

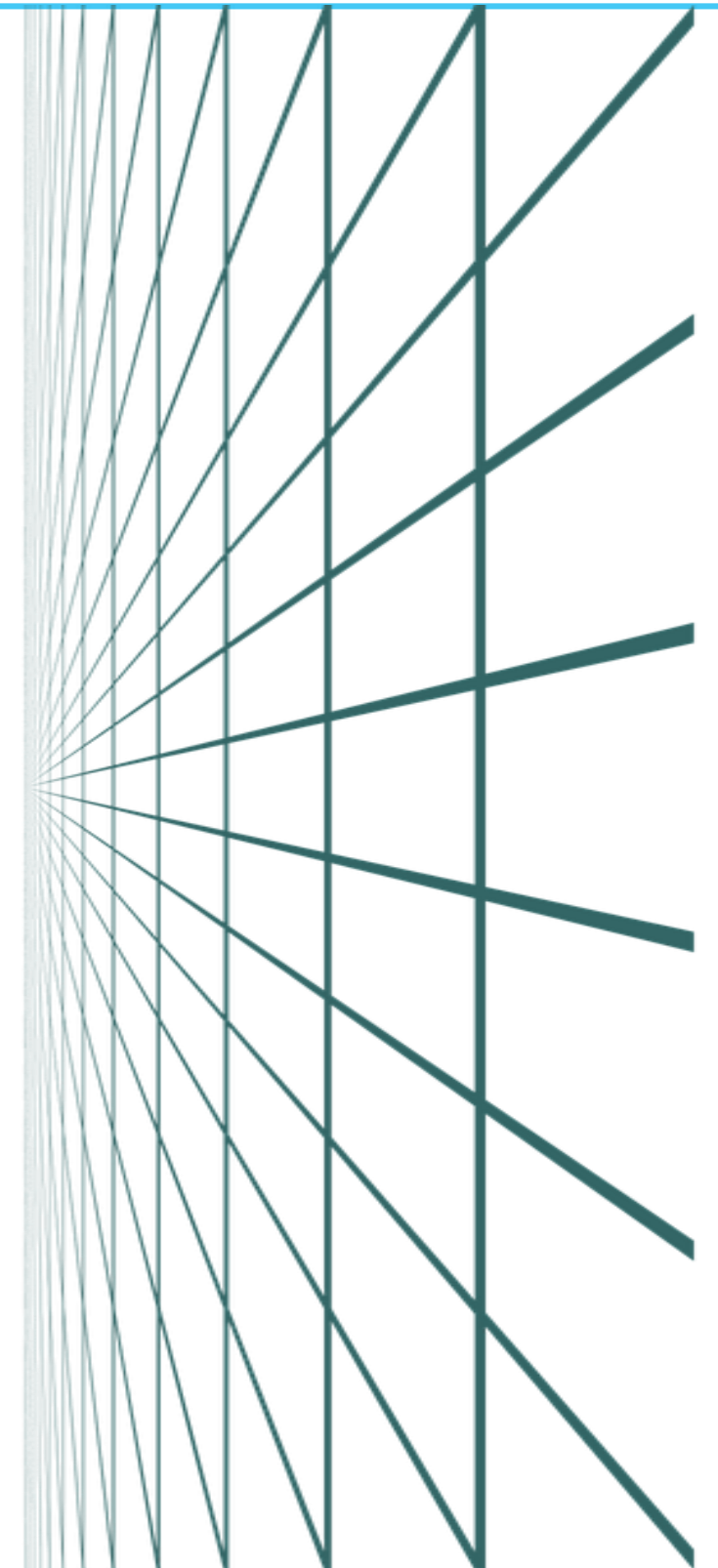


CSS Grid

Web, Mobile and Security
Frédéric Vlummens

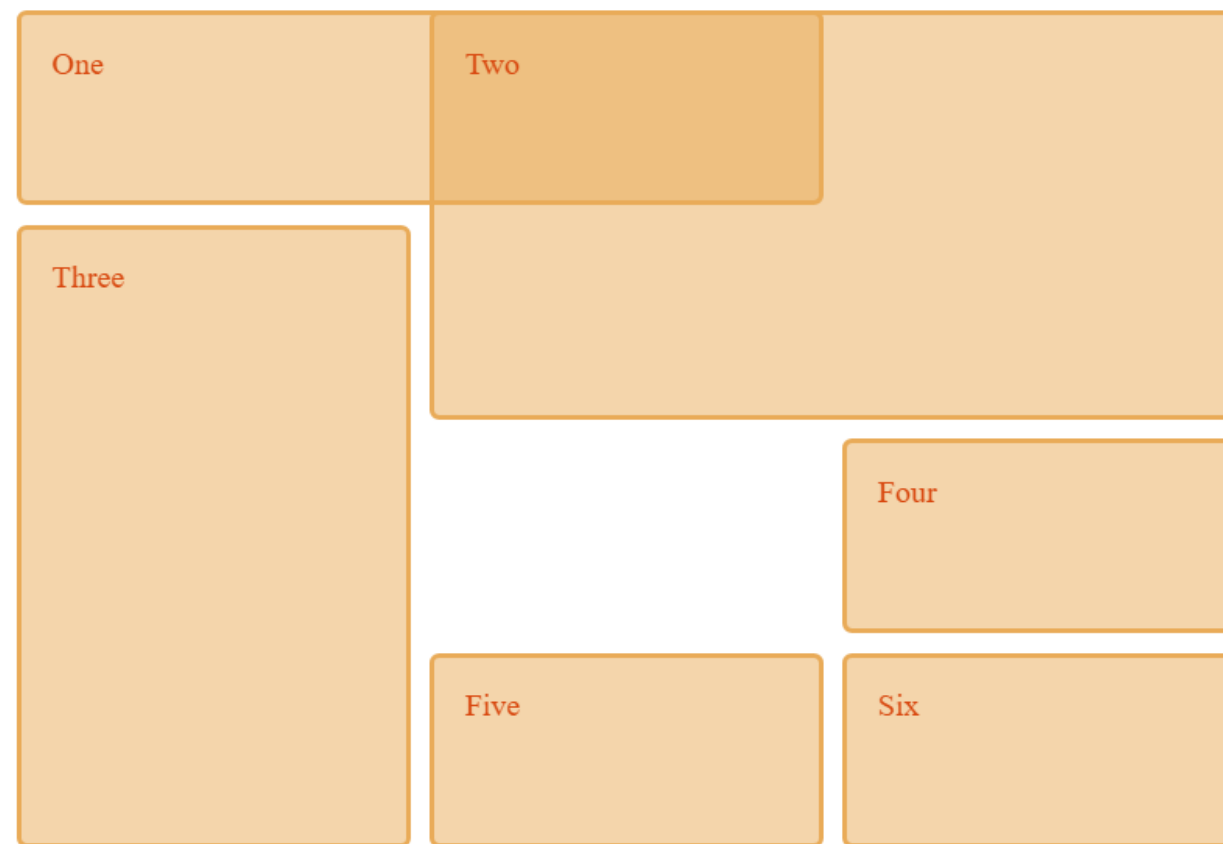
Agenda

- What is Grid Layout?
- Basic concepts
- Defining grids
- Implicit vs explicit grids
- (Named) grid lines
- Named grid areas
- Layering items
- Box alignment



What is Grid Layout?

- Goes beyond where Flexbox ends (note: both can be combined)
- Allows you to layout your website in rows and columns
- Other layouts also possible (overlaps, ...)



Source: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout

Basic concepts: grid

- “Intersecting set of horizontal and vertical lines”
- One set defines the **columns**, other the **rows**
- Elements placed on grid within columns/rows
- How to define a grid?

