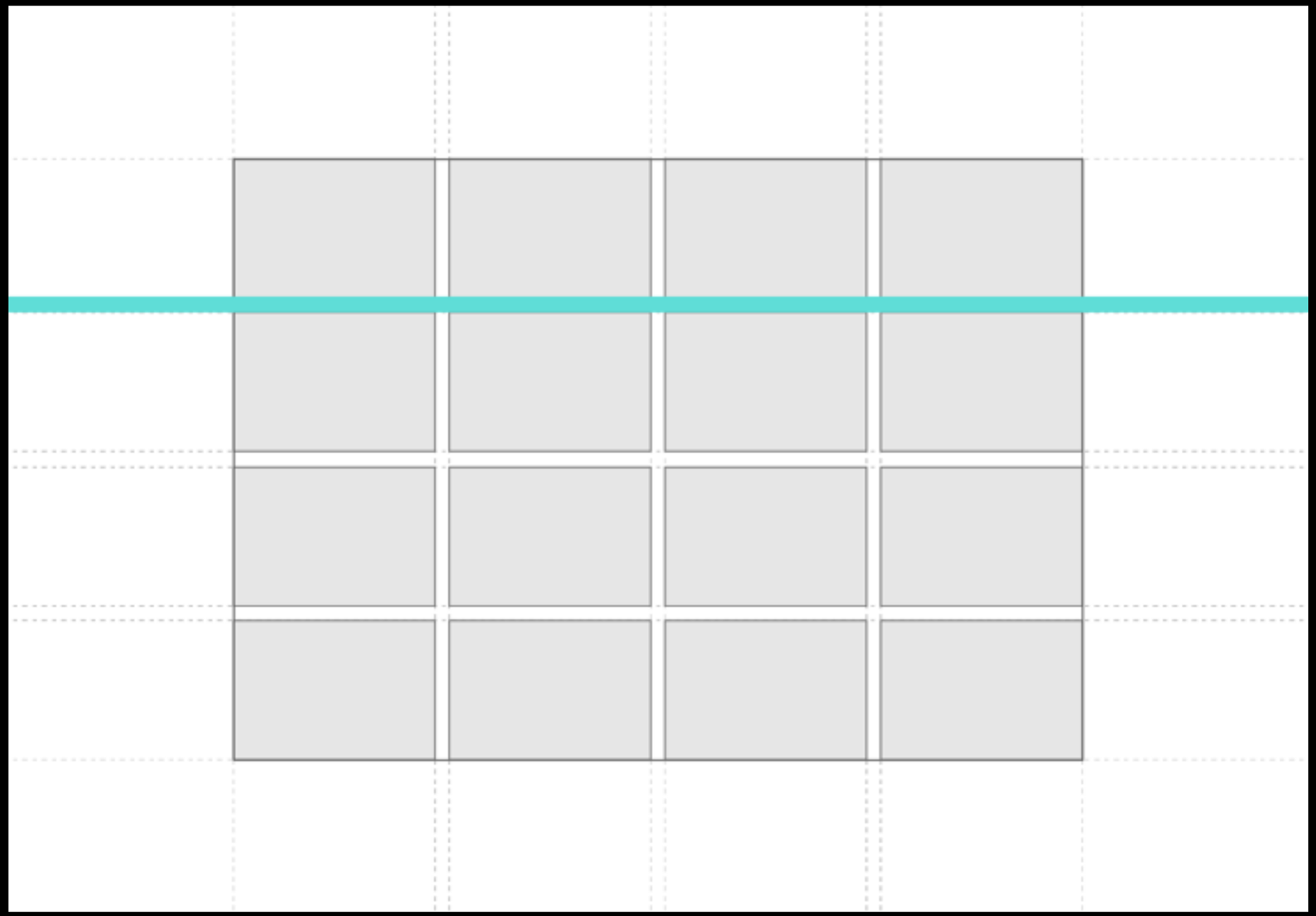
- Grid-area



- 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;
}
```

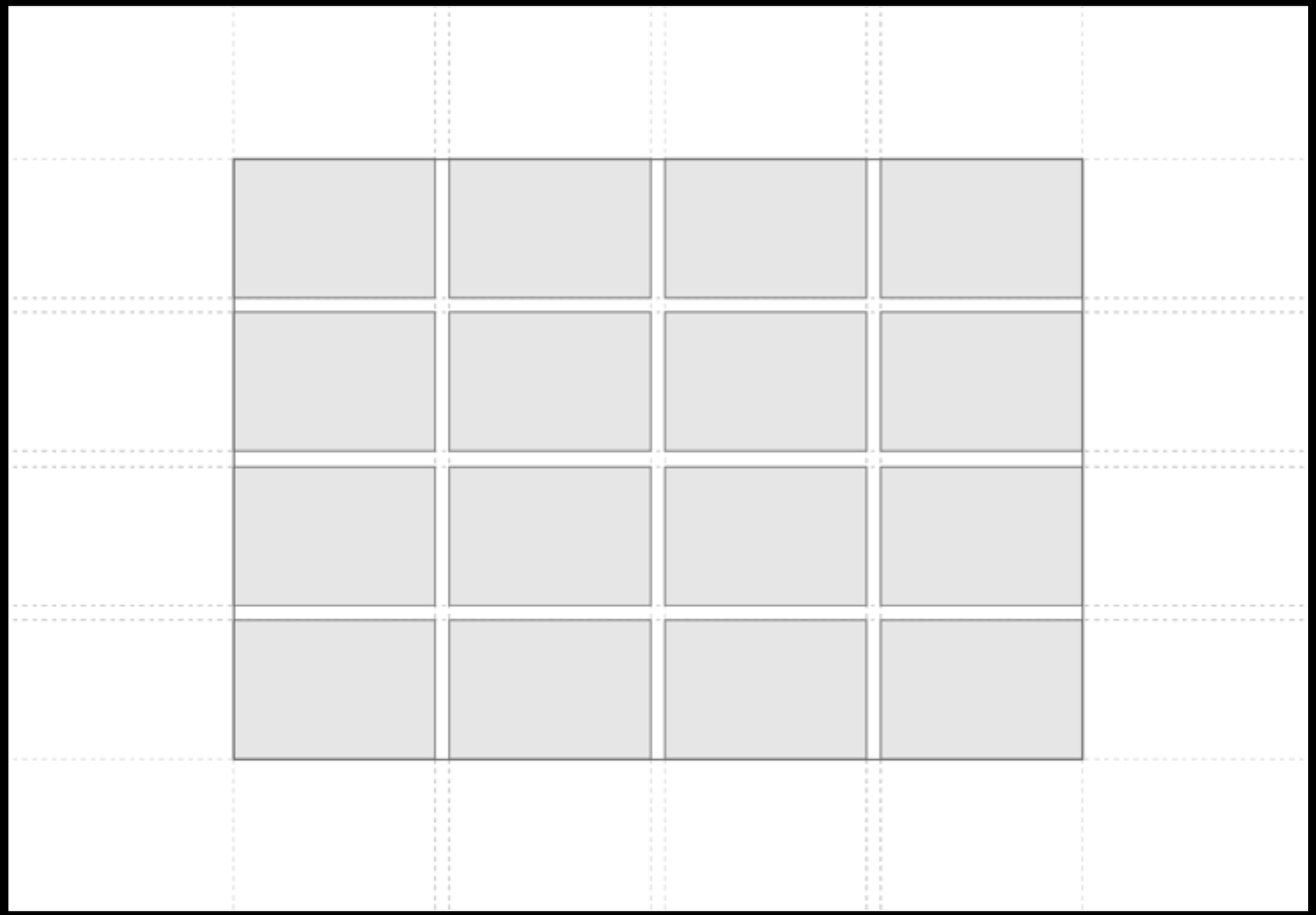
bit.ly/2K2gJSe

Defining a grid

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;
}
```

bit.ly/2K2gJSe



bit.ly/2K2gJSe

repeat()

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

bit.ly/2K2gJSe

Shorthand

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

