Building Intelligent Layouts with CSS Grid

@MicheBarks

“Intrinsic Web Design”

Jen Simmons


Why Wait
Ordbo


MUD


moz://a

Why do we need Grid?

2D layout

Why do we need Grid?

Grid systems

“Flexbox is great for taking a bunch of oddly-sized things and returning the most reasonable layout for those things”

Rachel Andrew

Why do we need Grid?

Grid systems

Why do we need Grid?

~~Grid systems~~

Fewer dependencies

Why do we need Grid?

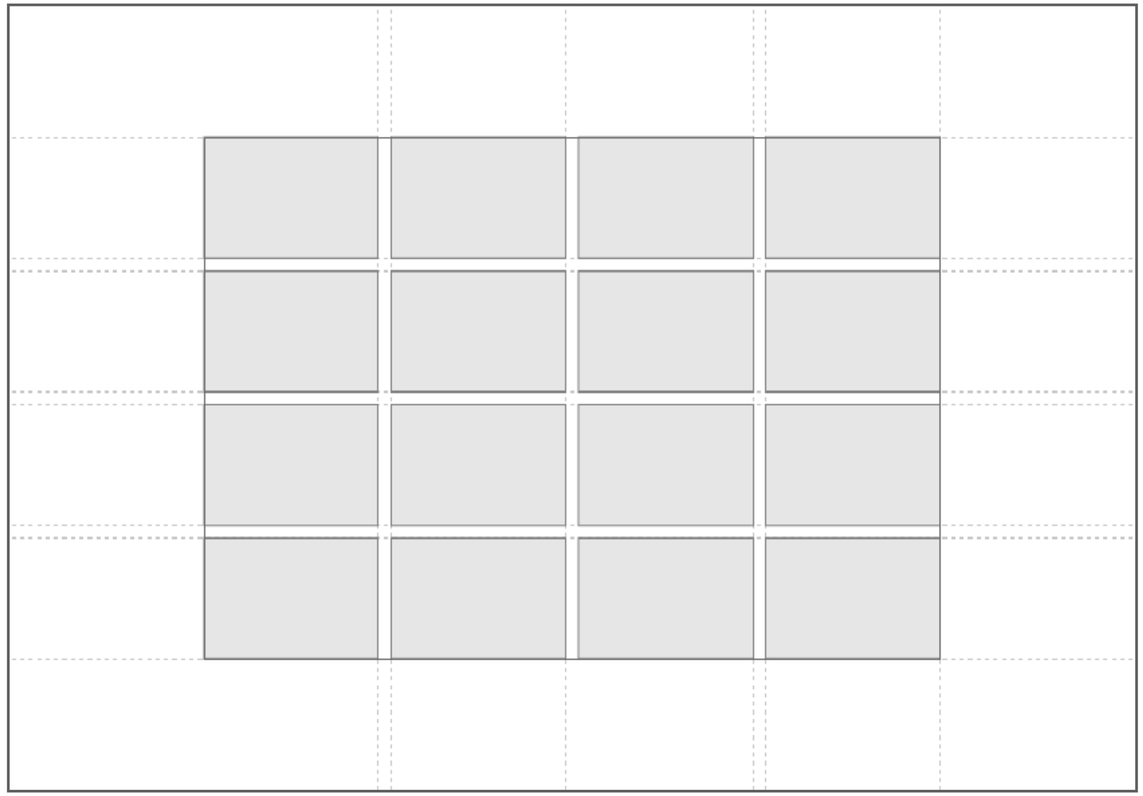
Complete control

Grid terminology

Grid terminology

Grid

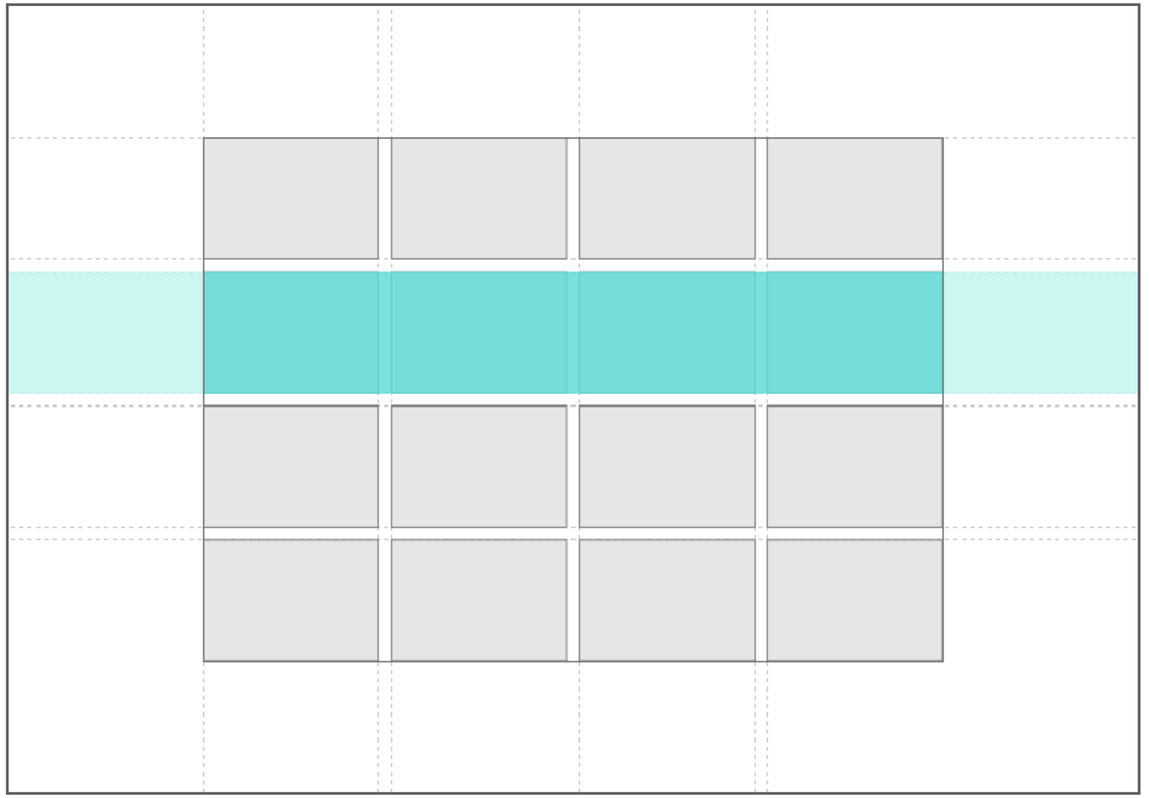
```
display: grid;
```



Grid terminology

Tracks

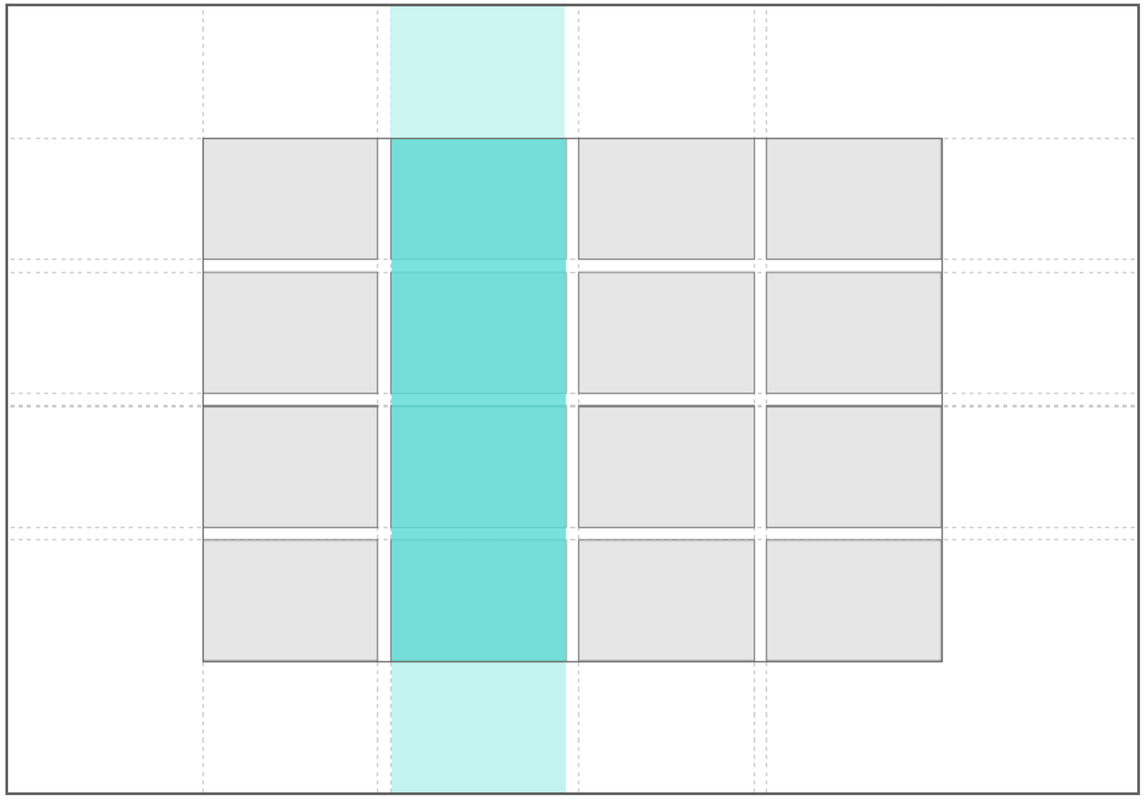
```
grid-template-rows
```



Grid terminology

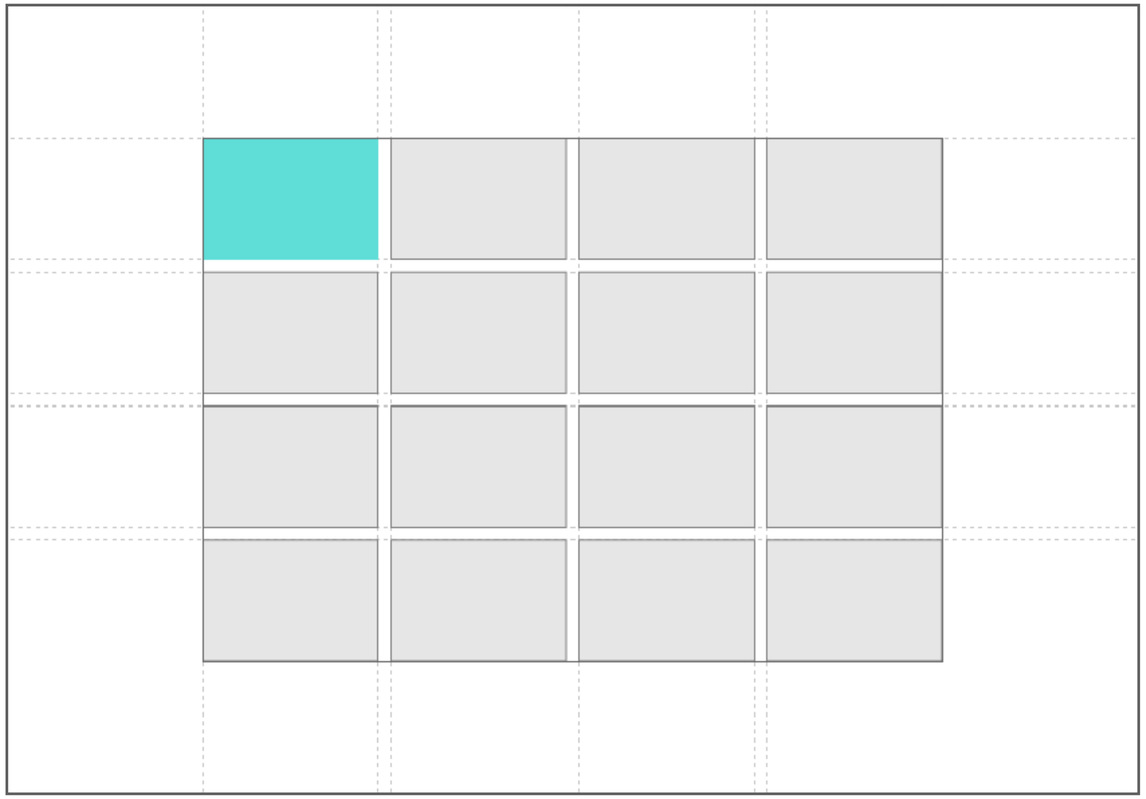
Tracks

```
grid-template-columns
```

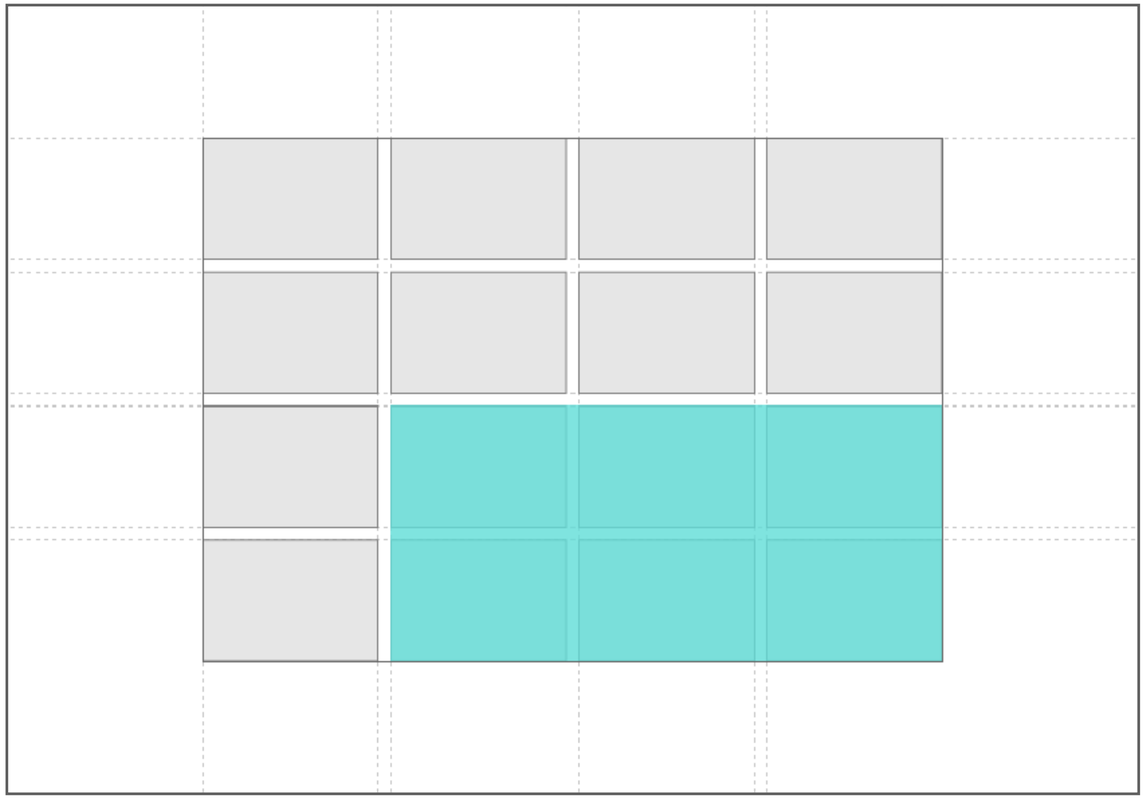


Grid terminology

Cells



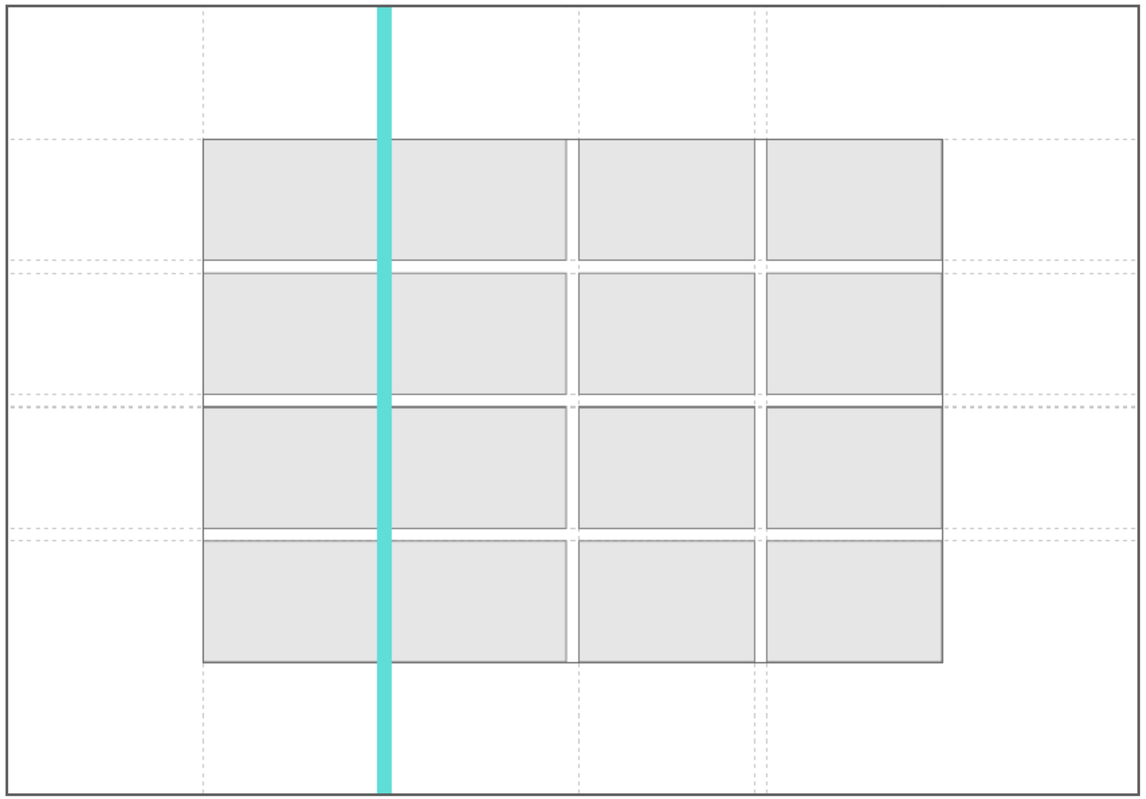
Grid areas



Grid terminology

Gutters

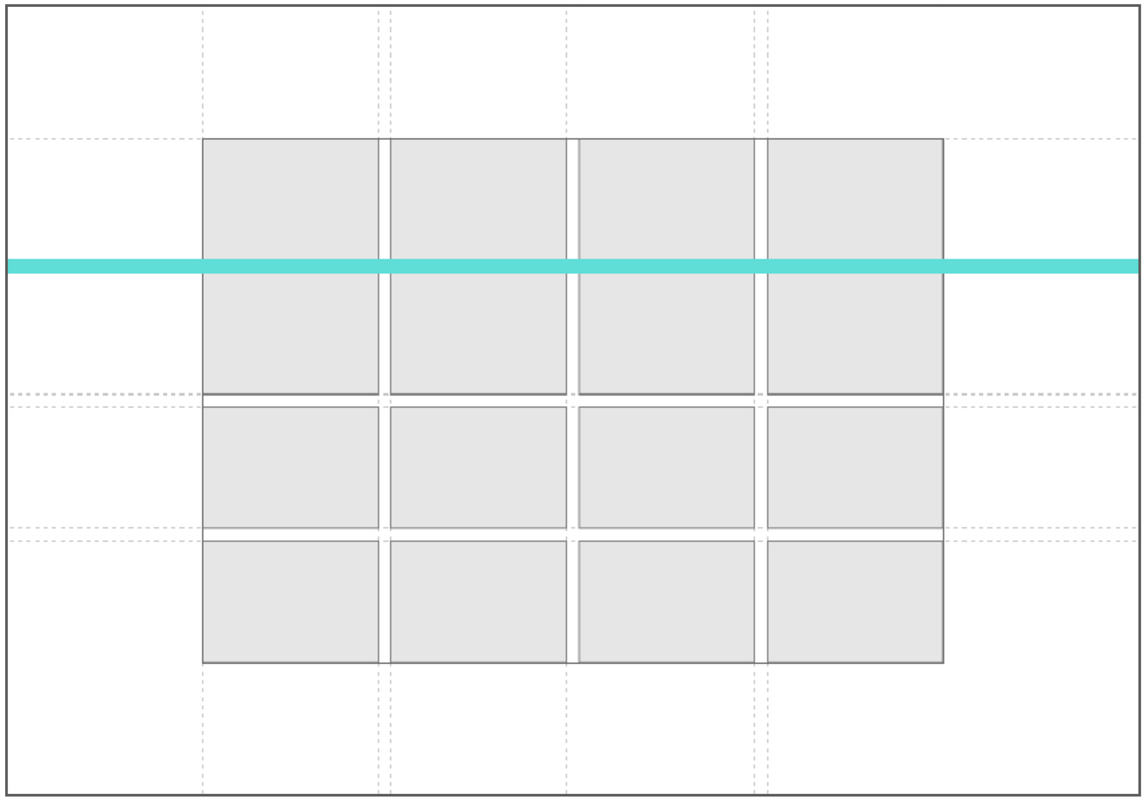
`column-gap`
(Formerly `grid-column-gap`)



Grid terminology

Gutters

`row-gap`
(Formerly `grid-row-gap`)



Defining a grid

CSS

```
.grid {
  display: grid;
}
```

codepen.io/michellebarker/pen/eLZwVg

CSS

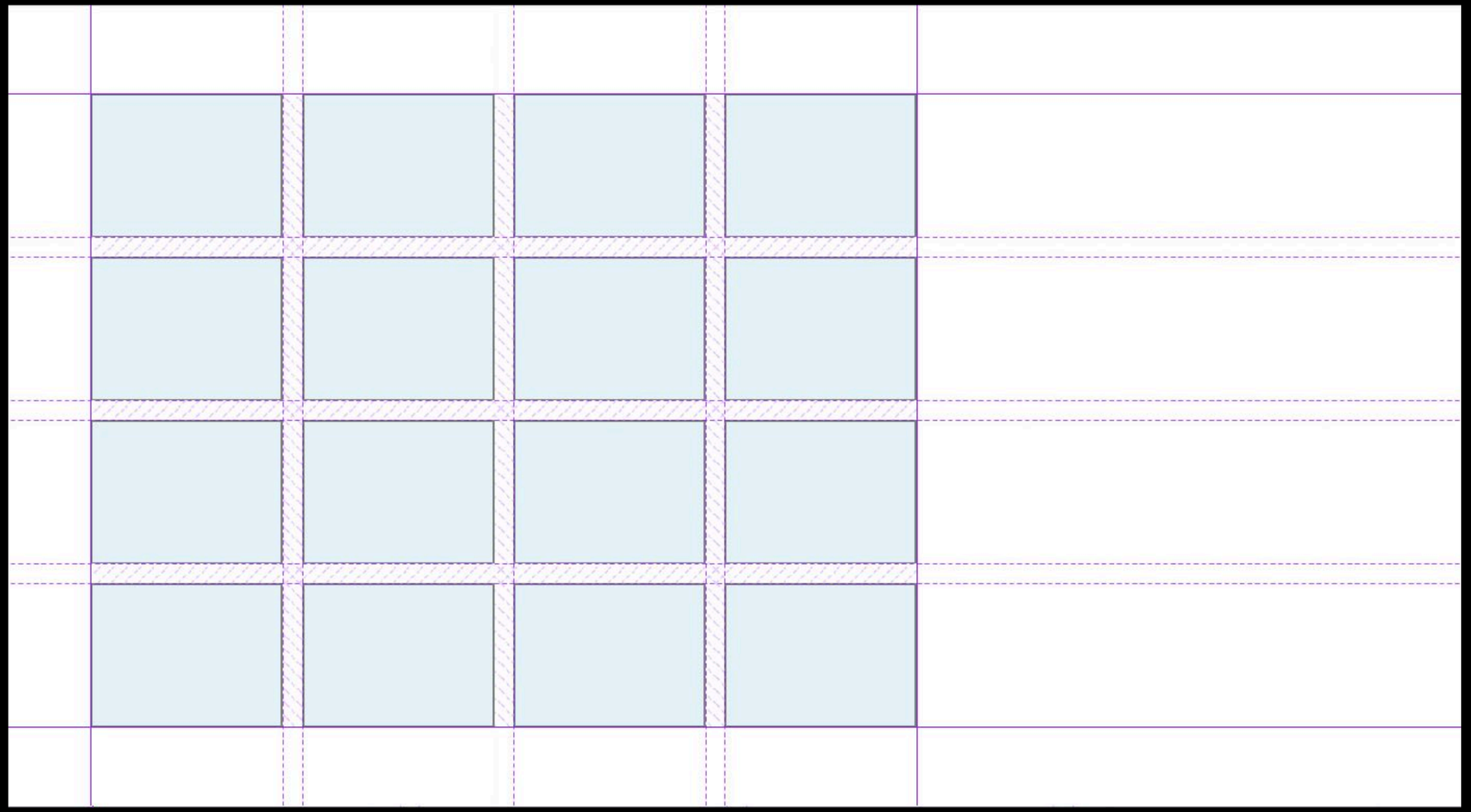
```
.grid {
  display: grid;
  grid-template-columns: 200px 200px 200px 200px;
  grid-template-rows: 150px 150px 150px 150px;
}
```

codepen.io/michellebarker/pen/eLZwVg

CSS

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

codepen.io/michellebarker/pen/eLZwVg



codepen.io/michellebarker/pen/eLZwVg

CSS

repeat()

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

codepen.io/michellebarker/pen/eLZwVg

CSS

Shorthand

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

codepen.io/michellebarker/pen/eLZwVg

CSS

Shorthand

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

codepen.io/michellebarker/pen/eLZwVg

CSS

Fraction units (fr)

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

codepen.io/michellebarker/pen/eLZwVg

Fr = fraction units

a fraction of the leftover space in the grid container

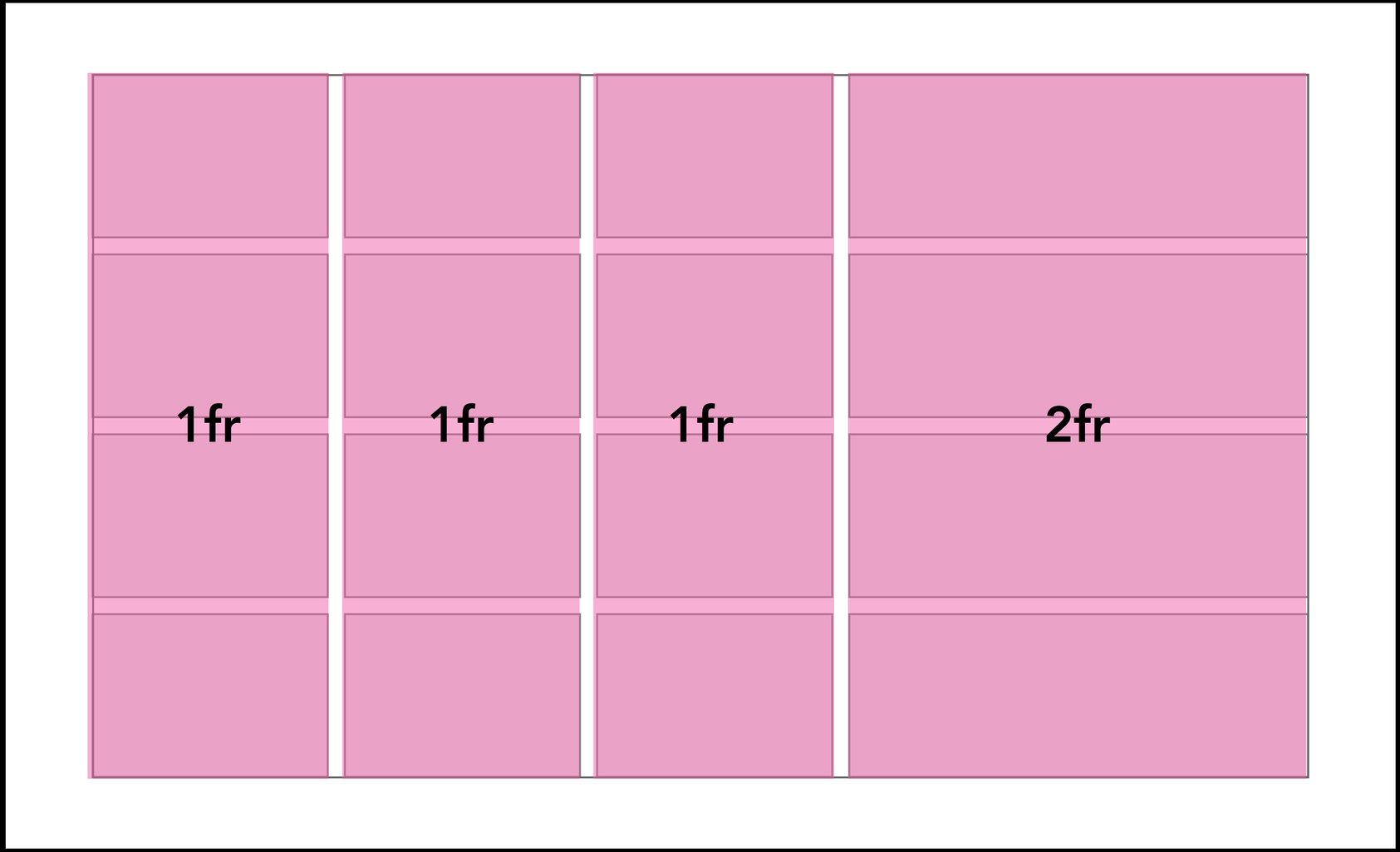
CSS

Fraction units (fr)

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

codepen.io/michellebarker/pen/eLZwVg

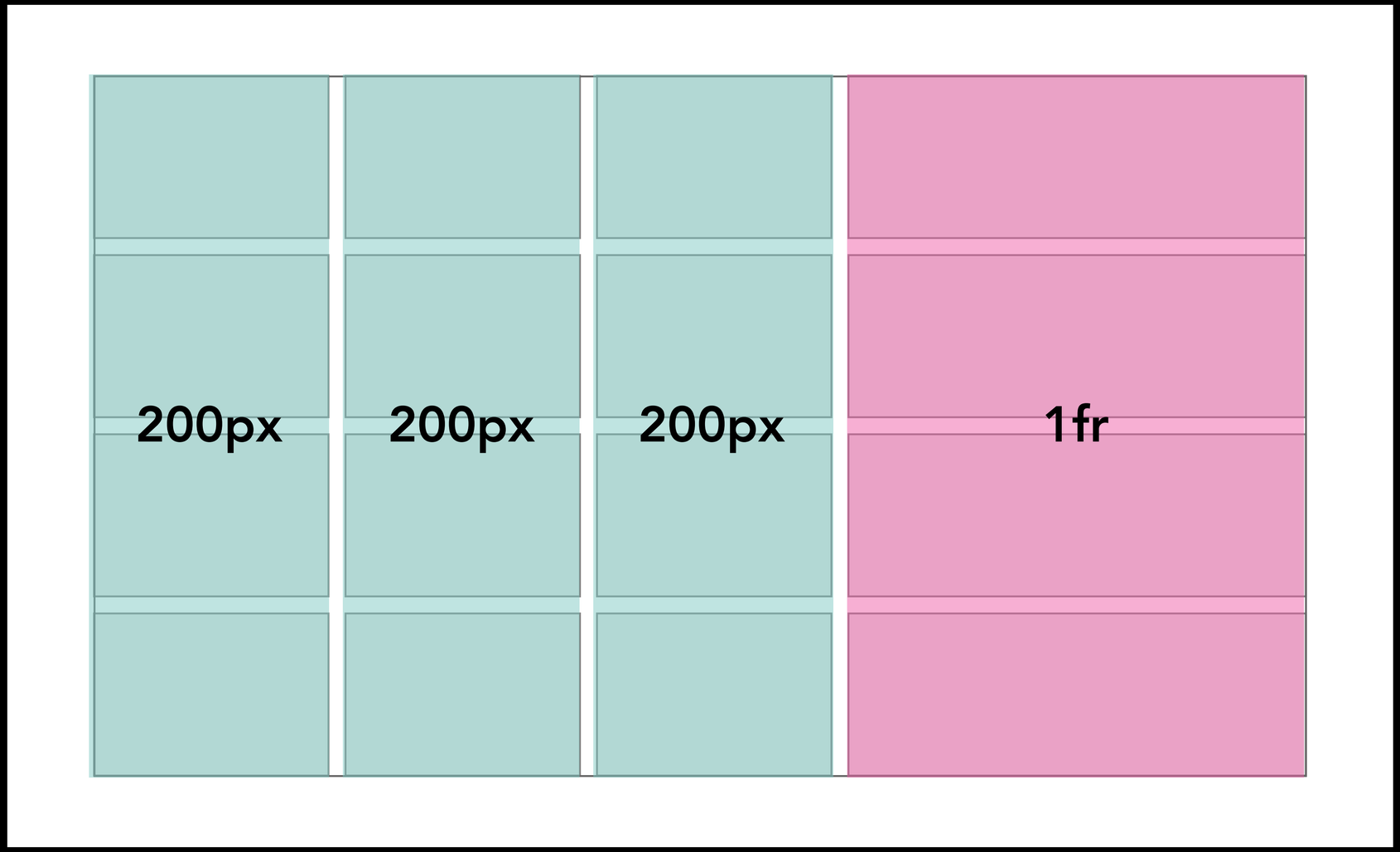
```
grid-template-columns: repeat(3, 1fr) 2fr;
```



CSS

Fraction units (fr)

```
.grid {
  display: grid;
  grid-template-columns: repeat(3, 200px) 1fr;
  grid-template-rows: repeat(4, 150px);
  gap: 20px;
}
```