bit.ly/2K2gJSe

Shorthand

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

grid-row-gap /
grid-column gap

bit.ly/2K2gJSe

`fr`

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

bit.ly/2K2gJSe

`fr`

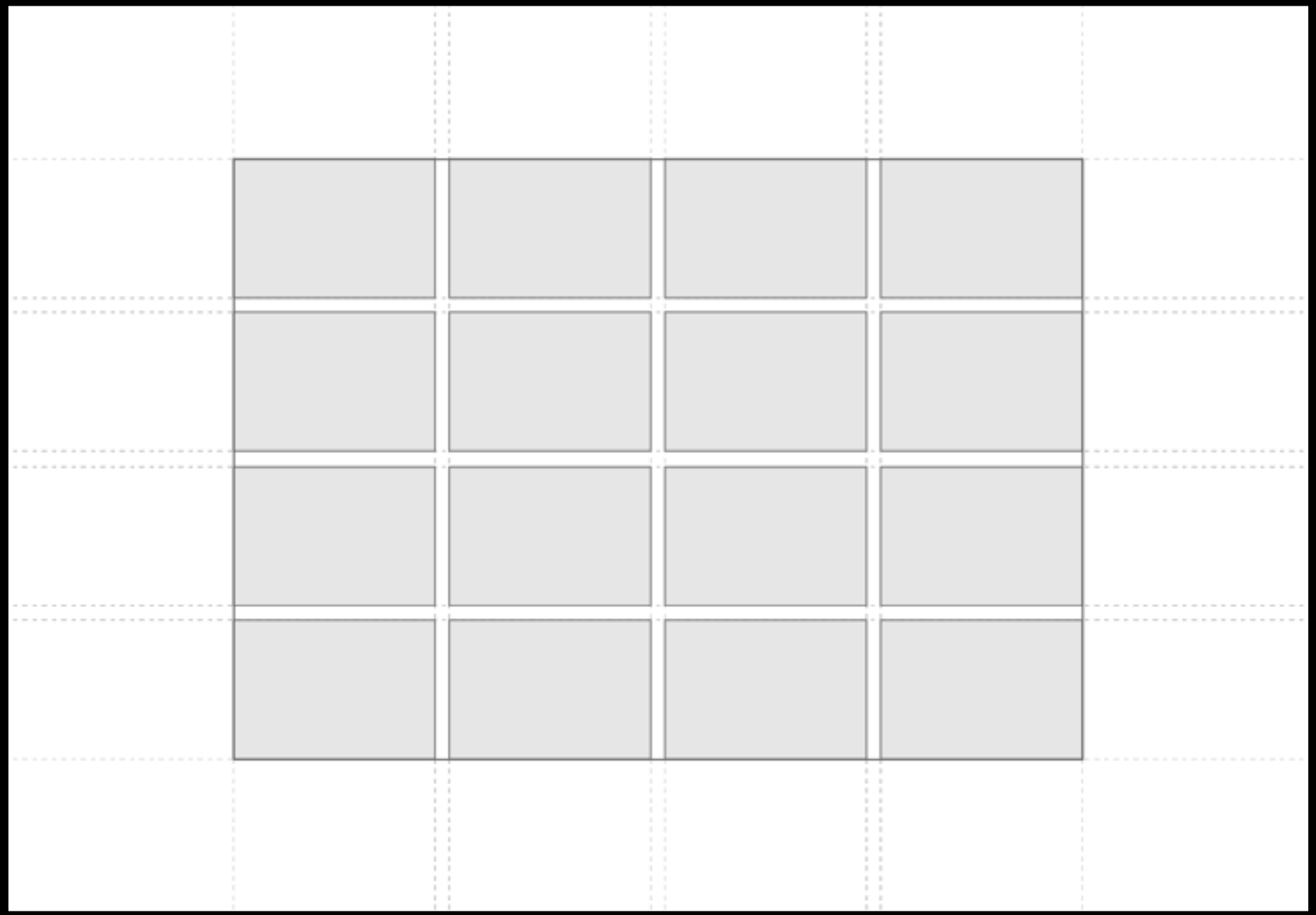
```css
.grid {
  display: grid;
  grid-template-columns: repeat(3, 200px) 1fr;
  grid-auto-rows: 150px;
  grid-gap: 20px;
}
```

bit.ly/2K2gJSe

Implicit tracks

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

bit.ly/2K2gJSe



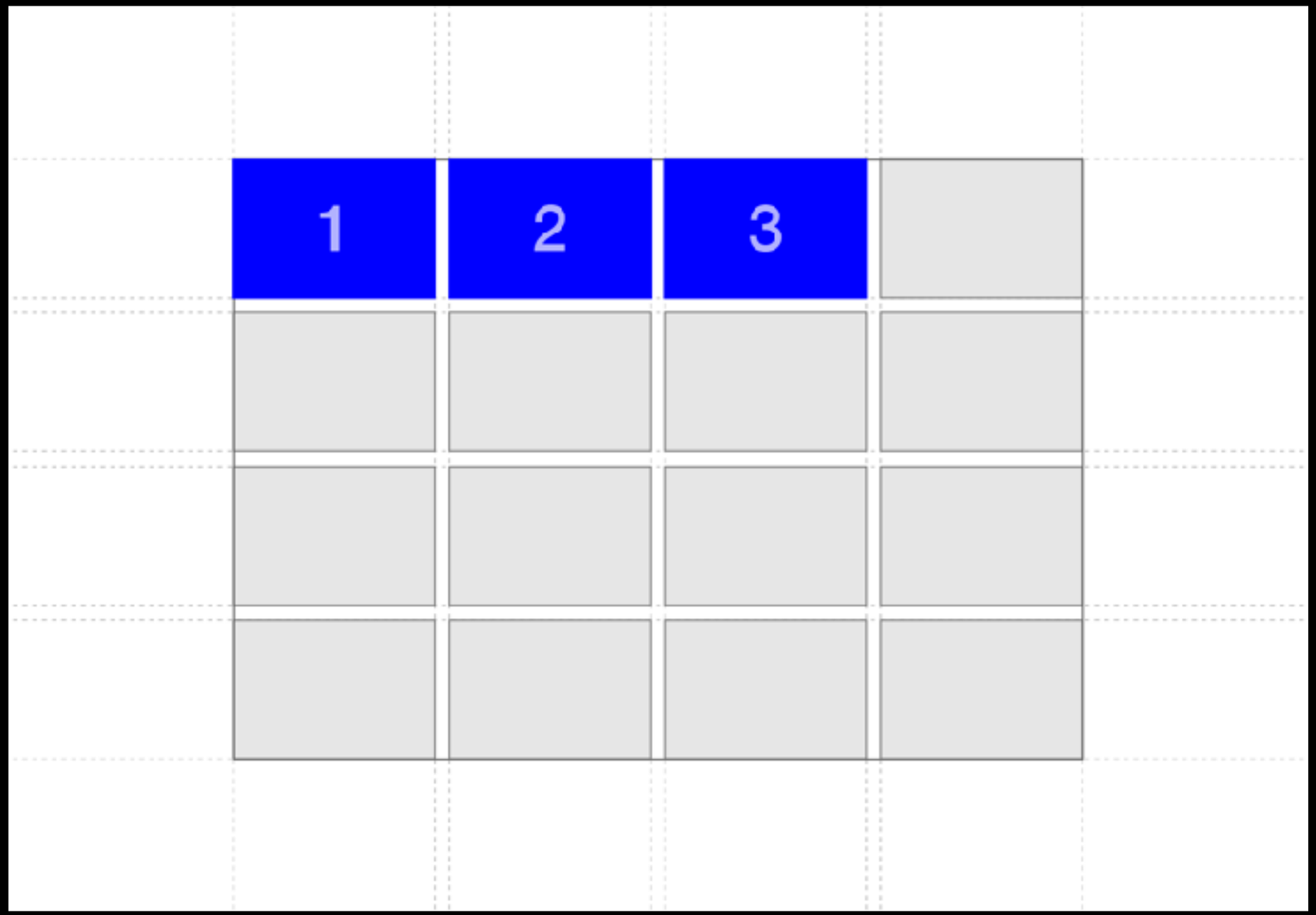
bit.ly/2K2gJSe

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>
```

bit.ly/2K2gJSe



bit.ly/2K2gJSe

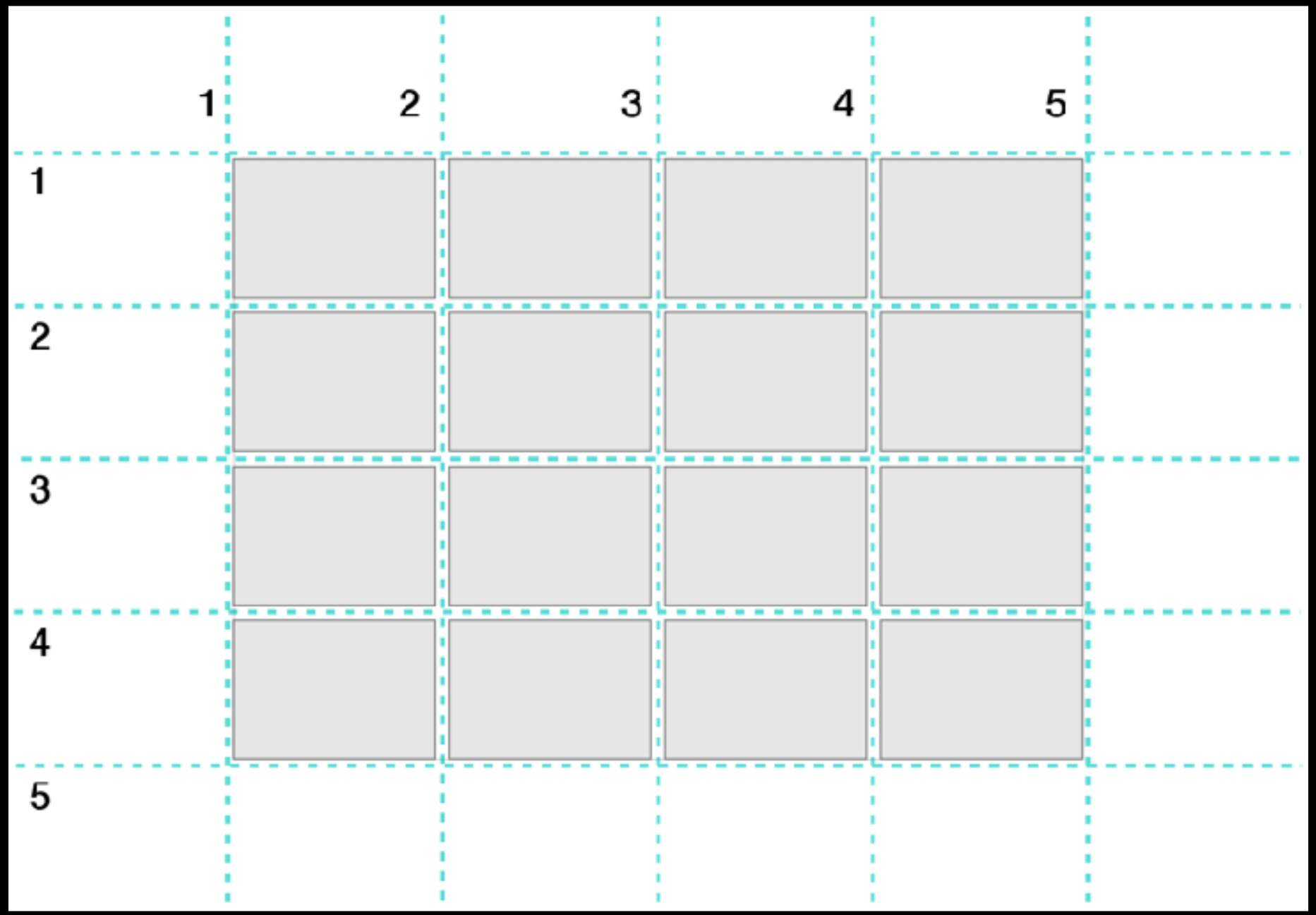
Placing items