200px
200px
200px
1fr

Implicit vs Explicit tracks

CSS

Implicit tracks

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

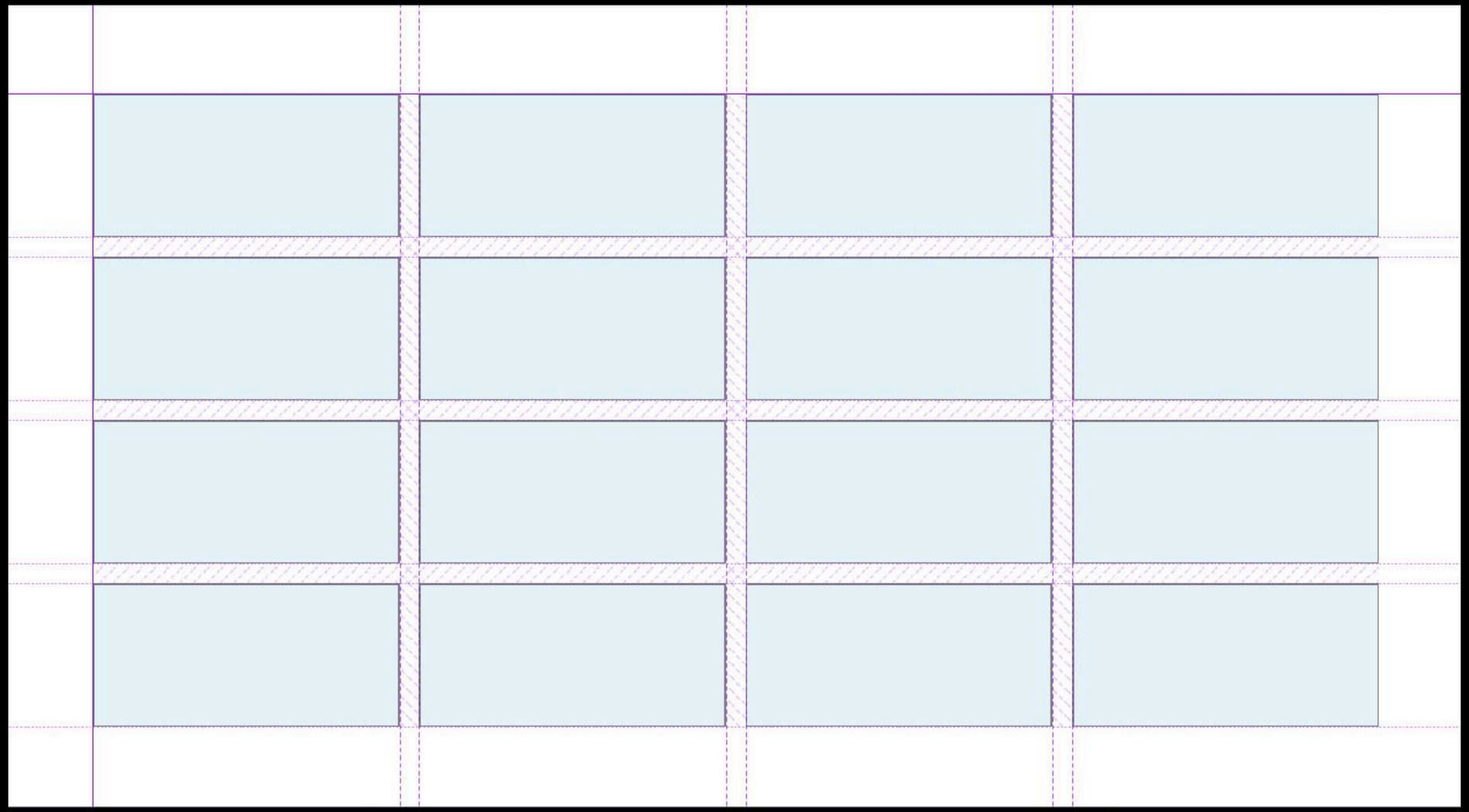
codepen.io/michellebarker/pen/eLZwVg

CSS

Implicit tracks

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

codepen.io/michellebarker/pen/eLZwVg



codepen.io/michellebarker/pen/eLZwVg

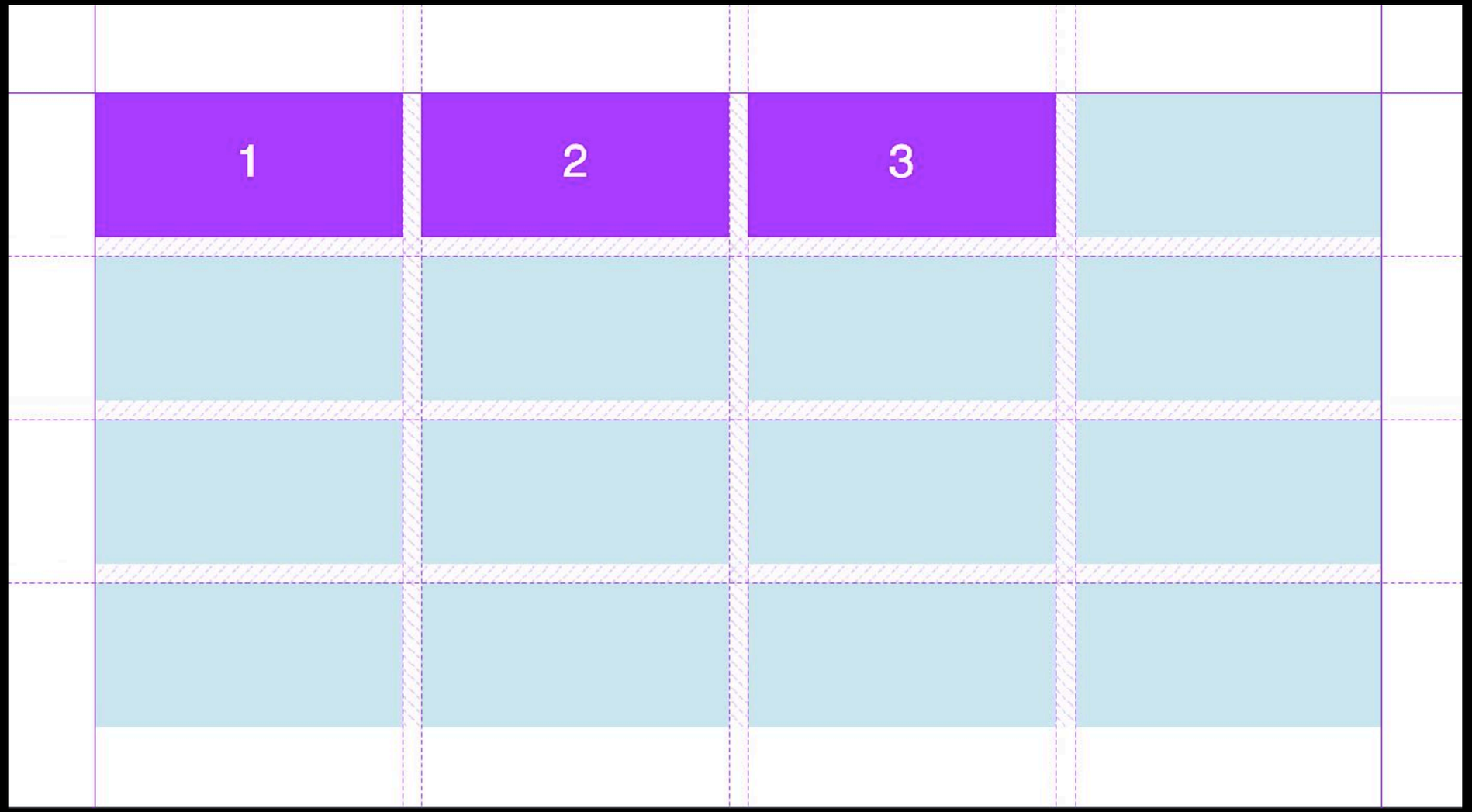
Placing items

HTML

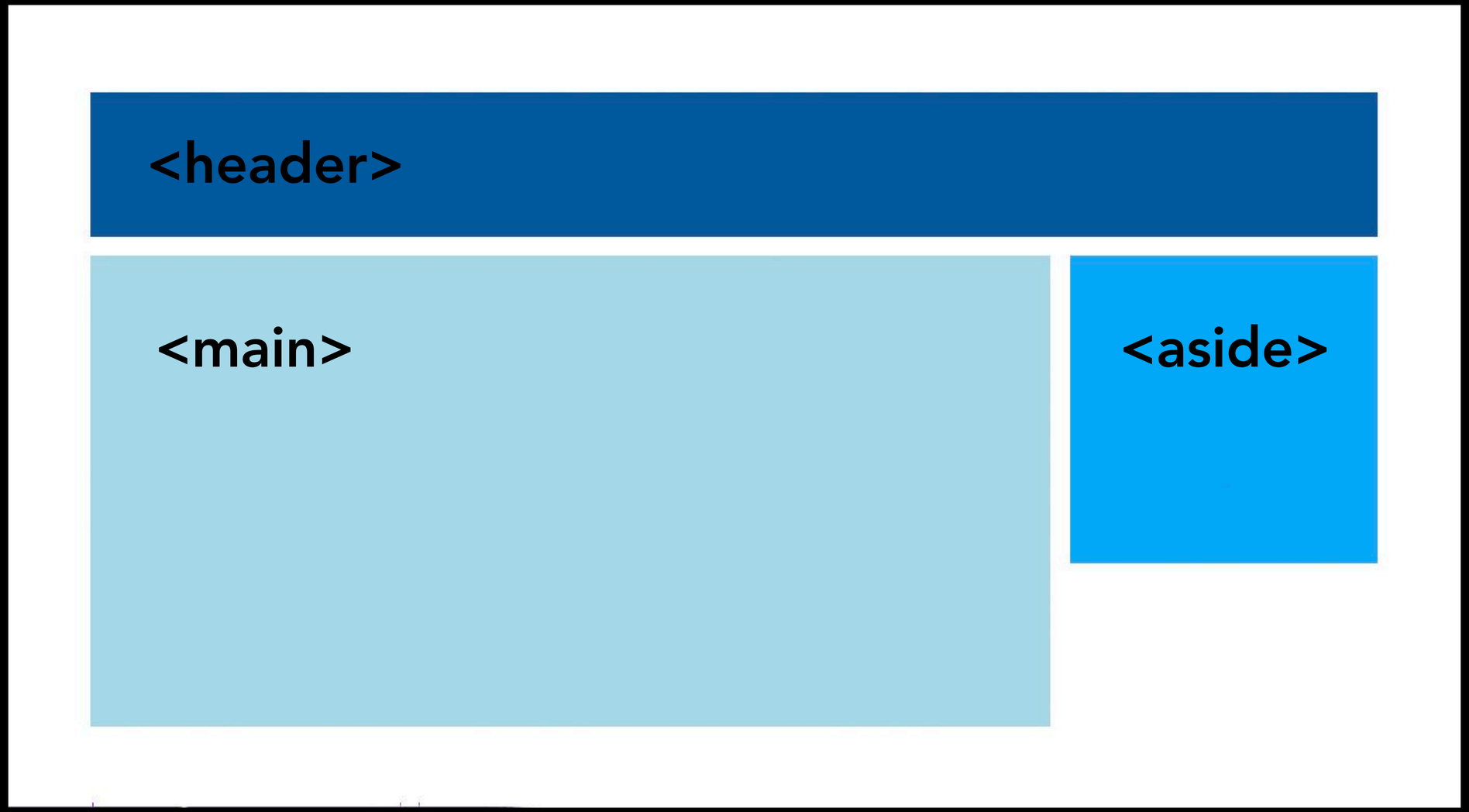
Auto-placement

```
<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



codepen.io/michellebarker/pen/gdrVXP

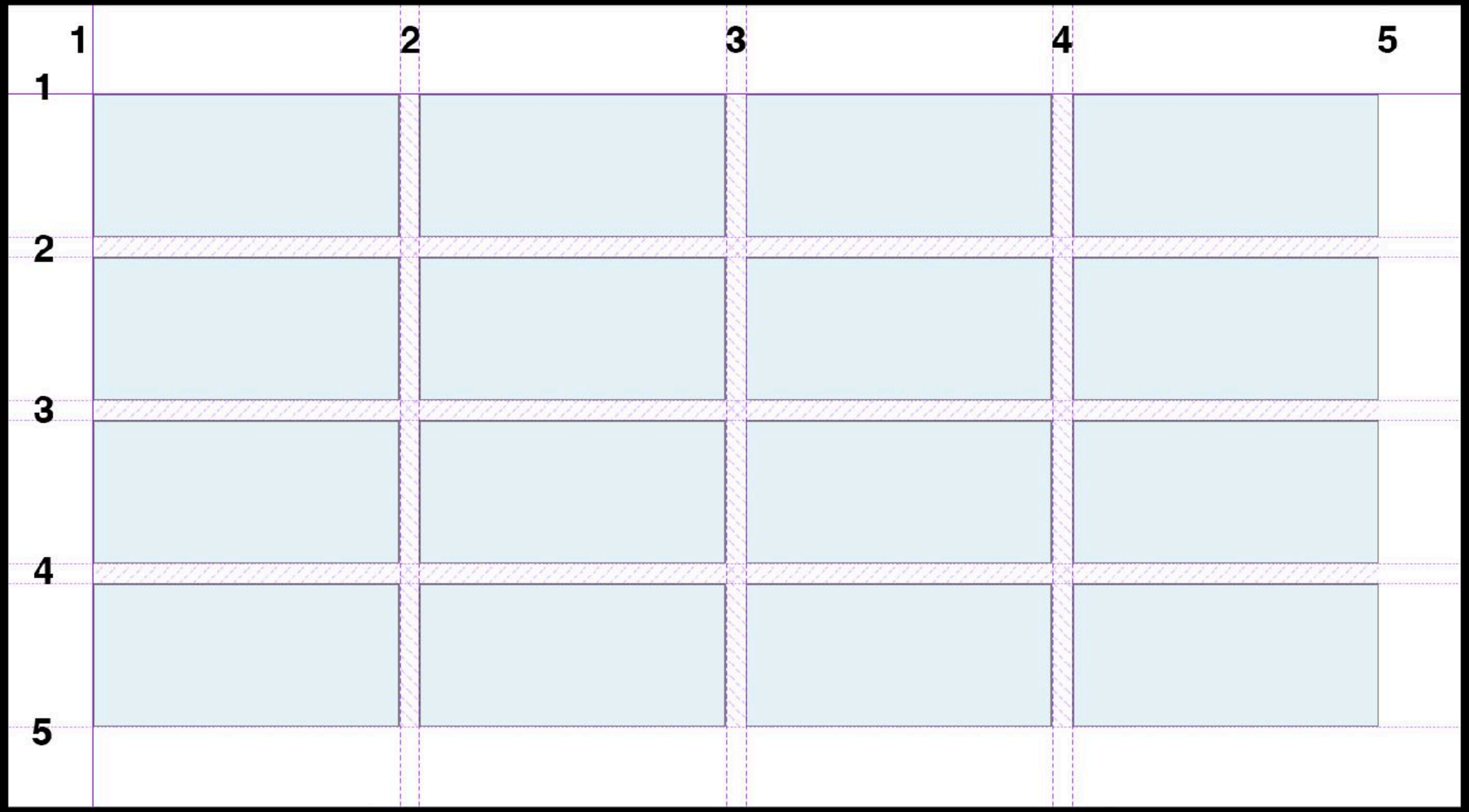


codepen.io/michellebarker/pen/YOqmjP

HTML

Placing items by grid line

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



codepen.io/michellebarker/pen/YOqmjP

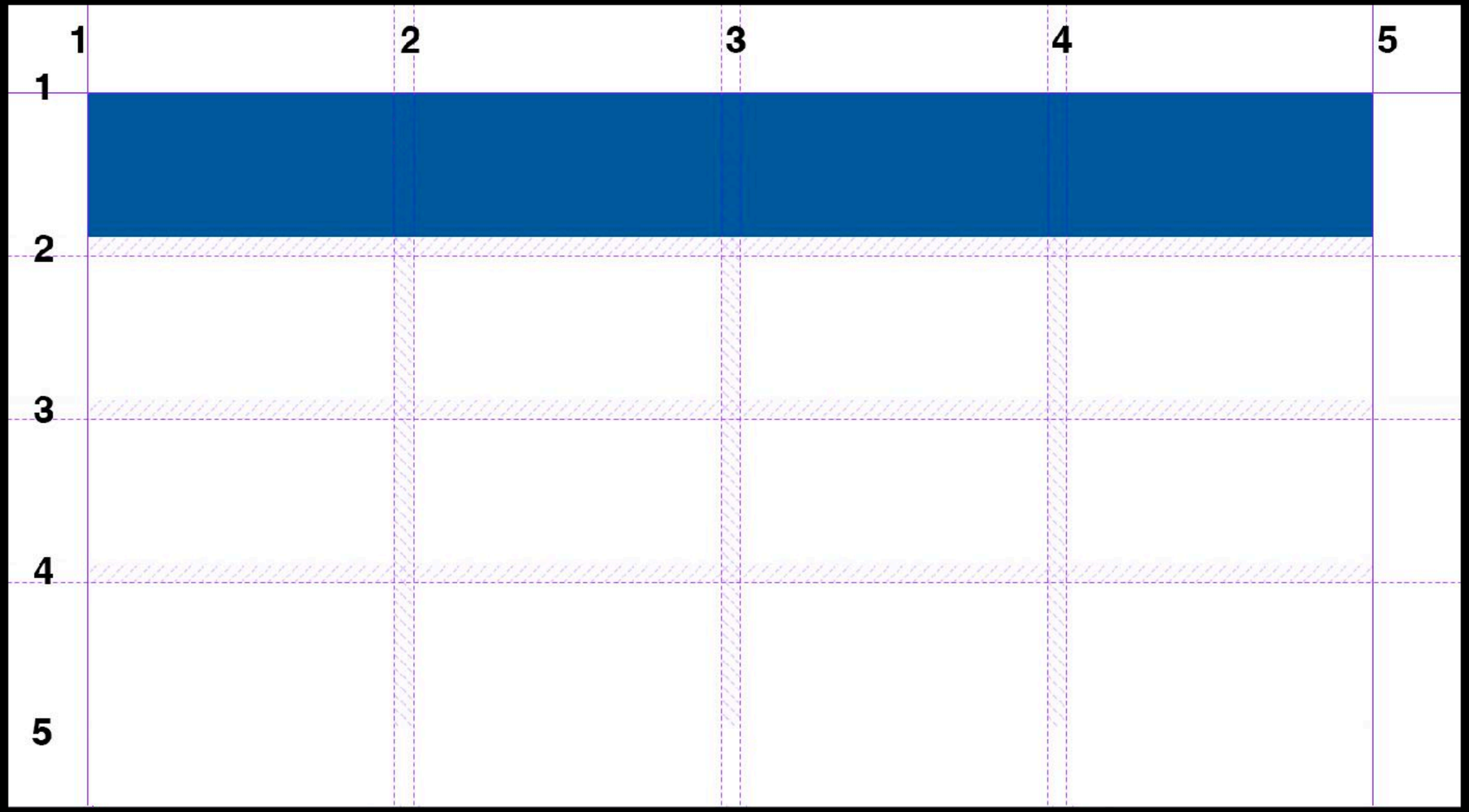
@MicheBarks

CSS

Placing items by grid line

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

codepen.io/michellebarker/pen/YOqmjP



codepen.io/michellebarker/pen/YOqmjP

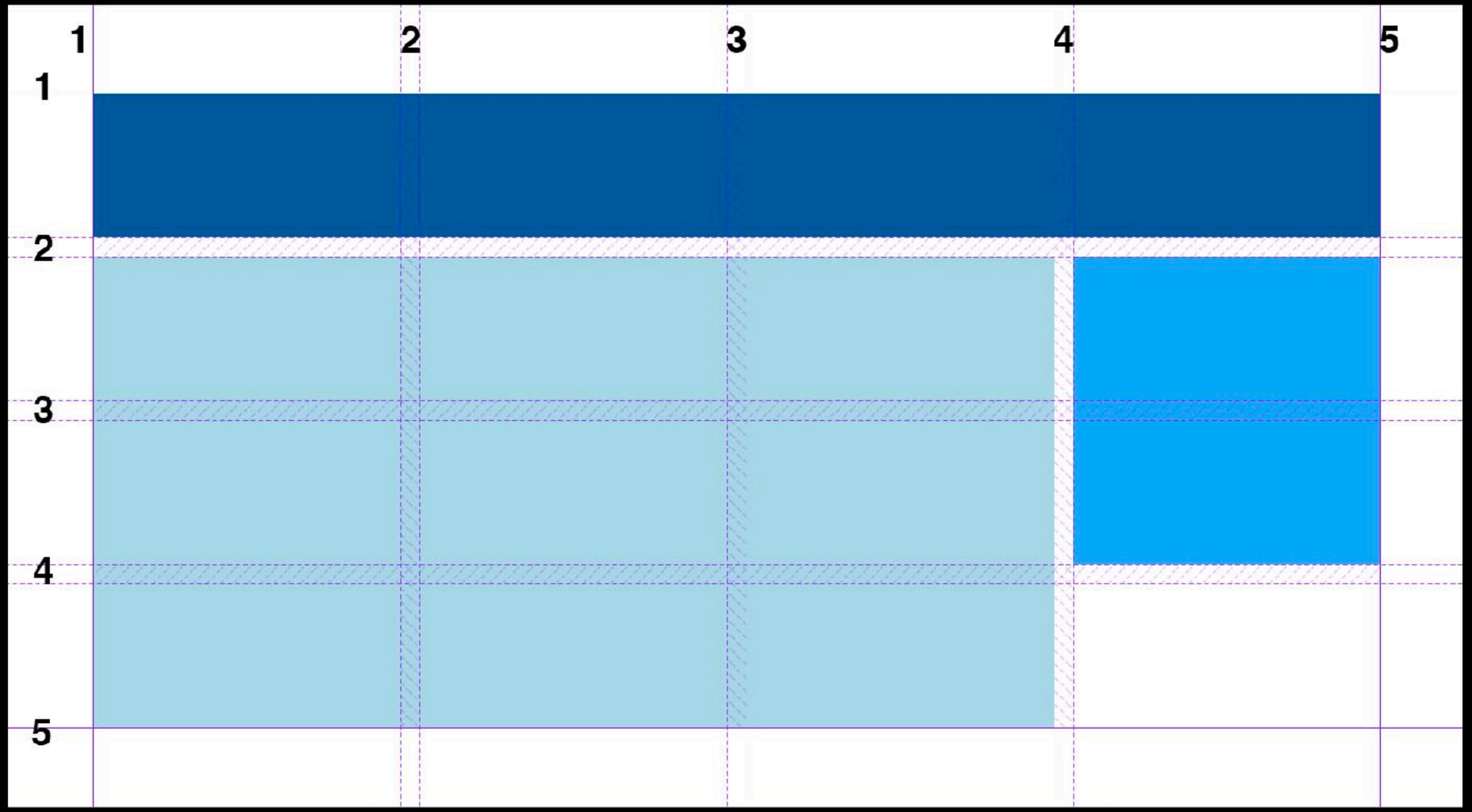
@MicheBarks

CSS

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;
}
```

codepen.io/michellebarker/pen/YOqmjP



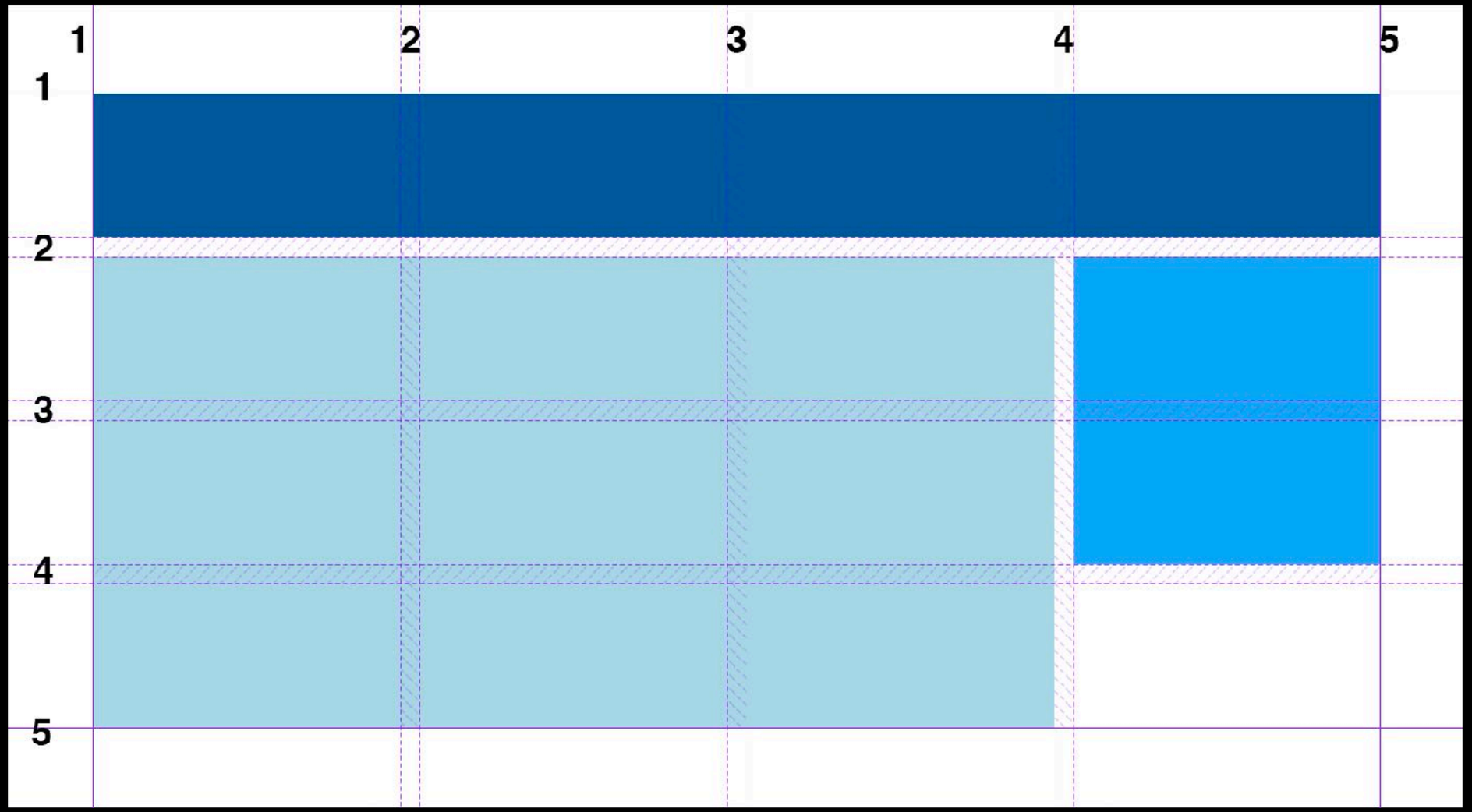
codepen.io/michellebarker/pen/YOqmjP

@MicheBarks

CSS

```
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;
}
```

codepen.io/michellebarker/pen/YOqmjP



codepen.io/michellebarker/pen/YOqmjP

@MicheBarks

CSS

Start line / end line

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

codepen.io/michellebarker/pen/YOqmjP

CSS

Span (with auto-placement)