```
<div class="grid">

  <header></header>
  <article></article>
  <aside></aside>

</div>
```

bit.ly/2K2gJSe



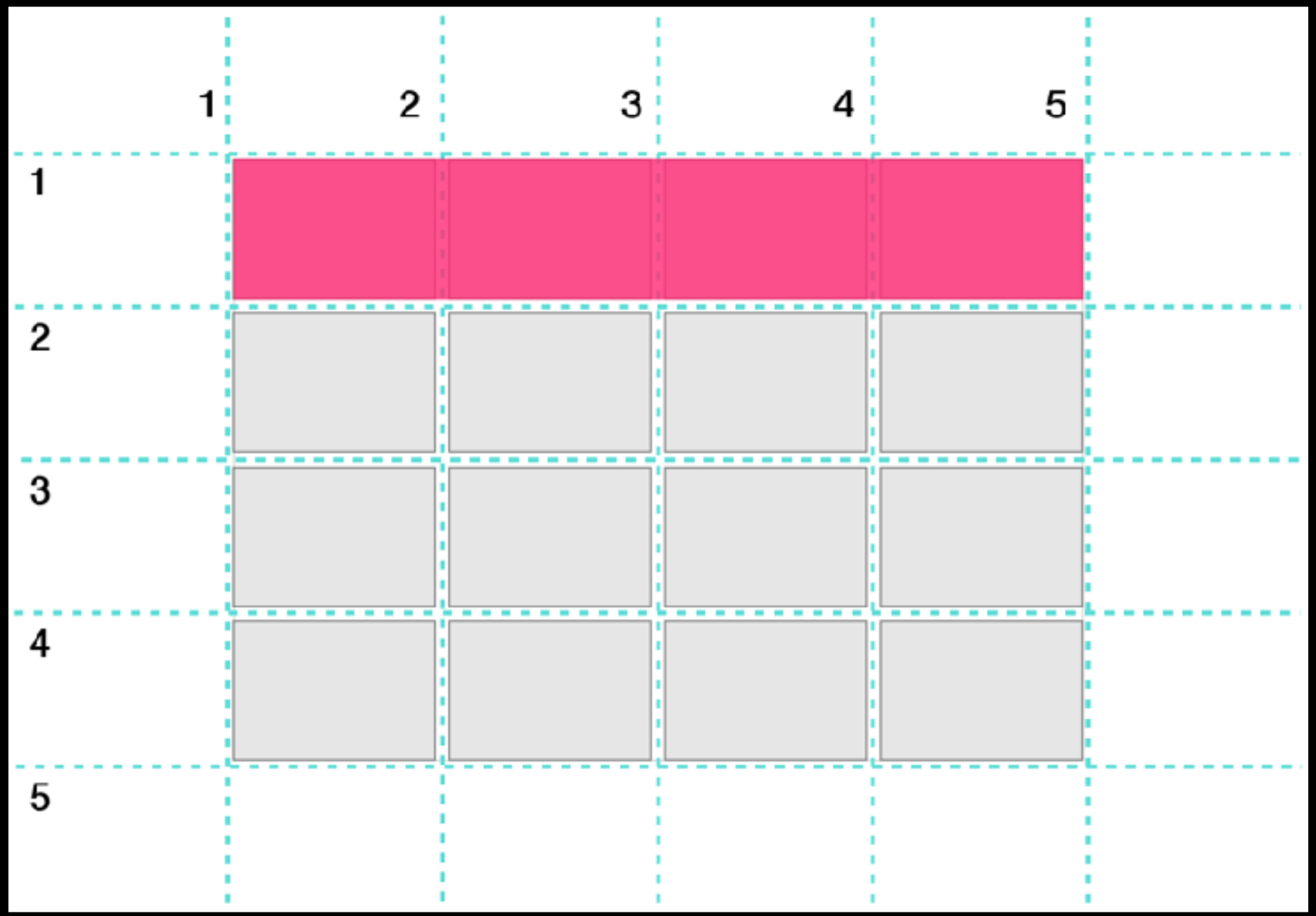
bit.ly/2K2gJSe

Placing items

Placing items by grid line

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

bit.ly/2K2gJSe



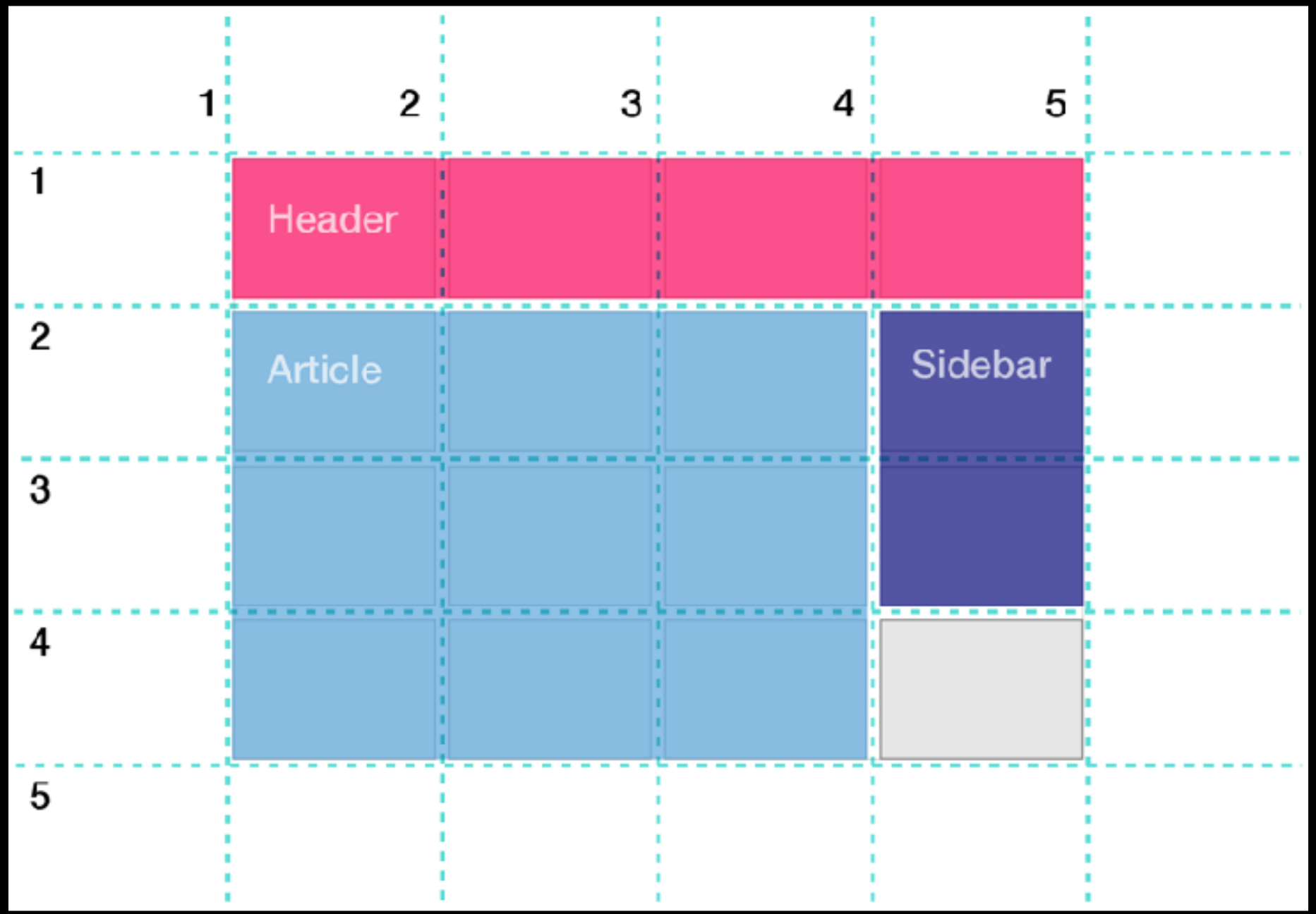
bit.ly/2K2gJSe

Placing items

```
article {
  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;
}
```

bit.ly/2K2gJSe



bit.ly/2K2gJSe

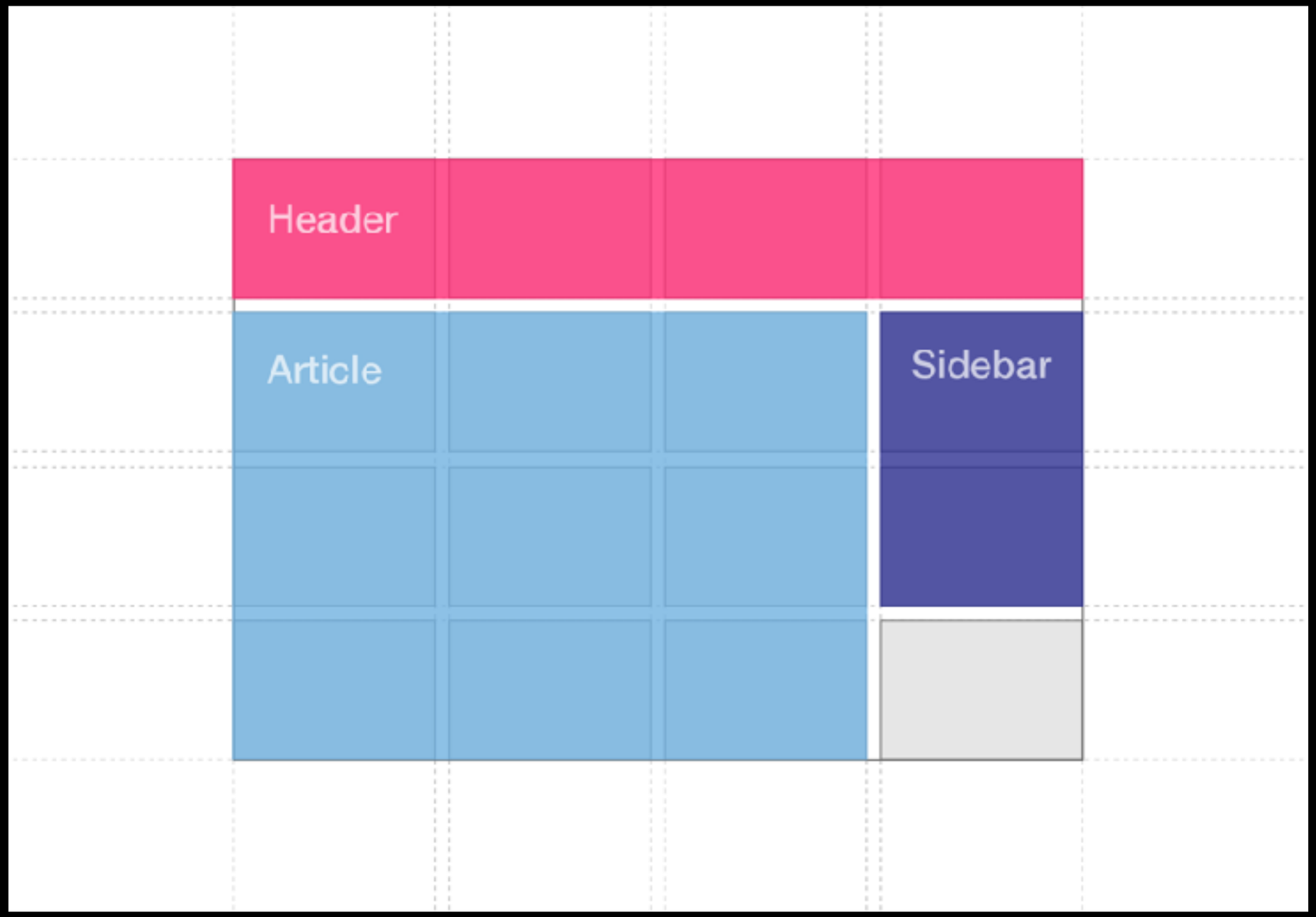
Placing items

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

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

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

bit.ly/2K2gJSe



bit.ly/2K2gJSe

Placing items

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

bit.ly/2K2gJSe

Placing items

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

bit.ly/2K2gJSe

Placing items

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

bit.ly/2K2gJSe

Placing items

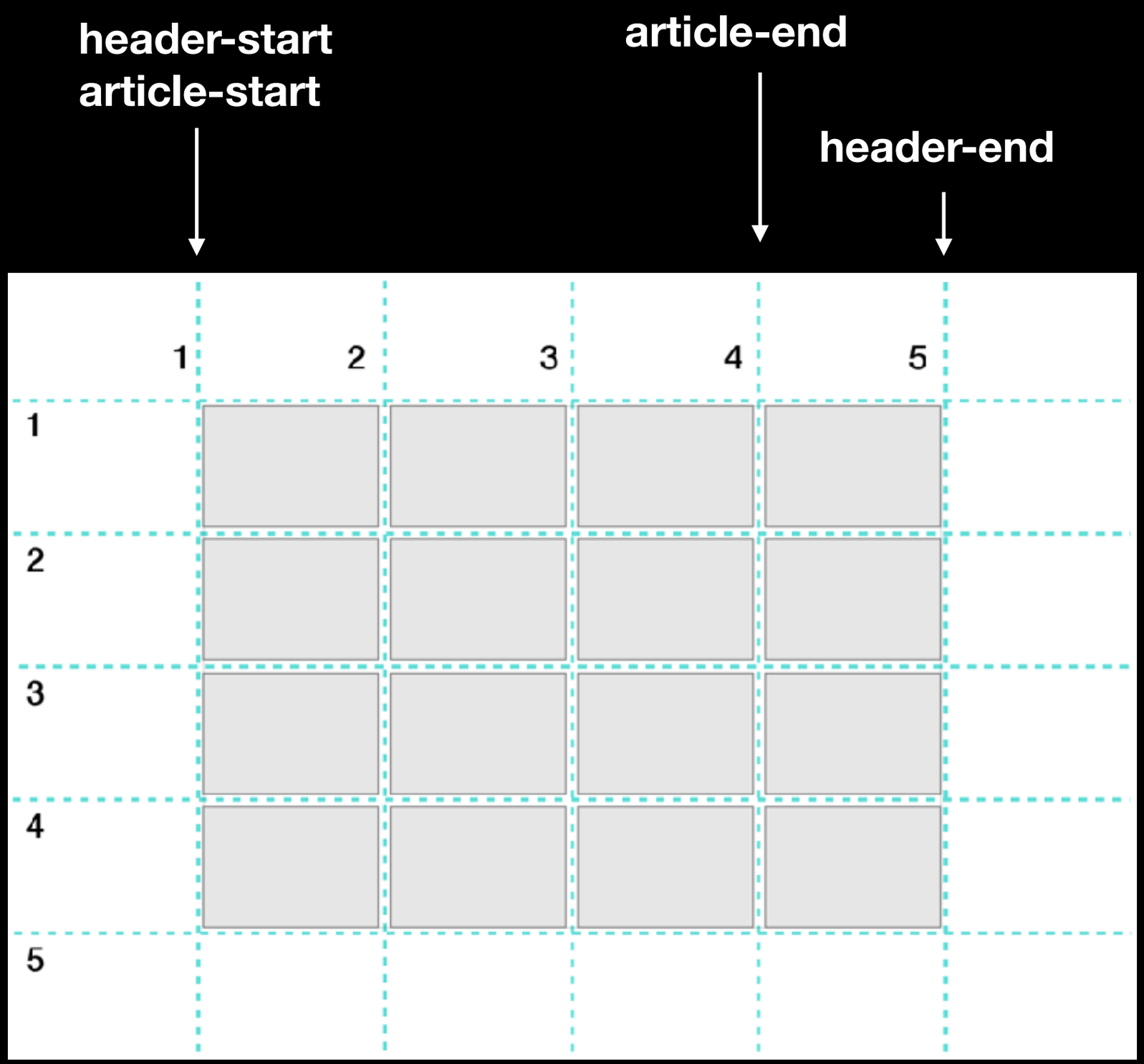
```
article {
  grid-column: span 3 / 4 ;
  grid-row: 2 / 5;
}
```

bit.ly/2K2gJSe

Grid areas

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

bit.ly/2K8ji6p



bit.ly/2K2gJSe

Grid areas

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

article {
  grid-column: article
  grid-row: 2 / 5;
}

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

bit.ly/2K8ji6p

Grid areas

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

bit.ly/2HZMO88

Grid areas

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

article {
  grid-area: article;
}

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

bit.ly/2HZMO88

CSS Variables

(a.k.a. Custom Properties)

CSS Variables

!=

Preprocessor Variables

CSS Variables == Dynamic variables

CSS Variables = let

Preprocessor Variables = const

Declaring 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;
}
```