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

codepen.io/michellebarker/pen/YOqmjP

CSS

Start line / span

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

codepen.io/michellebarker/pen/YOQLLZ

CSS

Span / end line

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

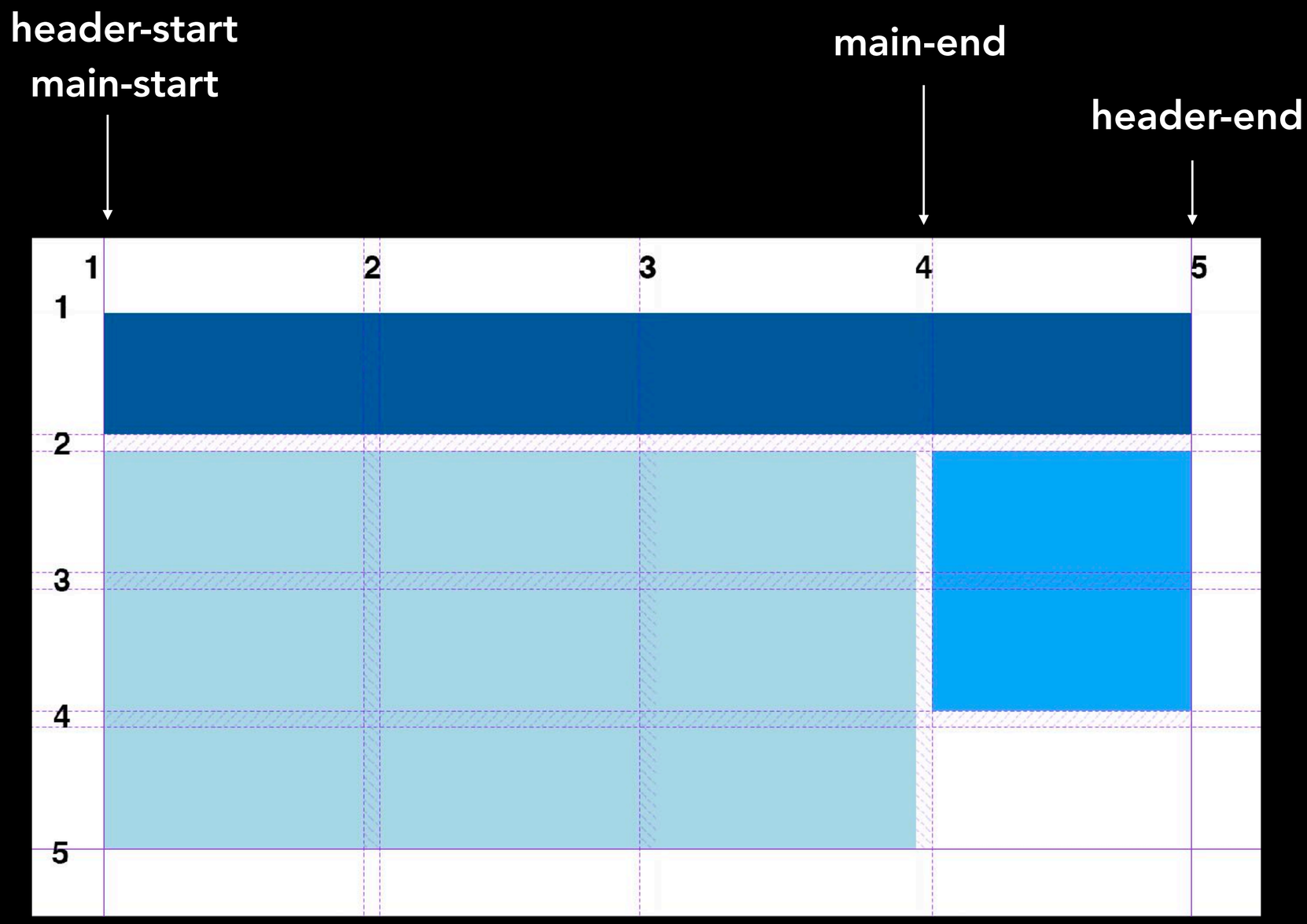
codepen.io/michellebarker/pen/YOqmjP

CSS

Naming grid lines

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

codepen.io/michellebarker/pen/OobJaa



codepen.io/michellebarker/pen/OobJaa

CSS

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

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

codepen.io/michellebarker/pen/rqZevv

CSS

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 .”;
  gap: 20px;
}
```

CSS

grid-template-areas

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

main {
  grid-area: m;
}

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

Building a Complex Component

A Case Study

@MicheBarks

We combine the most advanced technology with uniquely experienced talent, providing a bespoke service and a truly exceptional post-production experience every time.

→

SERVICES

FACILITIES

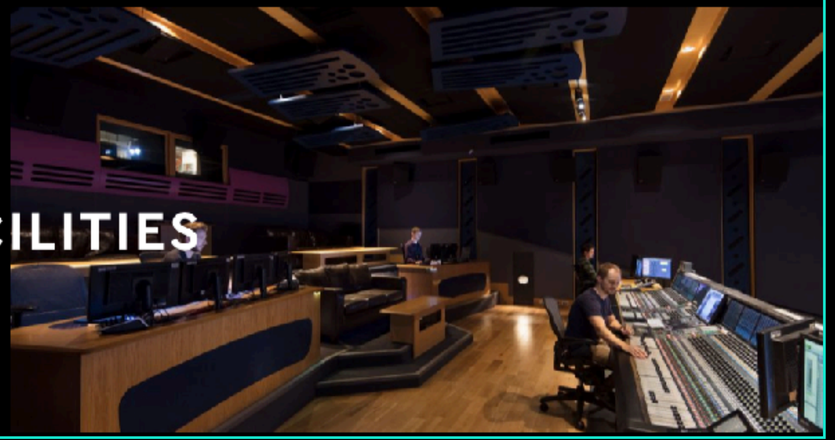
Built to satisfy our clients' every need, our wealth of facilities includes ADR recording, multiple re-recording stages, 4K screening room, and 50+ cutting rooms and production offices.

SERVICES

We combine the most advanced technology with uniquely experienced talent, providing a bespoke service and a truly exceptional post-production experience every time.

→

FACILITIES



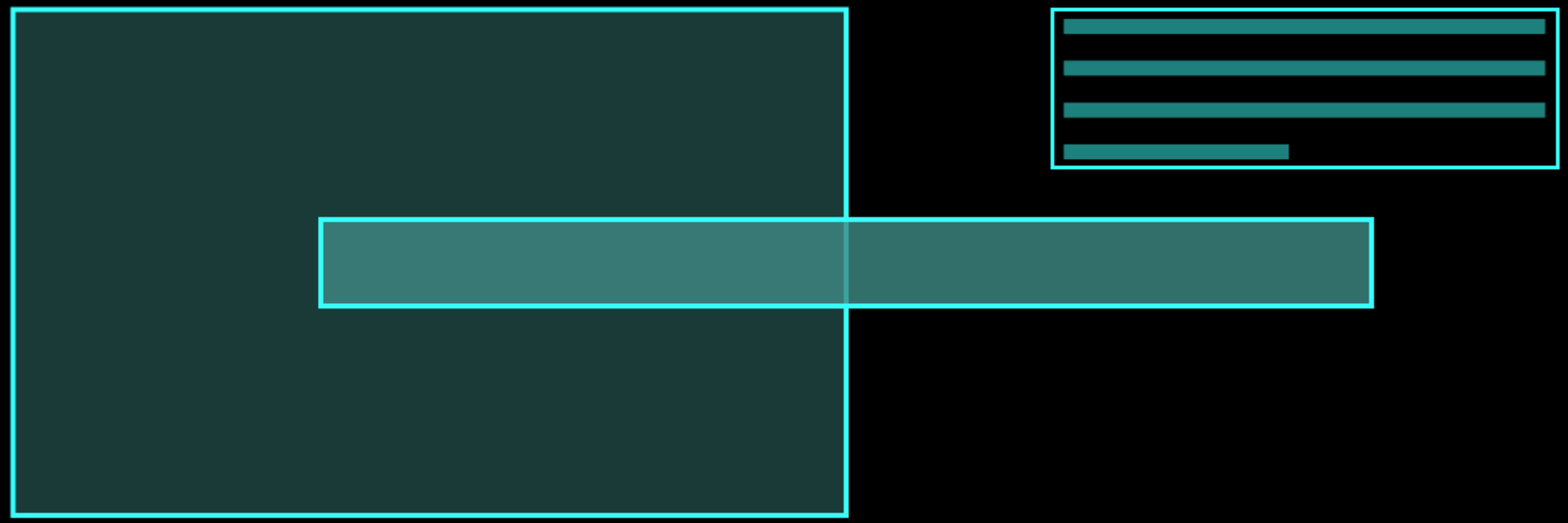
Built to satisfy our clients' every need, our wealth of facilities includes ADR recording, multiple re-recording stages, 4K screening room, and 50+ cutting rooms and production offices.

→

SERVICES

We combine the most advanced technology with uniquely experienced talent, providing a bespoke service and a truly exceptional post-production experience every time.

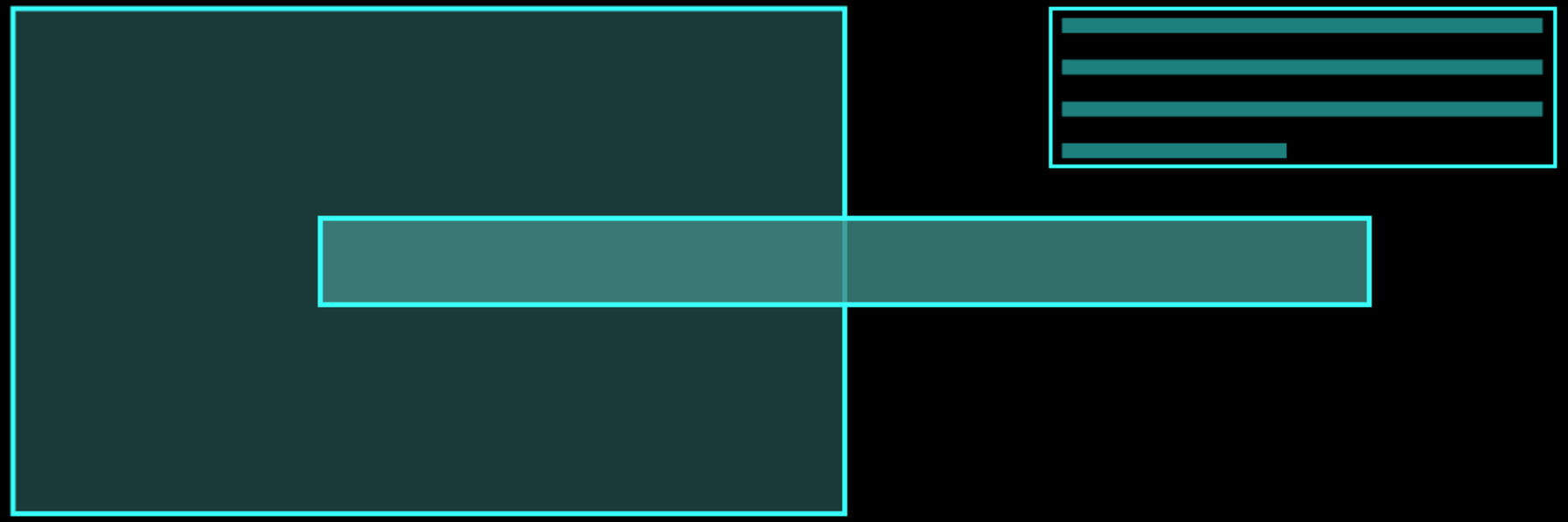
→



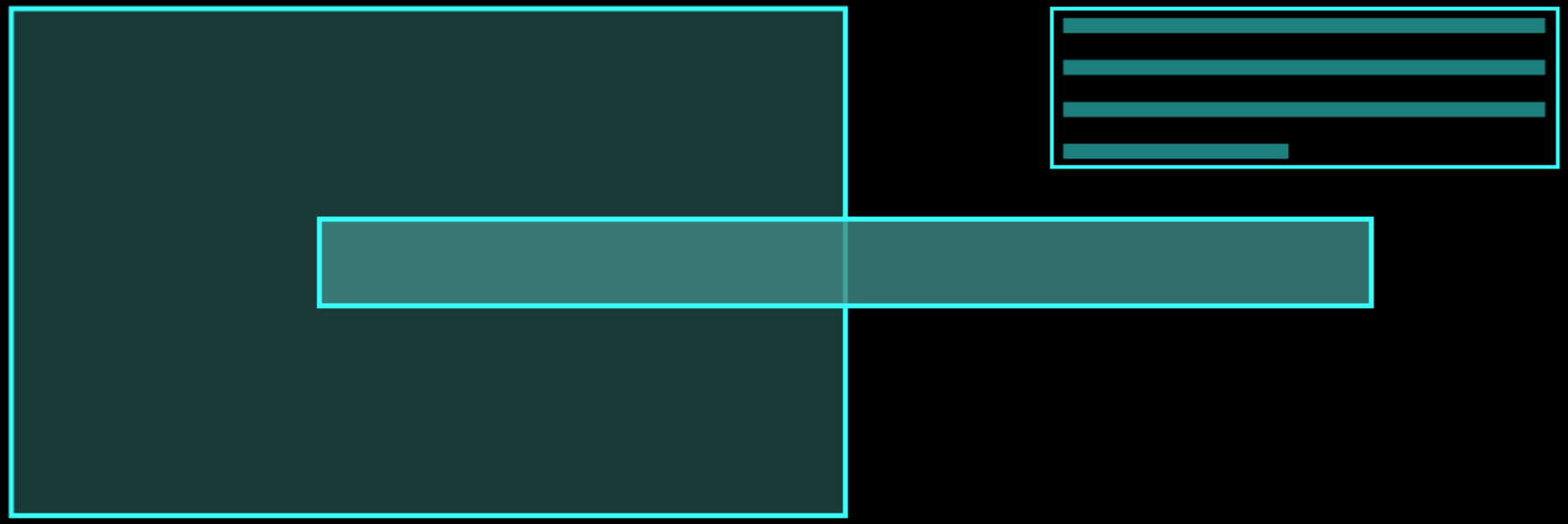
HTML