bit.ly/2tv5NCD

Defaults

```
.box {
  background-color: var(--bgColor, orange);
  max-width: 600px;
  height: 400px;
  margin: 20px;
}

.purple {
  --bgColor: rebeccaPurple;
}
```

bit.ly/2MIlzl3

Defaults

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

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

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

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

bit.ly/2MIlzI3

CSS Variables

+

CSS Grid

CSS Variables + Grid

```
.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;
}
```

CSS Variables + Grid

```css
:root {
  --colWidth: 30px;
  --gutter: 20px;

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

CSS Variables + Grid

```
.grid {
  display: grid;
  grid-template-columns:
    [outer-start]
    minmax(20px, 1fr)
    [wrapper-start]
    repeat(24, minmax(0, var(--columnWidth)))
    [wrapper-end]
    minmax(20px, 1fr)
    [outer-end];
  grid-gap: 0 var(--gutter);
  …
}
```

bit.ly/2Mbmj23

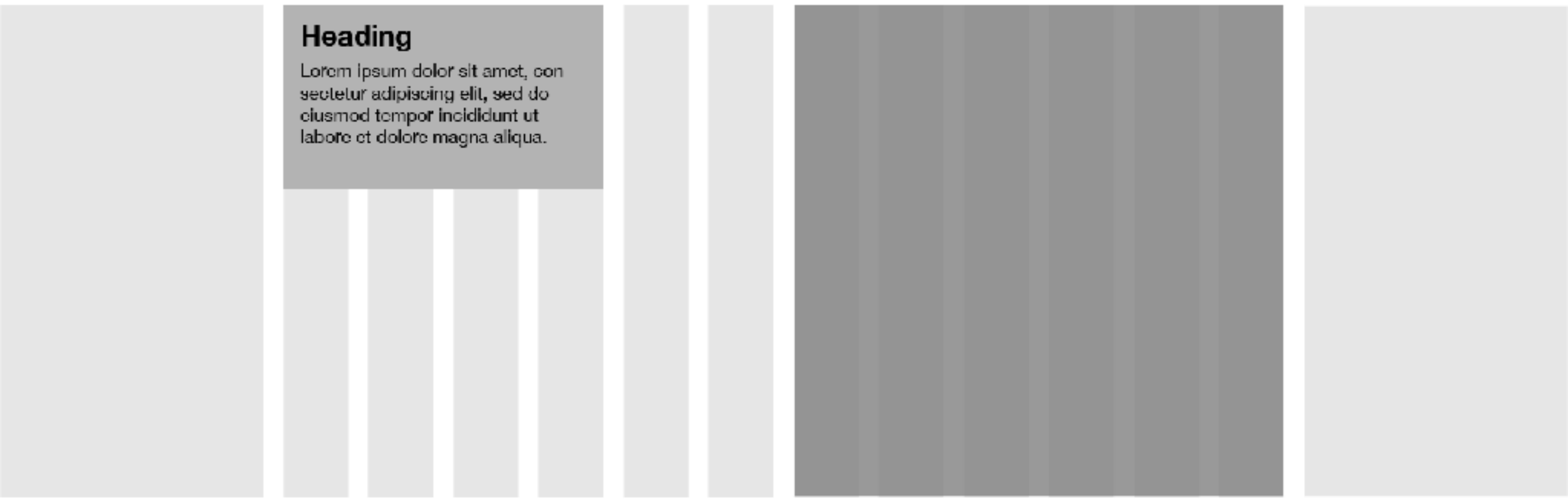
CSS Variables + Grid