```
<div class="grid">
  <h2 class="grid__heading"></h2>
  <figure class="grid__image"></figure>
  <div class="grid__body"></div>
</div>
```

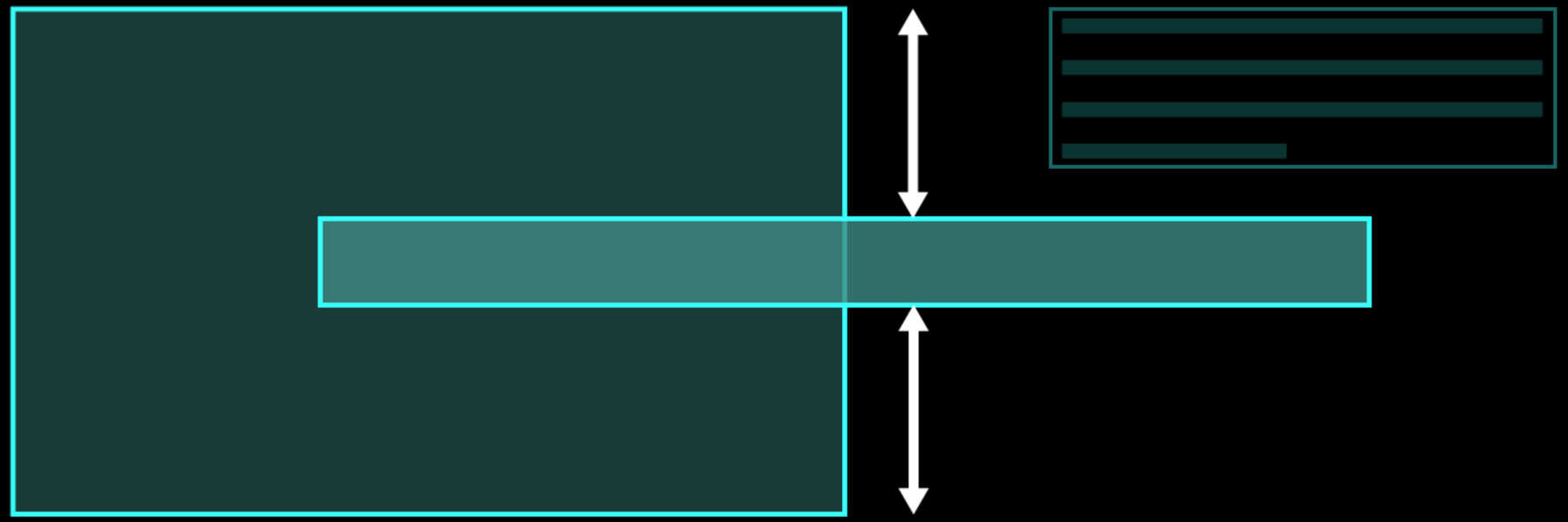
Some rules



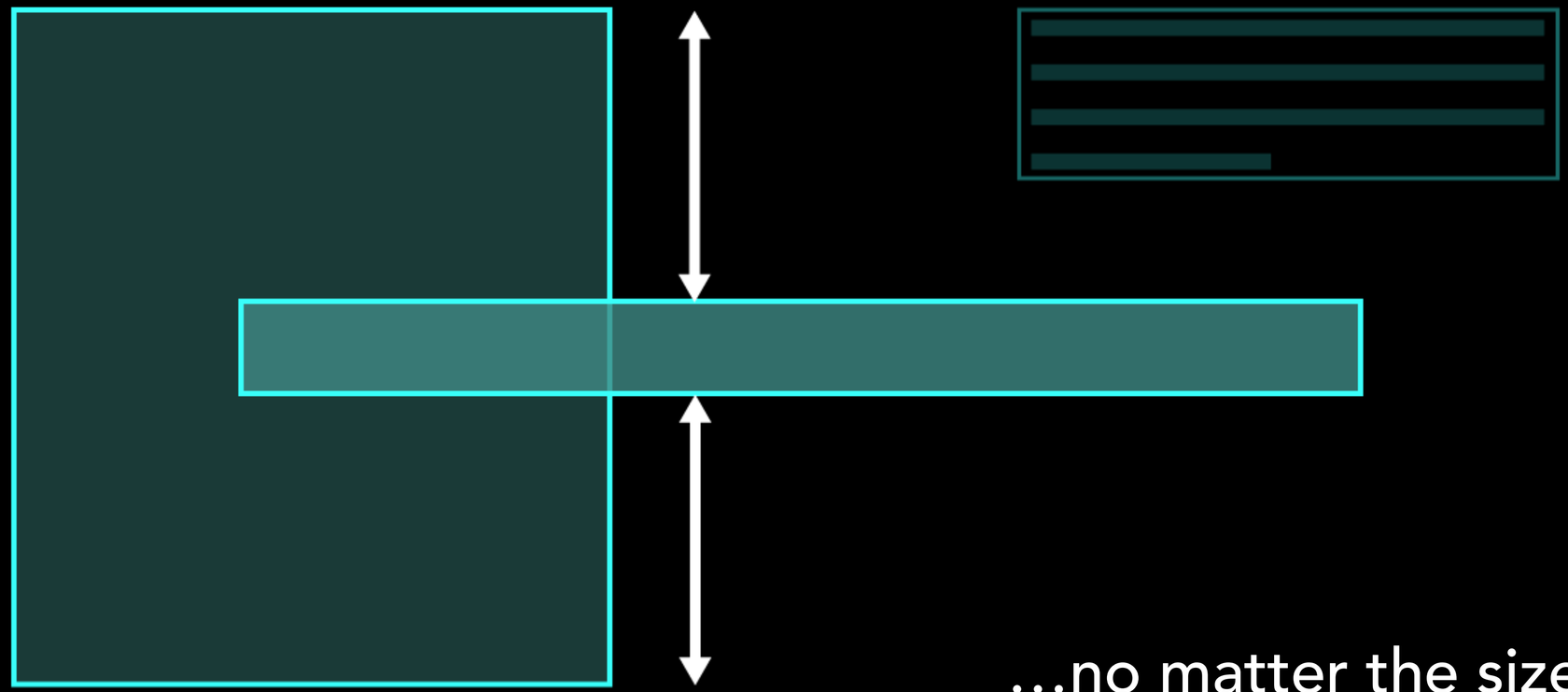
1. Dynamic content



2. The heading must always be vertically centred over the image

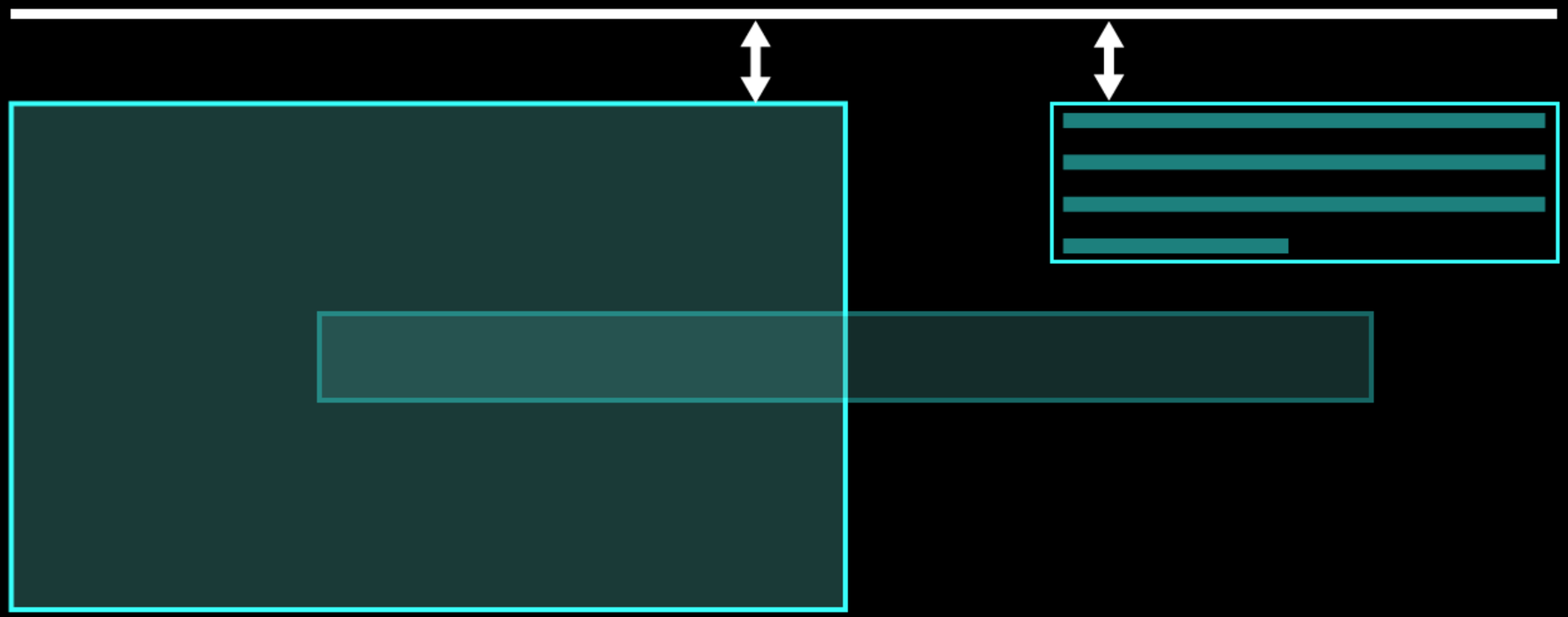


2. The heading must always be vertically centred over the image

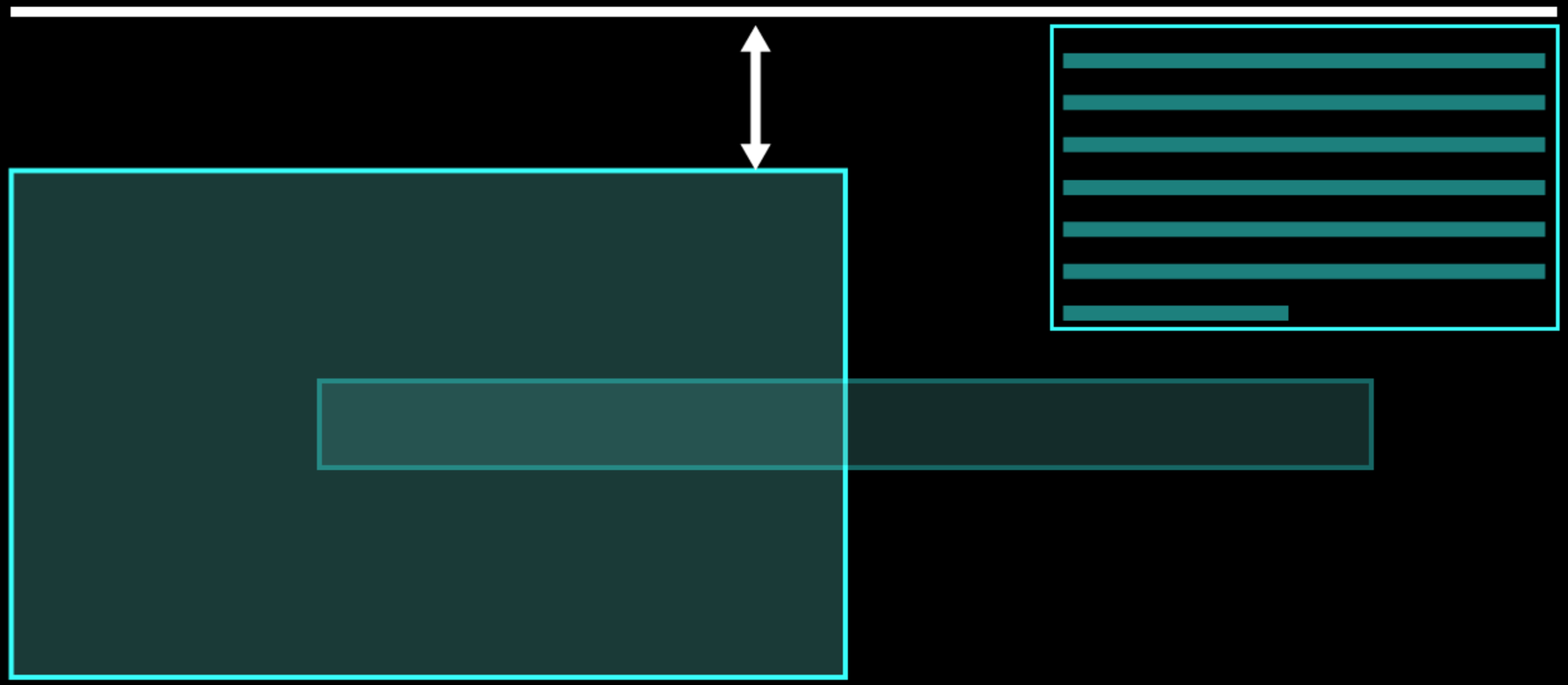


…no matter the size of the image or the text

2. Text block must align to the top of the image

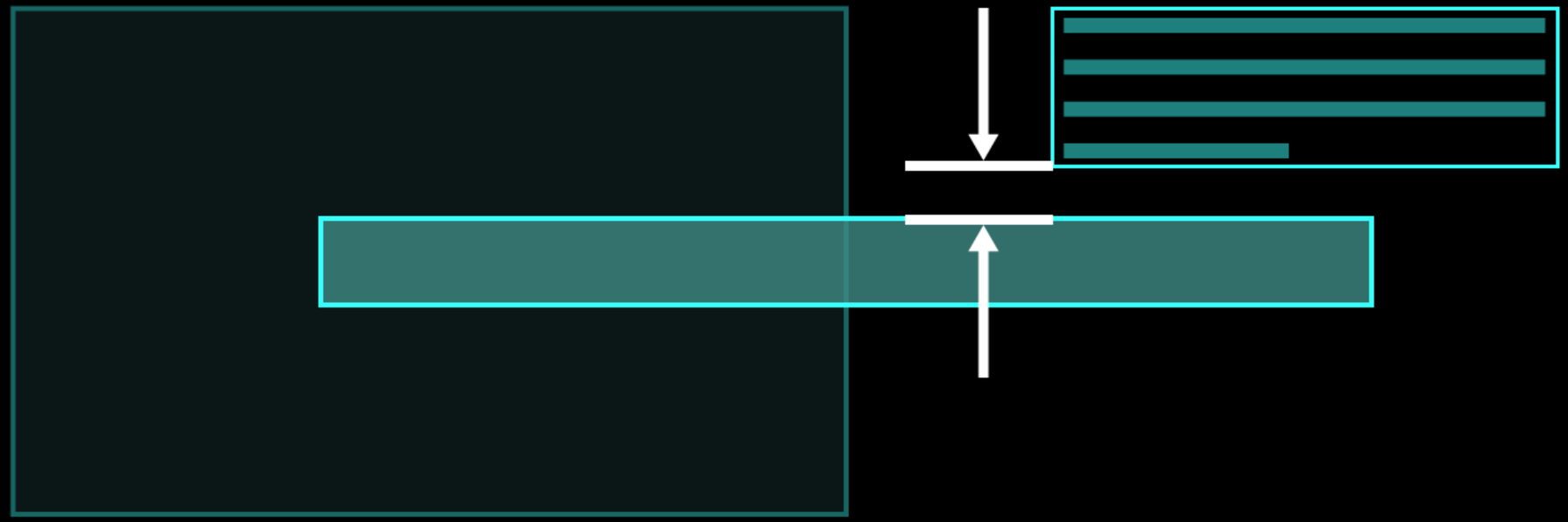


2. Text block must align to the top of the image



…unless the text is too long

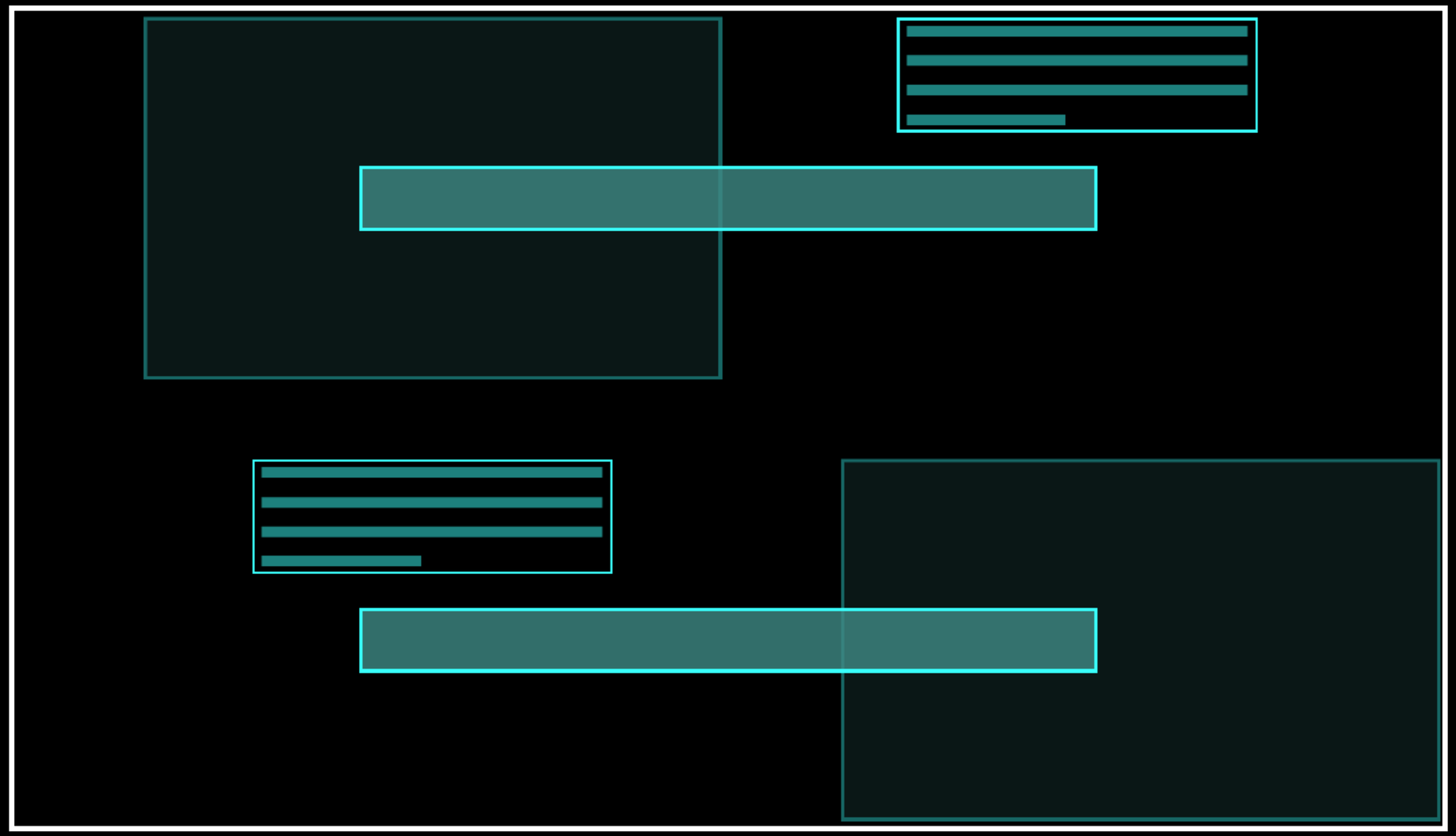
3. Space must always be maintained between the text block and heading

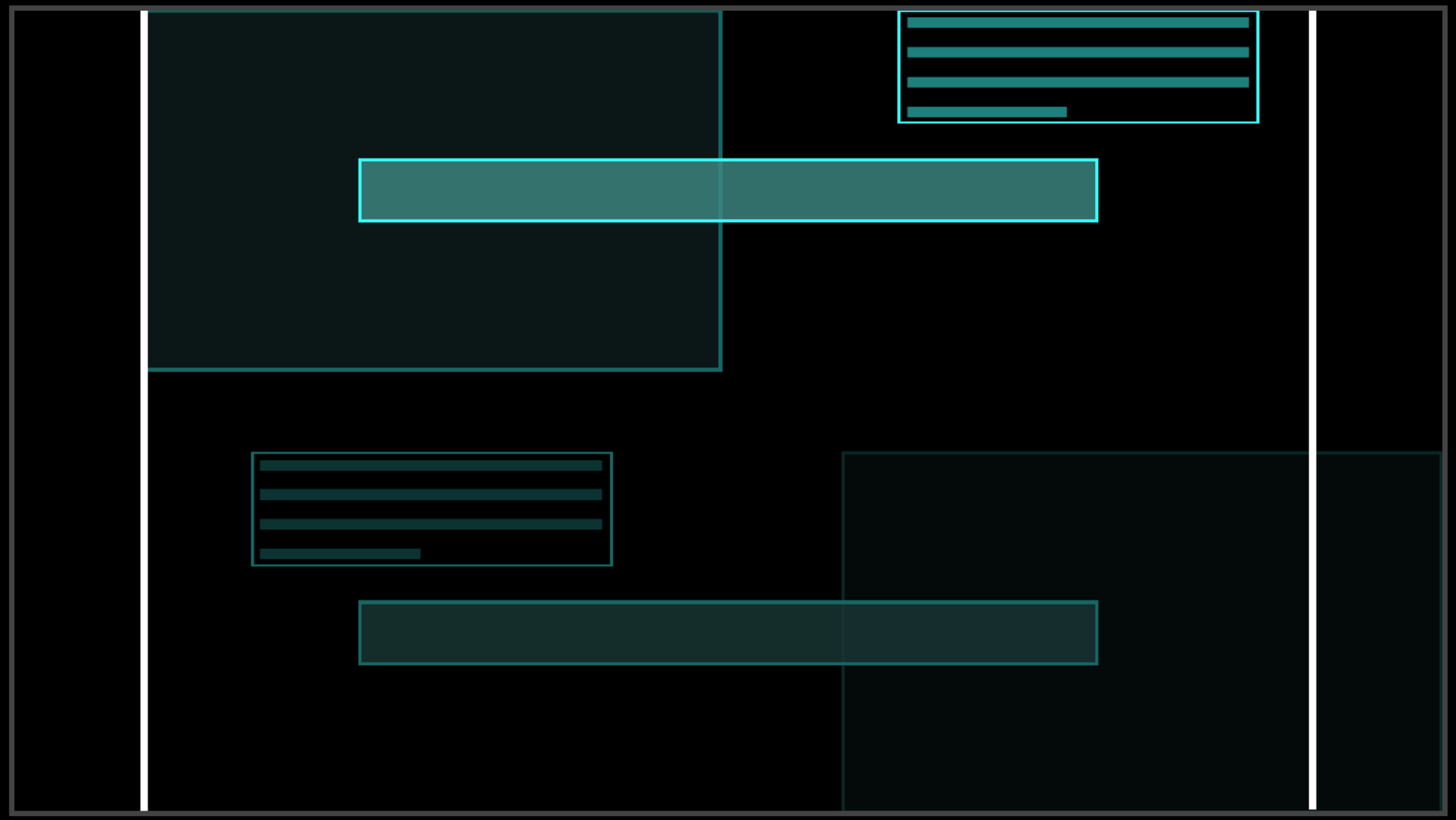


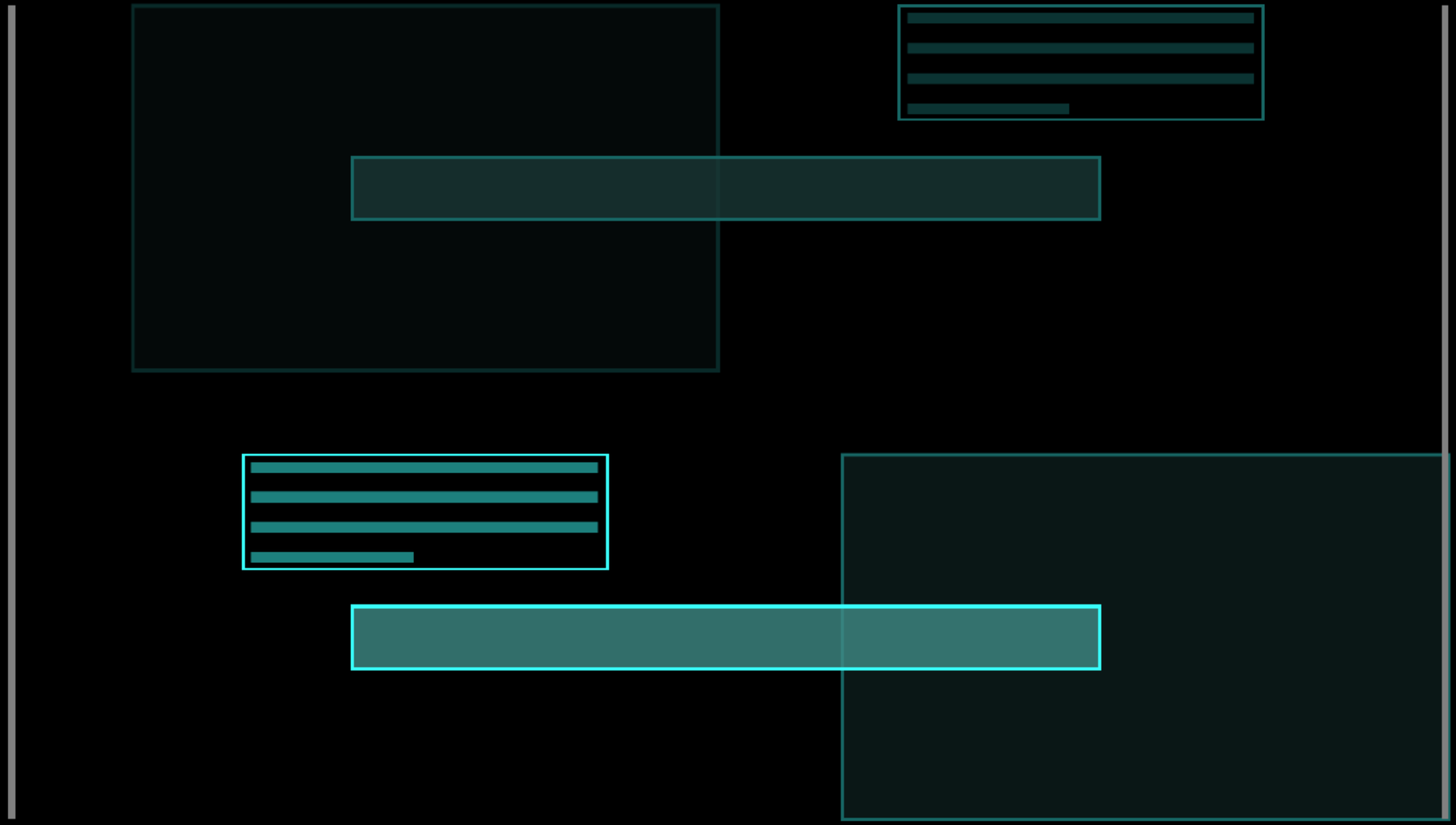
Context-aware

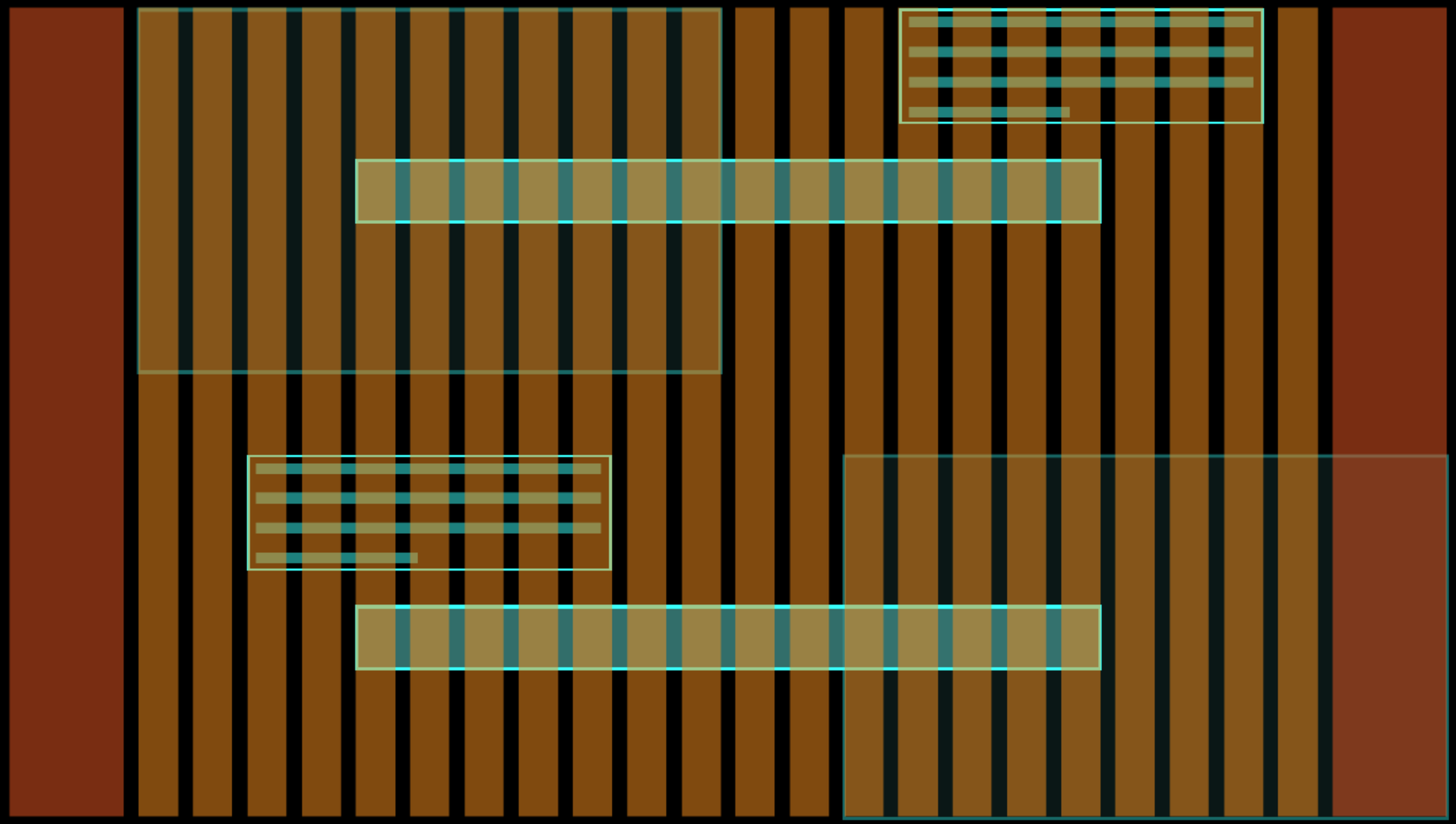
CSS Grid

2D Layout

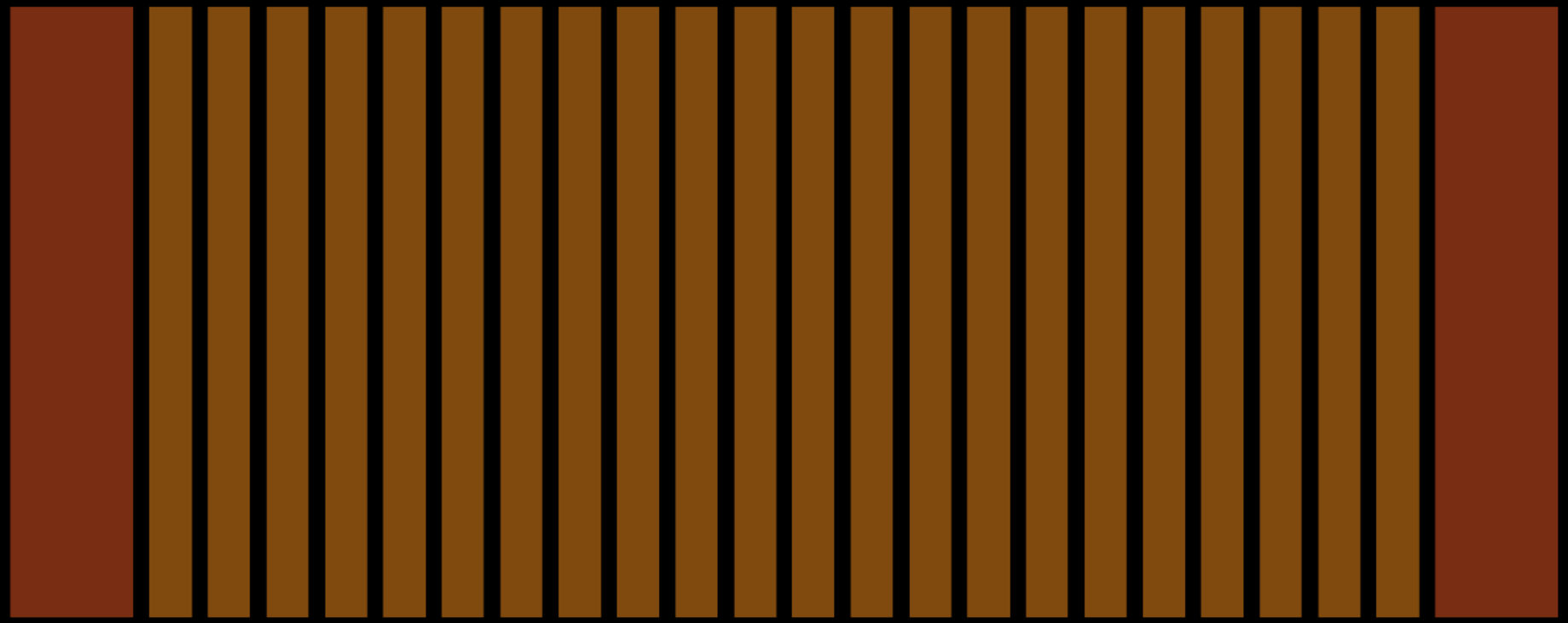




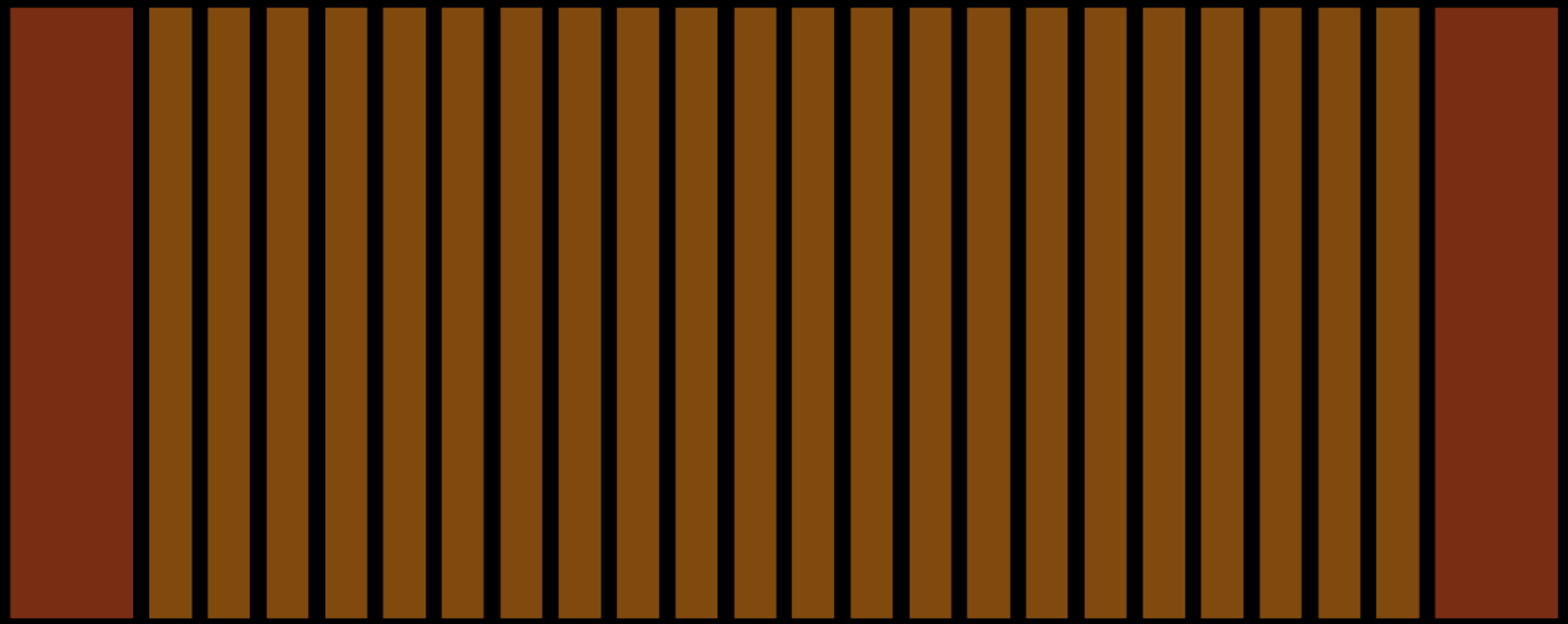


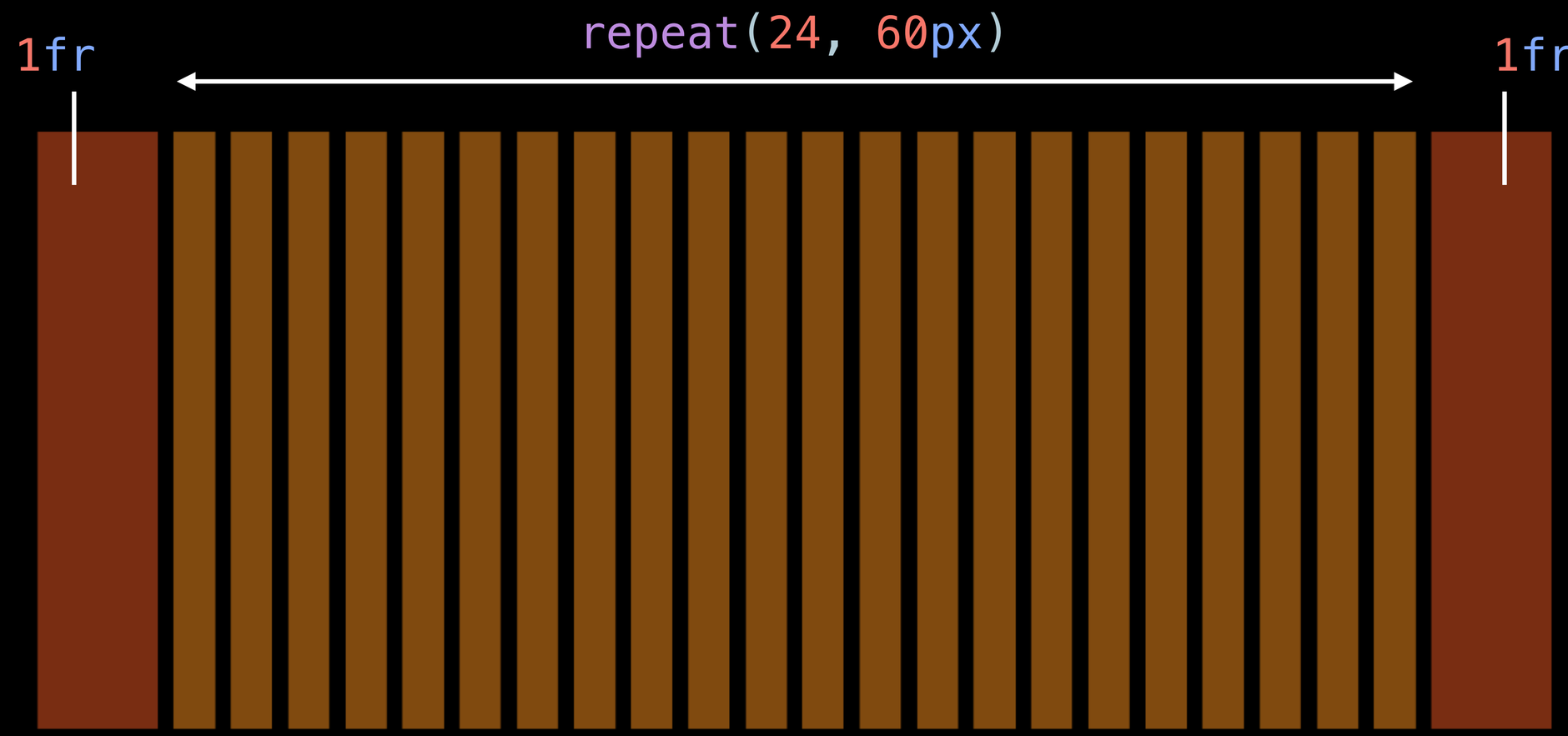


```
grid-template-columns: repeat(26, 1fr);
```

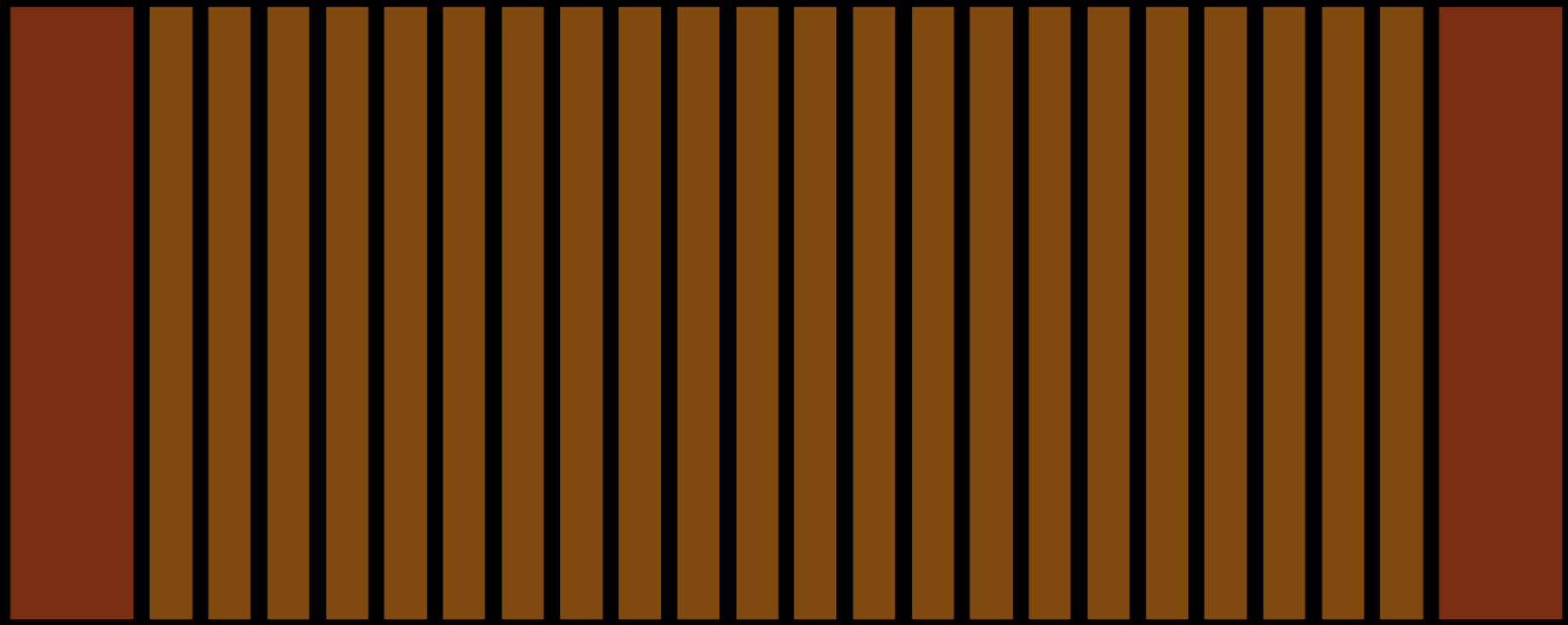


~~grid-template-columns: repeat(26, 1fr);~~



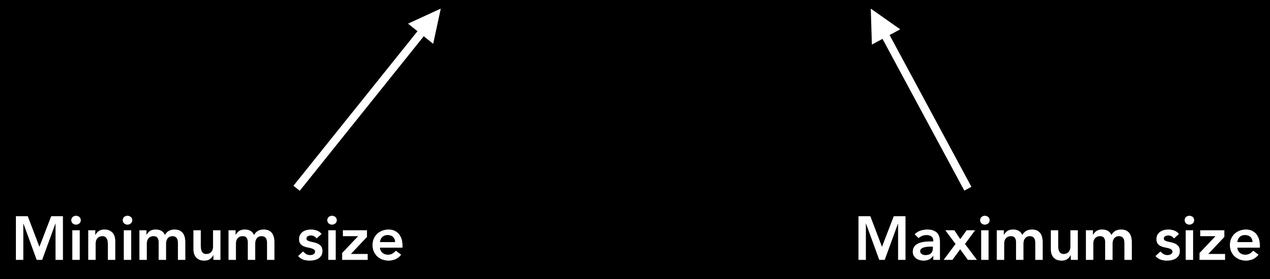

1fr
repeat(24, 60px)
1fr

```
grid-template-columns: 1fr repeat(24, 60px) 1fr;
```

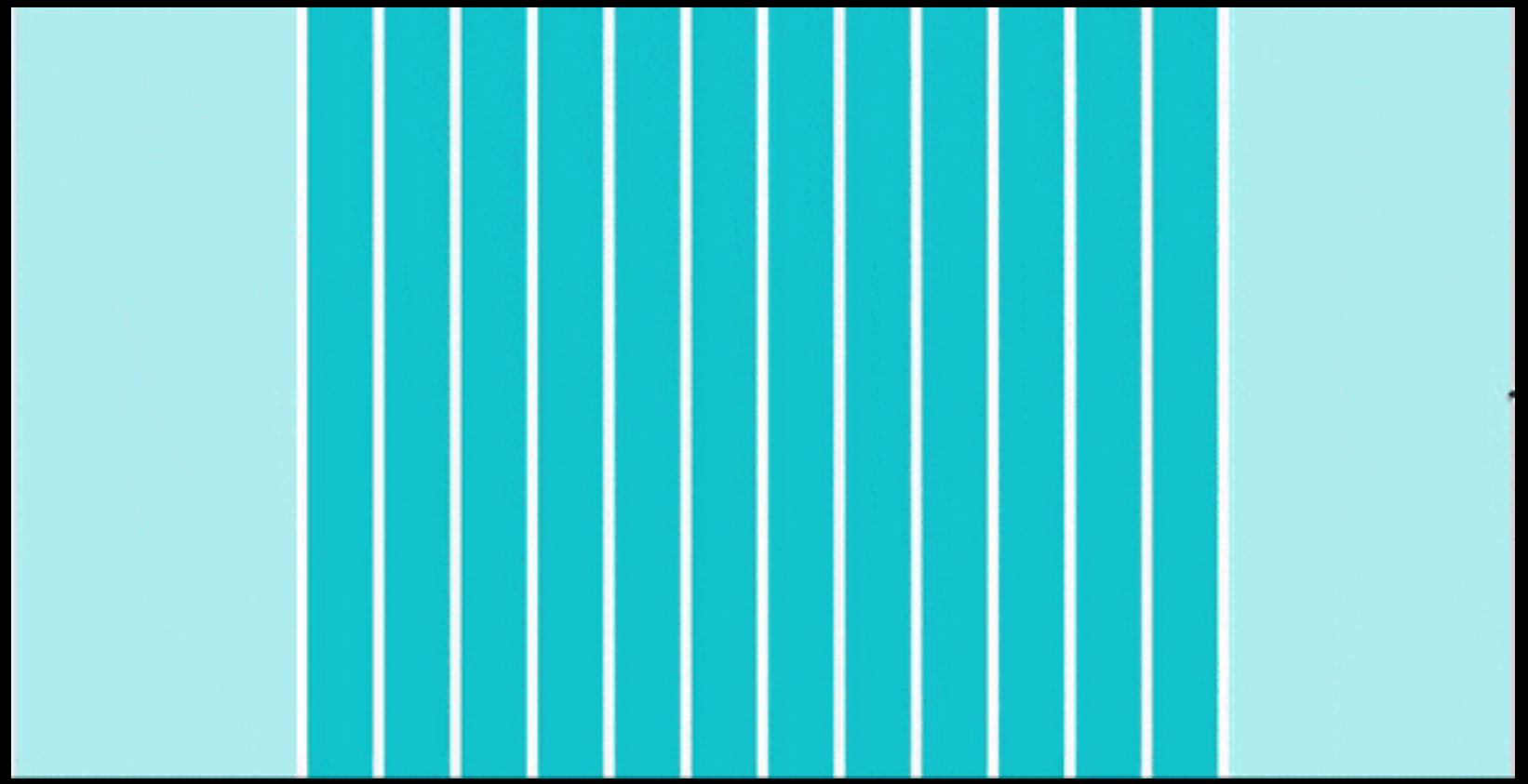


minmax()