```html
<div class="grid">
  <div class="grid__body"></div>
  <div class="grid__media"></div>
</div>
```

```css
.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;
}
```

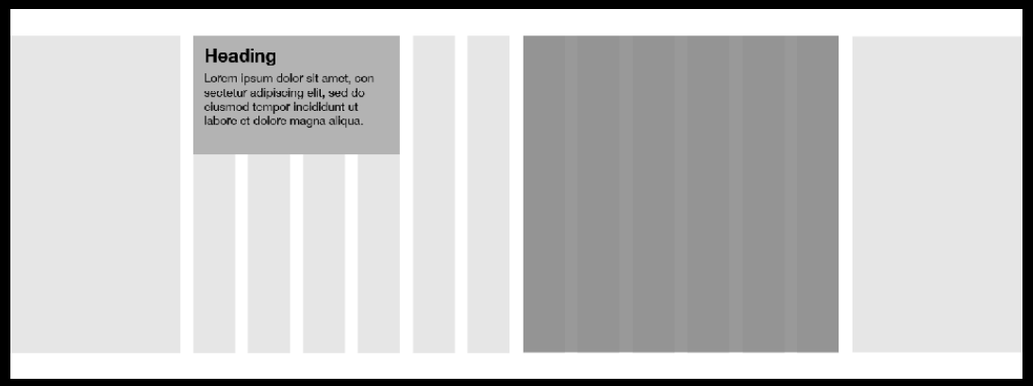
bit.ly/2MNWaI7

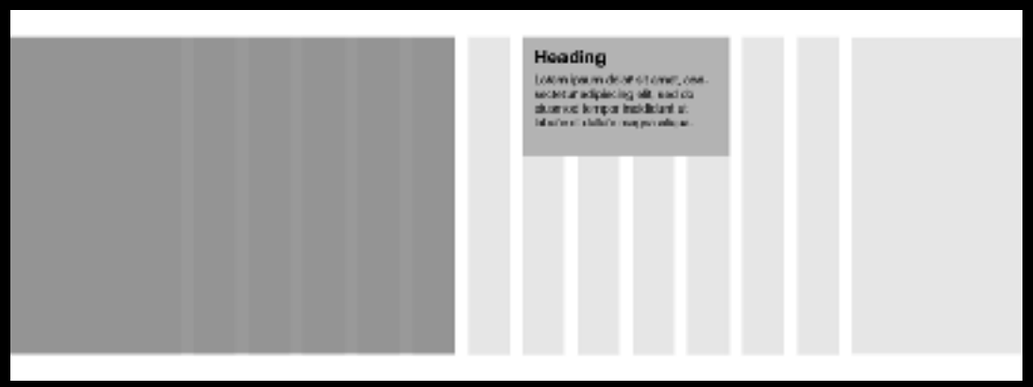
CSS Variables + Grid

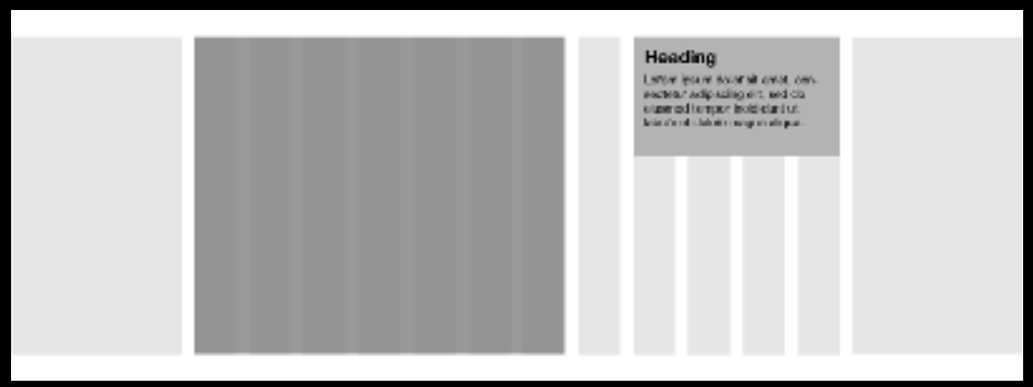


bit.ly/2MNWaI7

CSS Variables + Grid







bit.ly/2MNWaI7

CSS Variables + Grid

```
.grid__body {
  grid-column: var(--bodyStart, wrapper-start) /
span var(--bodySpan, 4);
}

.grid__media {
  grid-column: var(--mediaStart, 7) /
span var(--mediaSpan, 6);
}
```

bit.ly/2MNWaI7

CSS Variables + Grid

```
.grid:nth-child(2) {
  --bodyStart: 9;
  --mediaStart: wrapper-start;
}

.grid:nth-child(3) {
  --bodyStart: 9;
  --mediaStart: start;
}

.grid:nth-child(4) {
  --mediaSpan: 6;
}
```

bit.ly/2MNWaI7

CSS Variables

+

Javascript

CSS Variables + Javascript

Get property value:

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

(Returns a string)

Set property value:

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

Demos

Randomly generated grid:

bit.ly/2ywpq2H

User manipulated grid:

bit.ly/2K08rLF

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

Thank you for listening!

@mbarker_84 / @CSSInRealLife