```
minmax(20px, 1fr)
```



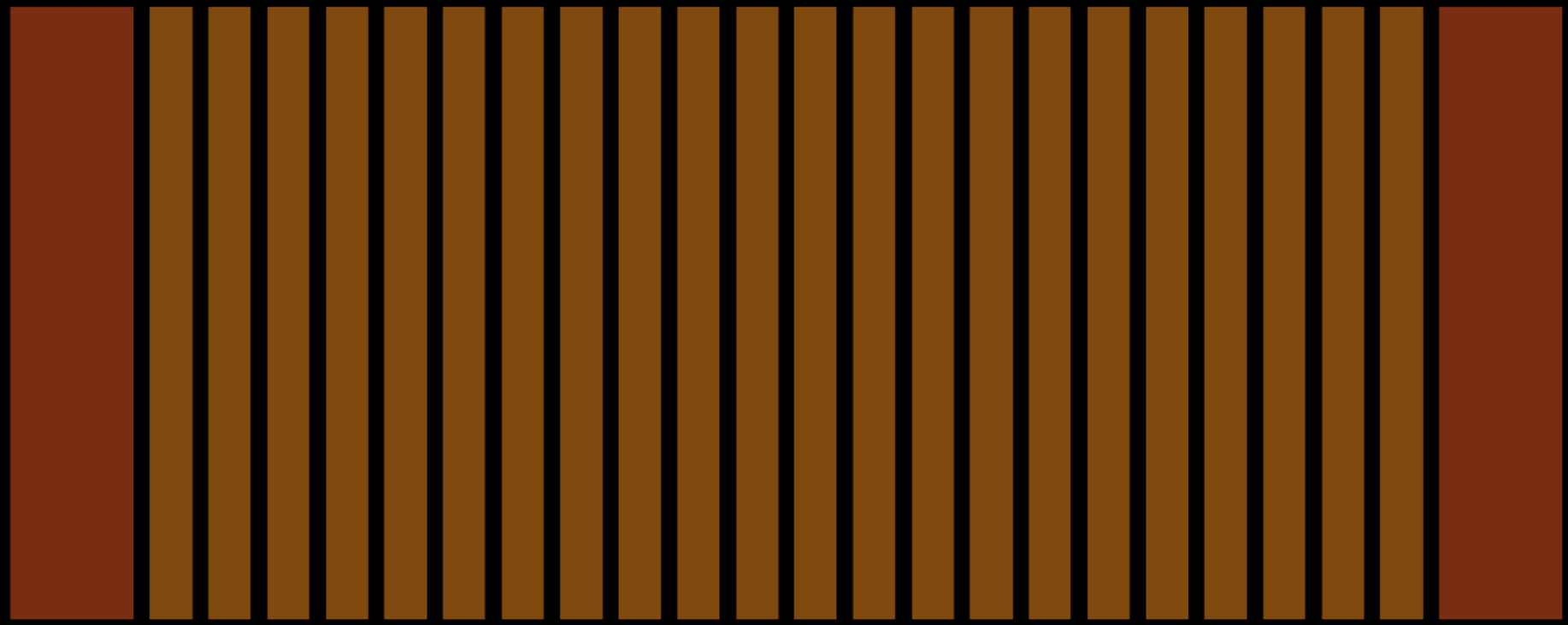
minmax(20px, 1fr)



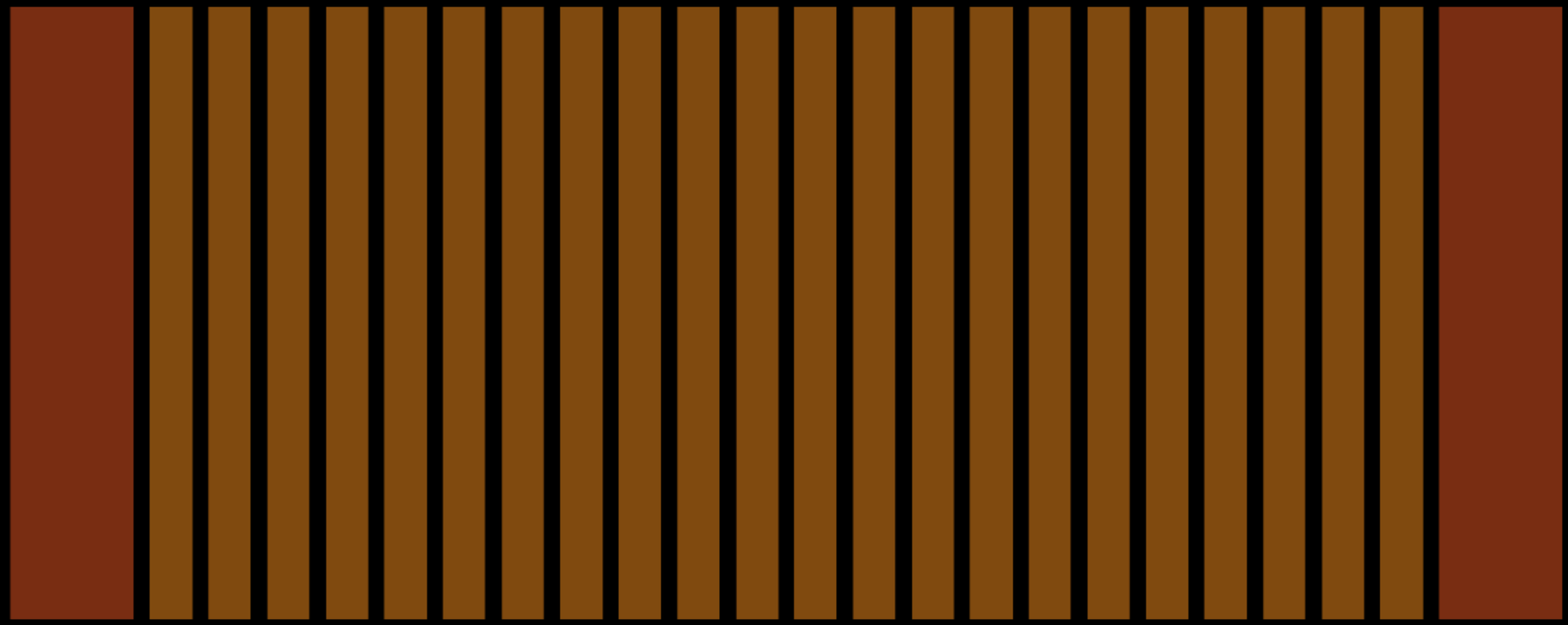
minmax(0, 60px)

codepen.io/michellebarker/pen/wYGMPQ

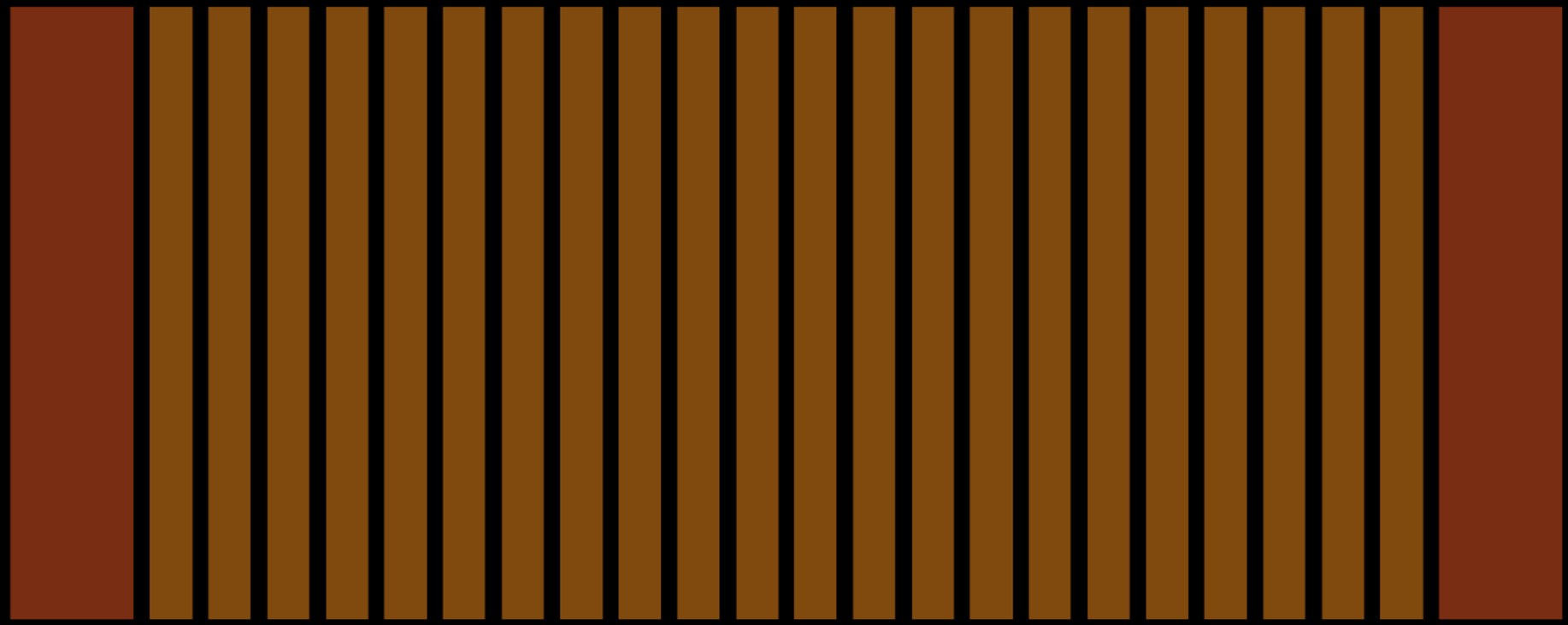
```
grid-template-columns:
minmax(20px, 1fr) repeat(24, minmax(0, 60px)) minmax(20px, 1fr);
```



```
grid-template-columns:
  minmax(20px, 1fr) repeat(24, minmax(0, 60px)) minmax(20px, 1fr);
```



```
grid-template-columns:
minmax(20px, 1fr) repeat(24, minmax(0, 60px)) minmax(20px, 1fr);
```



CSS

```
.grid {
  display: grid;
  grid-template-columns:
      minmax(20px, 1fr) // flexible outer column
      repeat(24, minmax(0, 60px)) // 24 central columns
      minmax(20px, 1fr); // flexible outer column
  column-gap: 20px;
}
```

```
grid-template-rows: 1fr auto 1fr;
```

CSS

```
.grid {
  display: grid;
  grid-template-columns:
      minmax(20px, 1fr)
      repeat(24, minmax(0, 60px))
      minmax(20px, 1fr);
  grid-template-rows: 1fr auto 1fr;
  column-gap: 20px;
}
```

CSS

```
.grid {
  display: grid;
  grid-template-columns:
      minmax(20px, 1fr)
      repeat(24, minmax(0, 60px))
      minmax(20px, 1fr);
  grid-template-rows: 1fr auto 1fr;
  gap: 40px 20px;
}
```

CSS

Placing items

```
.grid {
  grid-template-columns:
      [outer-start]
      minmax(20px, 1fr)
      [wrapper-start]
      repeat(24, minmax(0, 60px))
      [wrapper-end]
      minmax(20px, 1fr)
      [outer-end];
  grid-template-rows:
      1fr [heading-start] auto [heading-end] 1fr;
  …
}
```

CSS

Placing items

```
.grid {
  grid-template-columns:
      [outer-start]
      minmax(20px, 1fr)
      [wrapper-start]
      repeat(24, minmax(0, 60px))
      [wrapper-end]
      minmax(20px, 1fr)
      [outer-end];
  grid-template-rows:
      1fr [heading-start] auto [heading-end] 1fr;
  …
}
```

CSS

Placing items

```
.grid {
  grid-template-columns:
      [outer-start]
      minmax(20px, 1fr)
      [wrapper-start]
      repeat(24, minmax(0, 60px))
      [wrapper-end]
      minmax(20px, 1fr)
      [outer-end];
  grid-template-rows:
      1fr [heading-start] auto [heading-end] 1fr;
  …
}
```

CSS

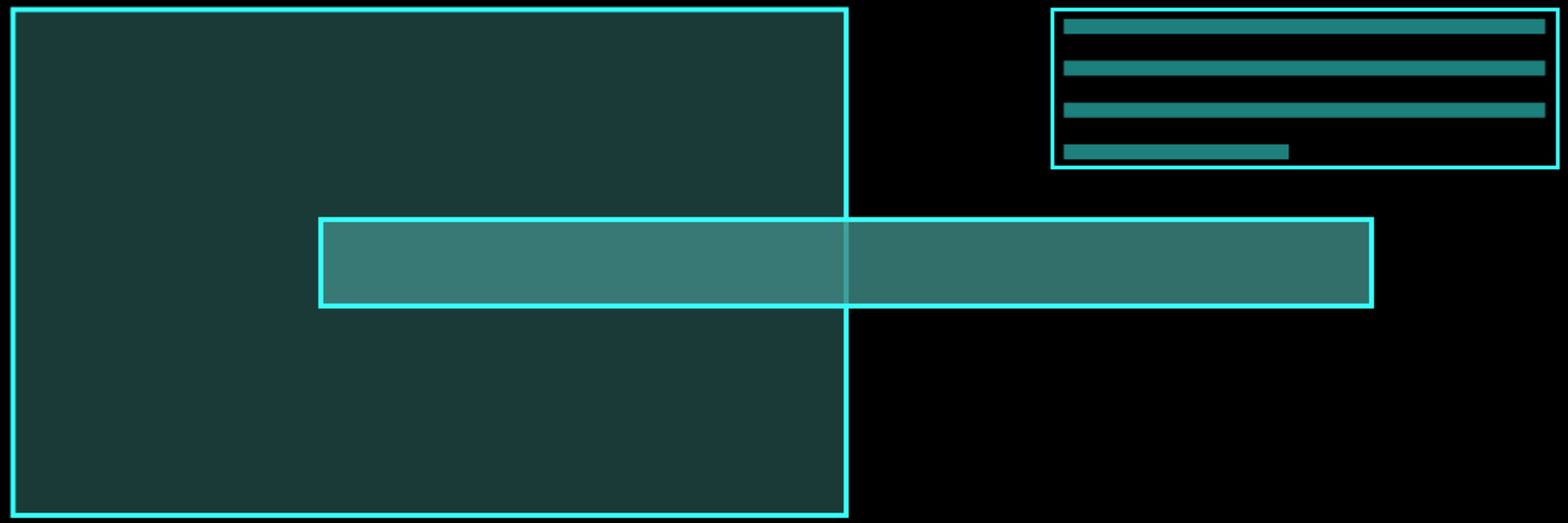
```css
.grid__image {
  grid-column: wrapper-start / 14;
  grid-row: 1 / -1;
}

.grid__body {
  grid-column: span 5 / wrapper-end;
  grid-row: 1;
  align-self: flex-start;
}

.grid__heading {
  grid-column: 5 / -5;
  grid-row: heading;
}
```

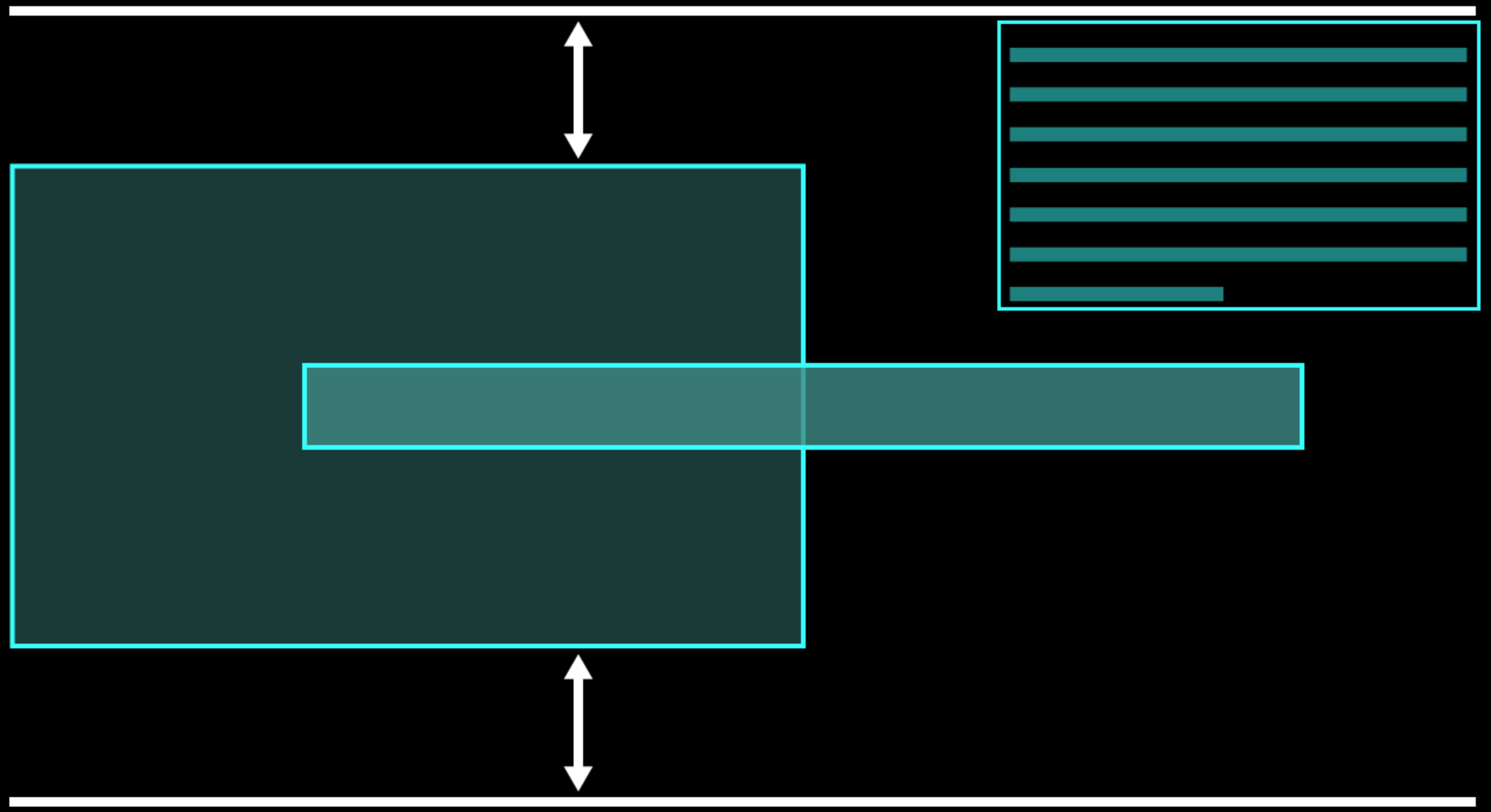
CSS

```
.grid__image {
  grid-column: wrapper-start / 14;
  grid-row: 1 / -1;
}

.grid__body {
  grid-column: span 5 / wrapper-end;
  grid-row: 1;
  align-self: flex-start;
}

.grid__heading {
  grid-column: 5 / -5;
  grid-row: heading;
}
```

CSS

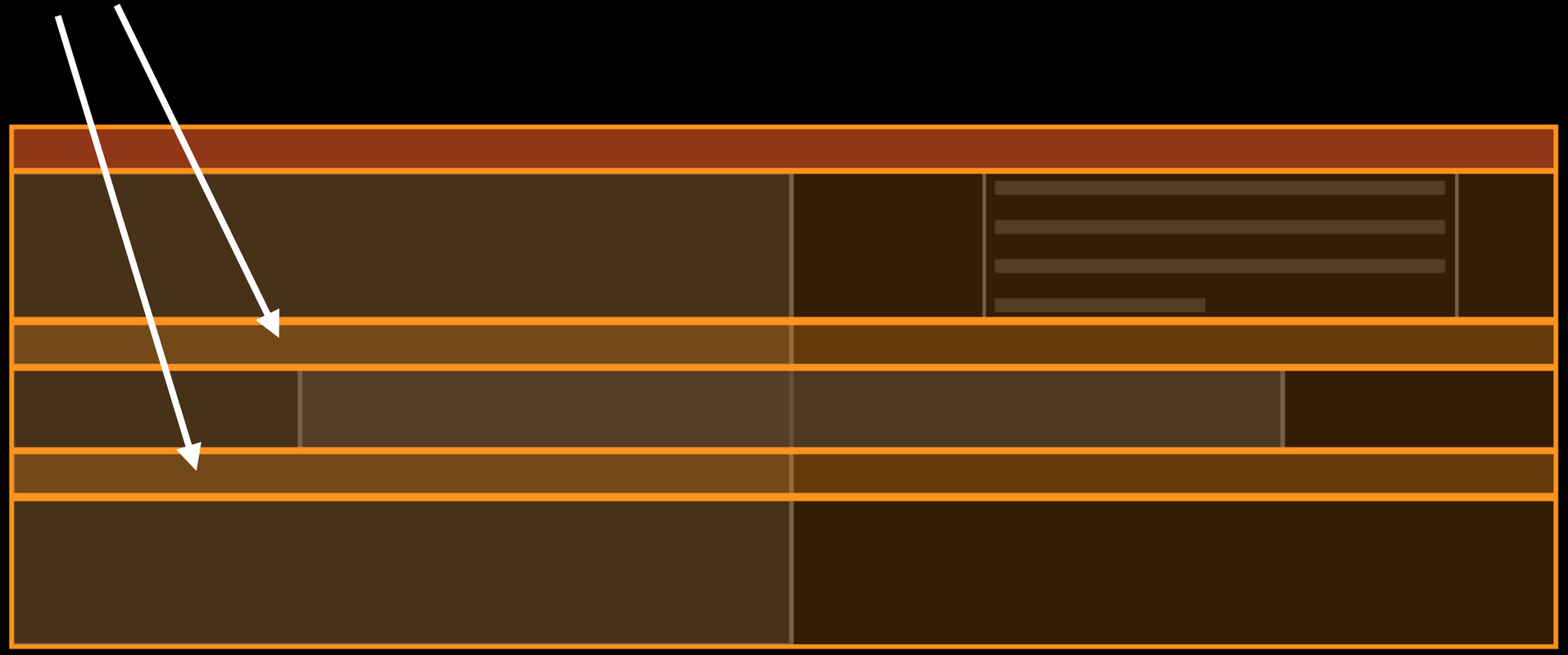
```
.grid__image {
  grid-column: wrapper-start / 14;
  grid-row: 1 / -1;
}

.grid__body {
  grid-column: span 5 / wrapper-end;
  grid-row: 1;
  align-self: flex-start;
}

.grid__heading {
  grid-column: 5 / -5;
  grid-row: heading;
}
```

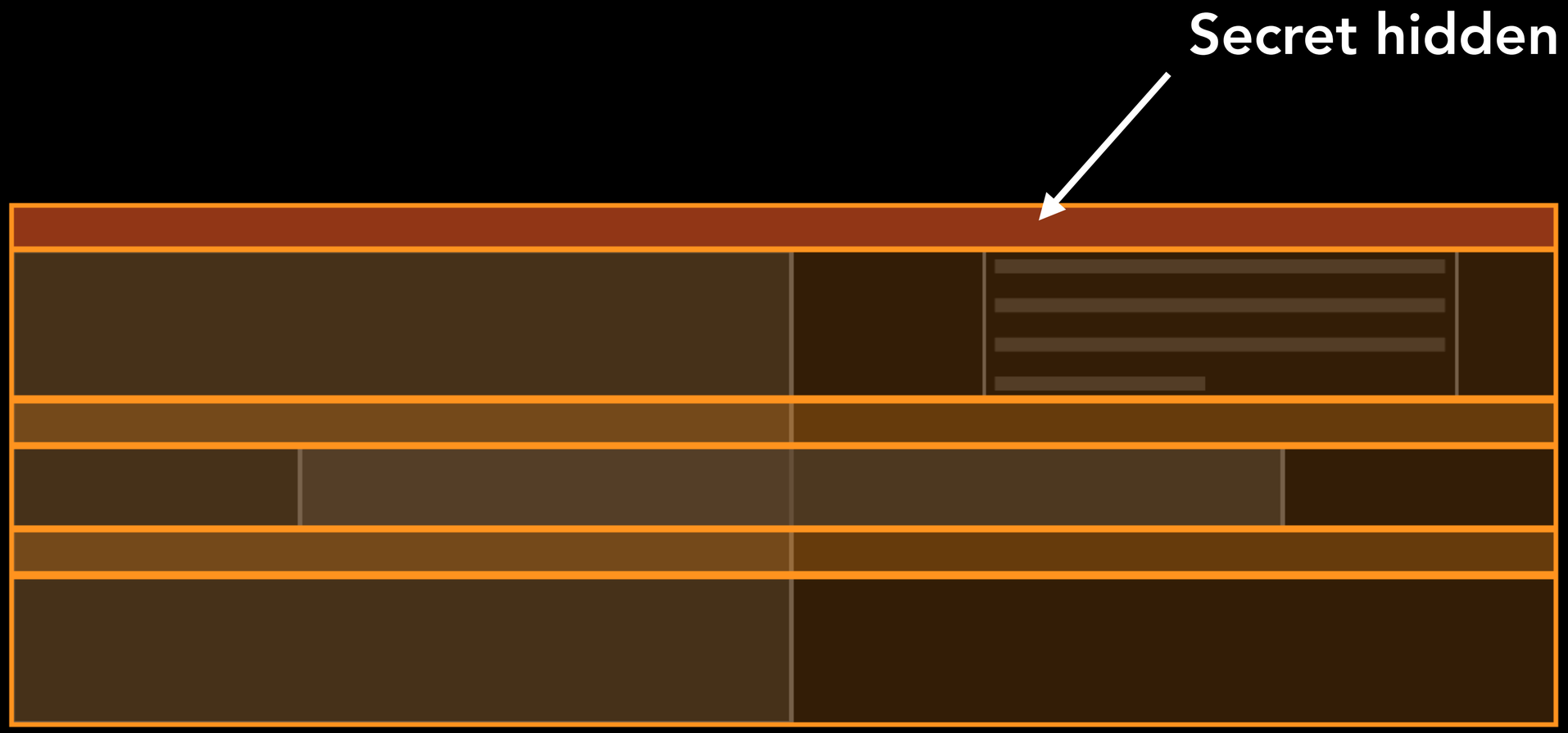




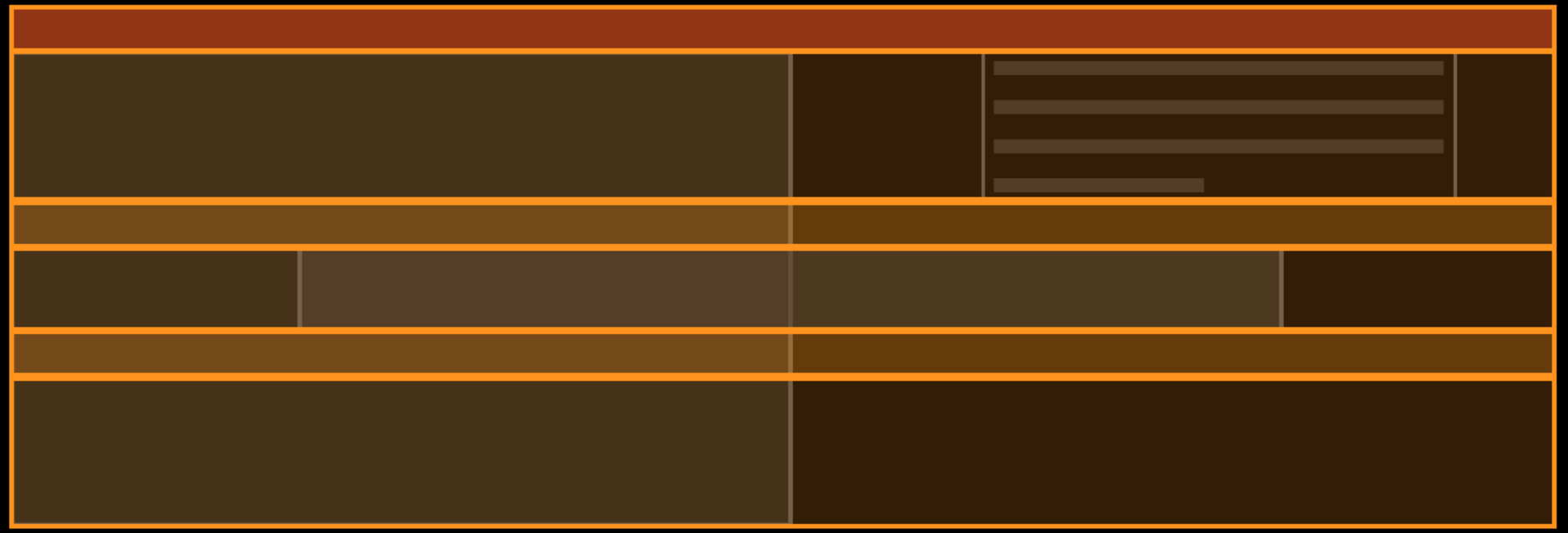
I didn’t solve this problem

Gutter rows

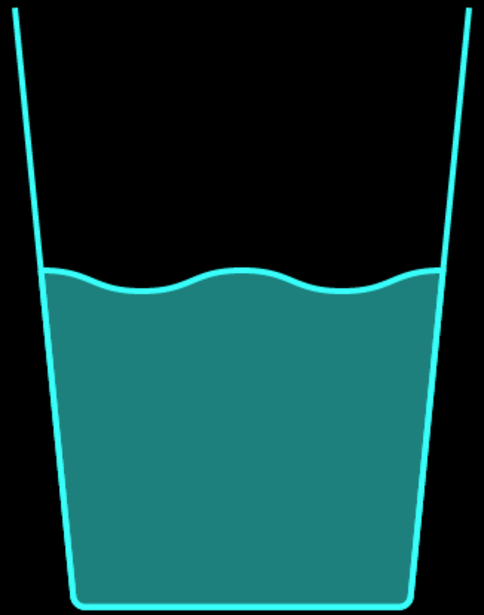


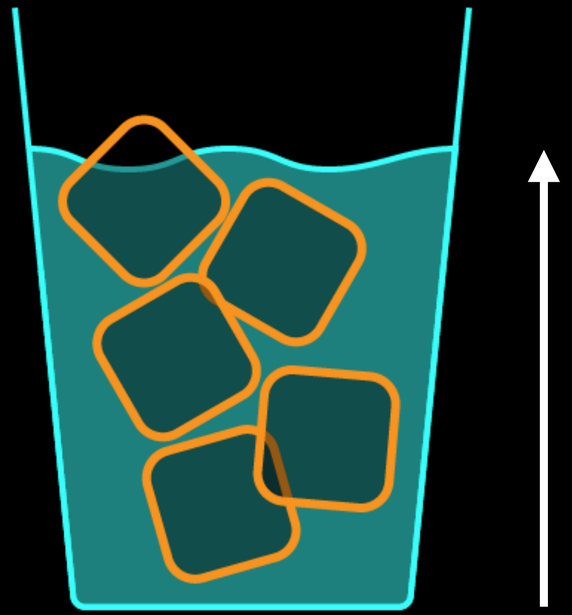

Secret hidden row

```
grid-template-rows: 1fr auto 40px auto 40px auto;
```



```
grid-template-rows: 1fr auto 40px auto 40px auto;
```

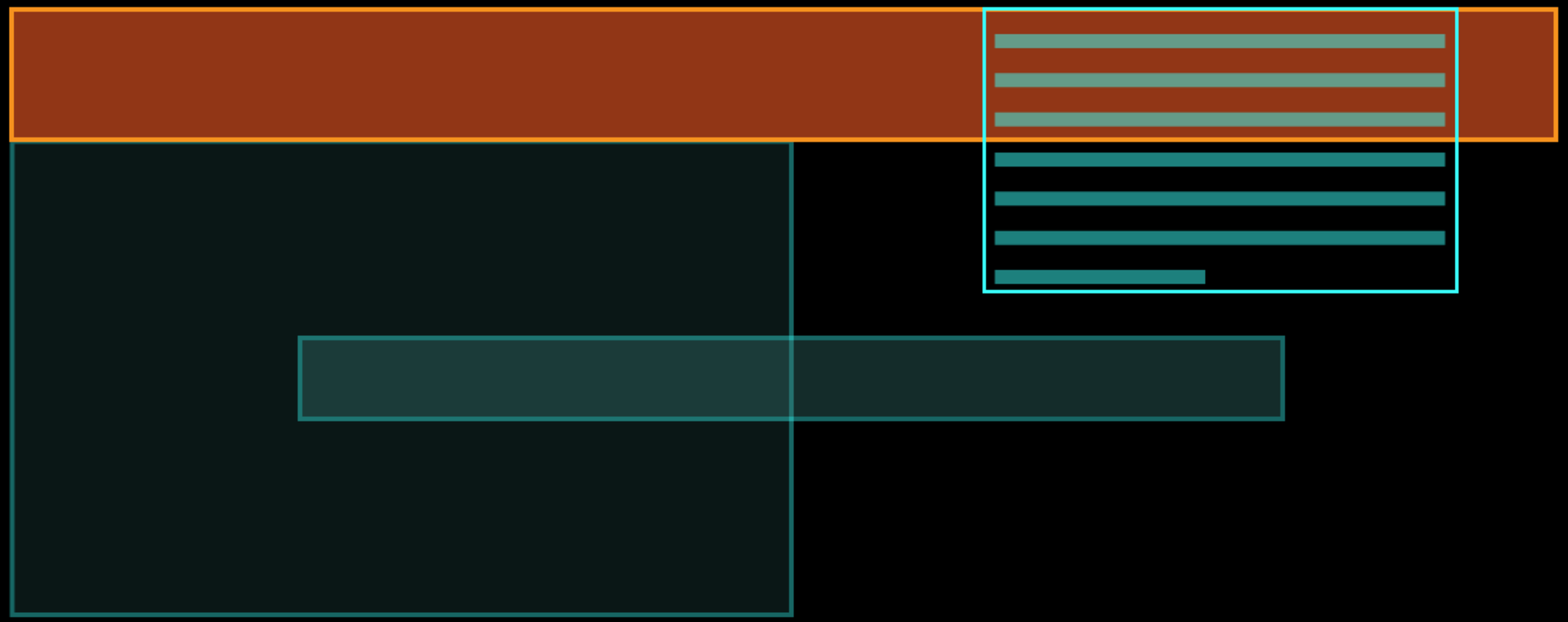


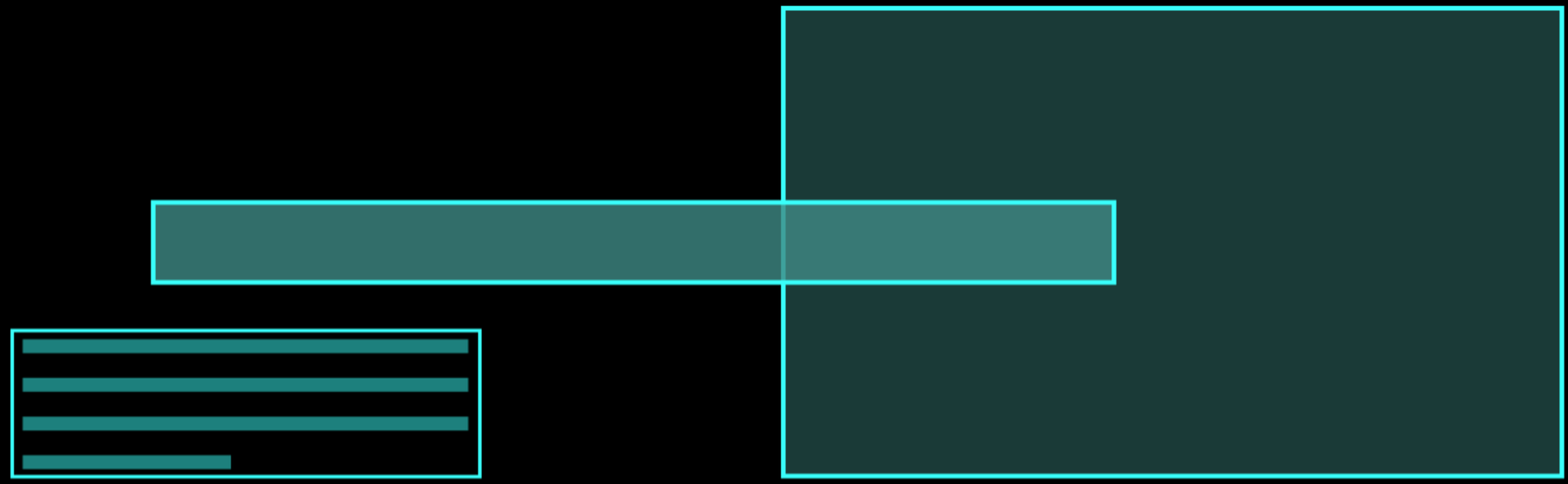


CSS

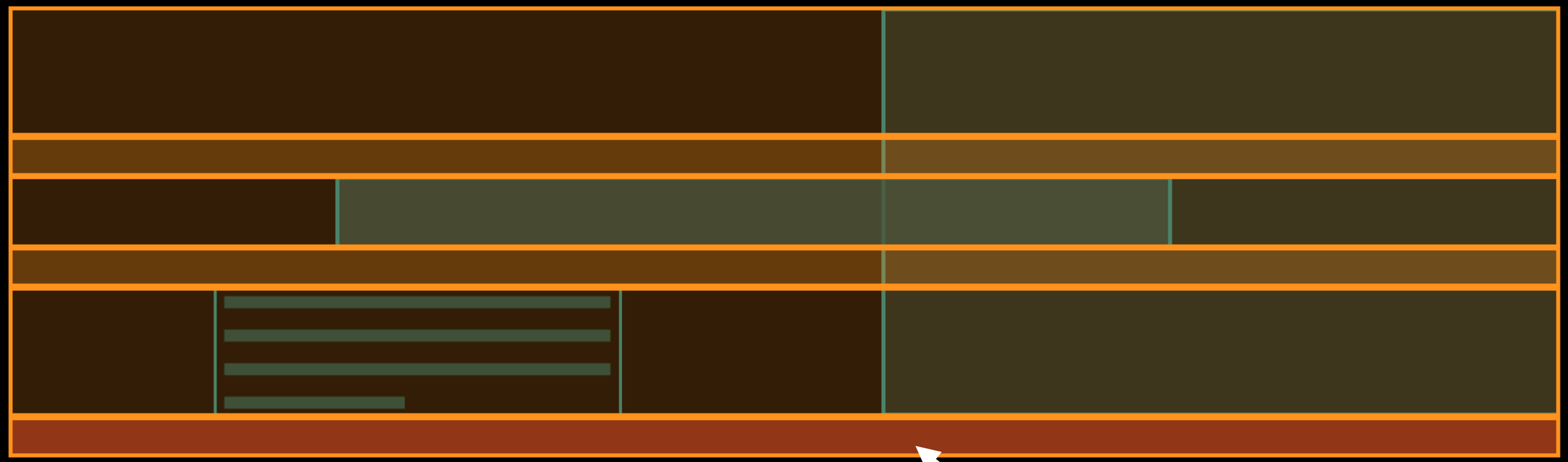
```
.grid {
  …
  grid-template-rows:
      [text-start] 1fr auto [text-end]
      40px // gap
      [heading-start] auto [heading-start]
      40px // gap
      auto;
  …
}
```

```
.grid__body {
  grid-row: text;
}
```





```
grid-template-rows: auto 40px auto 40px auto 1fr;
```



Secret hidden row

CSS

```
.grid--text-bottom {
  grid-template-rows:
      auto 40px
      [heading-start] auto [heading-end]
      40px [text-start] auto
      1fr [text-end]; // hidden row
}
```

CSS

```
.grid--text-bottom .grid__body {
  align-self: flex-end;
}
```

Component variants

SCSS

```scss
/* Second variant of our component */
.grid--2 {
  .grid__image {
    grid-column: 13 / outer-end;
  }

  .grid__body {
    grid-column: wrapper-start / span 5;
  }
}
```

CSS Variables

(a.k.a. Custom Properties)

@MicheBarks

CSS Variables == Dynamic variables

CSS

Defining variables

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

CSS

Using variables

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

CSS

```
.grid__image {
  grid-column:
   var(--imageColStart) / var(--imageColEnd);
  grid-row: media;
}

.grid__body {
  grid-column: var(--textColStart) / span 5;
  grid-row: text;
}
```

CSS

```
.grid__image {
  grid-column:
   var(--imageColStart) / var(-imageColEnd);
  grid-row: media;
}

.grid__body {
  grid-column: var(--textColStart) / span 5;
  grid-row: text;
}
```

CSS

```
.grid__image {
  grid-column:
   var(--mediaColStart) / var(--mediaColEnd);
  grid-row: media;
}

.grid__body {
  grid-column: var(--textColStart) / span 5;
  grid-row: text;
}
```

CSS

```css
.grid {
  --imageColStart: wrapper-start;
  --imageColEnd: 14;
  --textColStart: -8;
}

.grid--2 {
  --imageColStart: 13;
  --imageColEnd: outer-end;
  --textColStart: 3;
}
```

codepen.io/michellebarker/pen/yGzWme

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

@MicheBarks / @CSSInRealLife