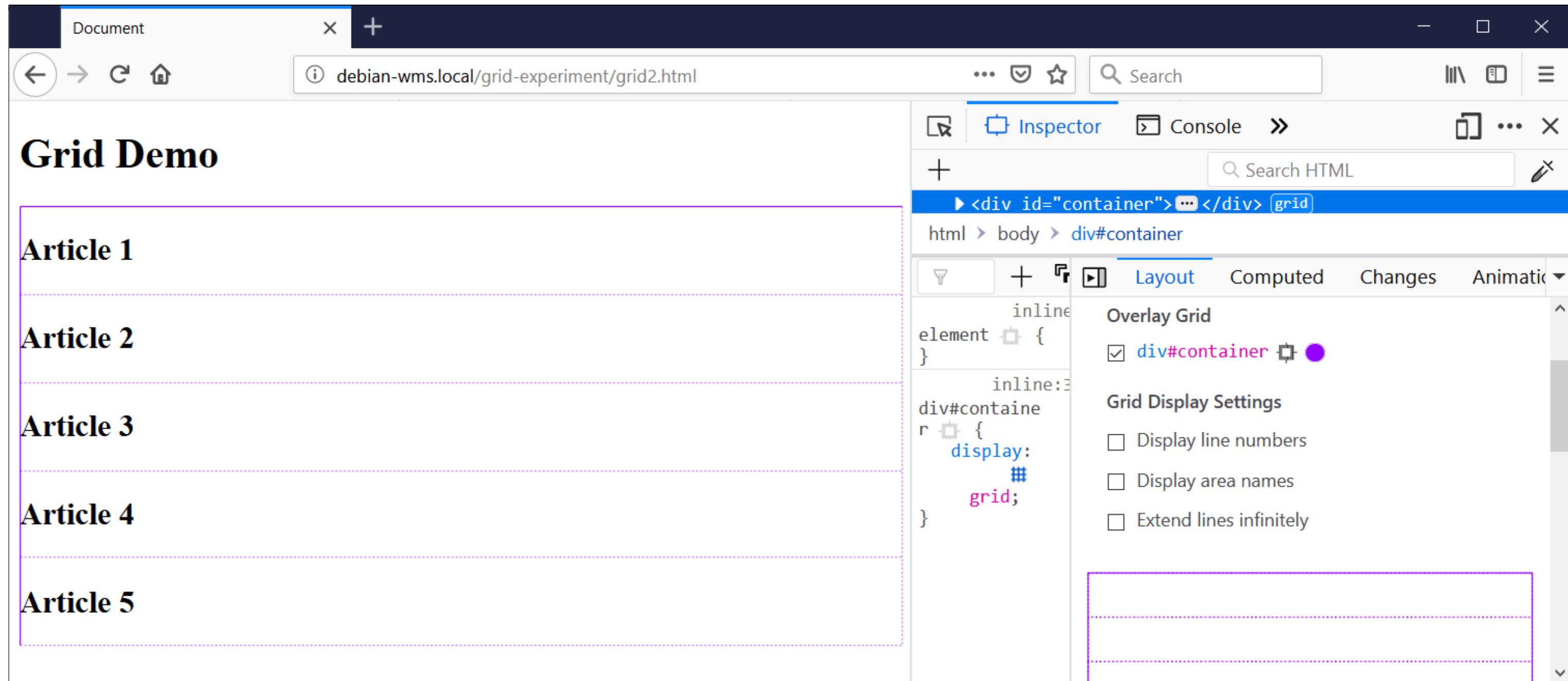
Defining a grid

```
<h1>Grid Demo</h1>

<div id="container">
  <article><h2>Article 1</h2></article>
  <article><h2>Article 2</h2></article>
  <article><h2>Article 3</h2></article>
  <article><h2>Article 4</h2></article>
  <article><h2>Article 5</h2></article>
</div>
```

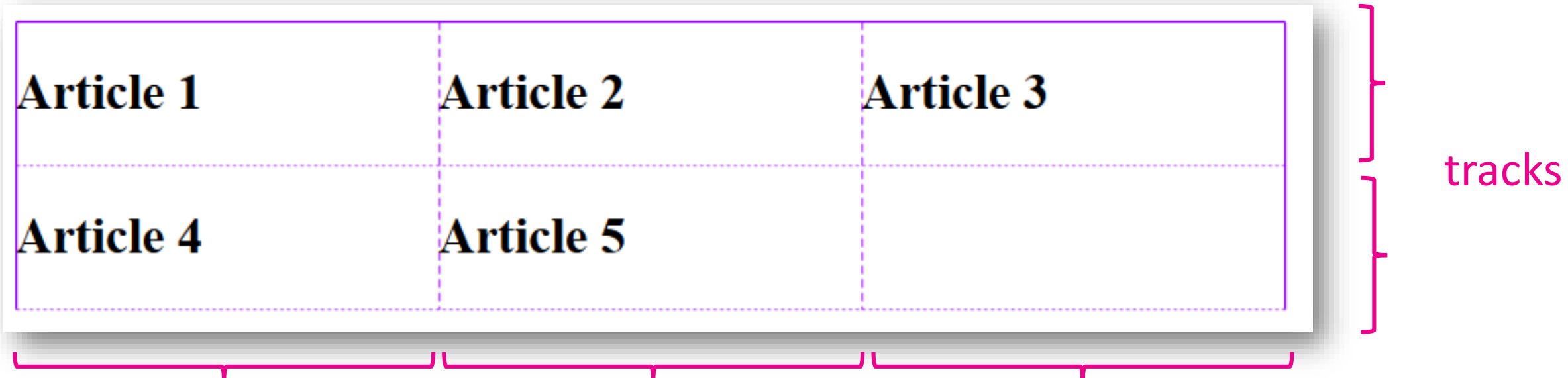
```
div#container {
  display: grid;
}
```

Viewing the grid layout using Firefox developer tools



Defining grid tracks

```
div#container {  
  display: grid;  
  grid-template-columns: 200px 200px 200px;  
}
```



Rows and columns defined using **grid-template-columns** and **grid-template-rows**

This defines so-called grid tracks (=the space between any two lines on grid)

New unit fr (fraction) can also be used

```
div#container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 2fr;  
}
```

Article 1

Article 2

Article 3

Article 4

Article 5

Using repeat()

```
div#container {  
  display: grid;  
  grid-template-columns: repeat(2, 1fr) 2fr  
}
```

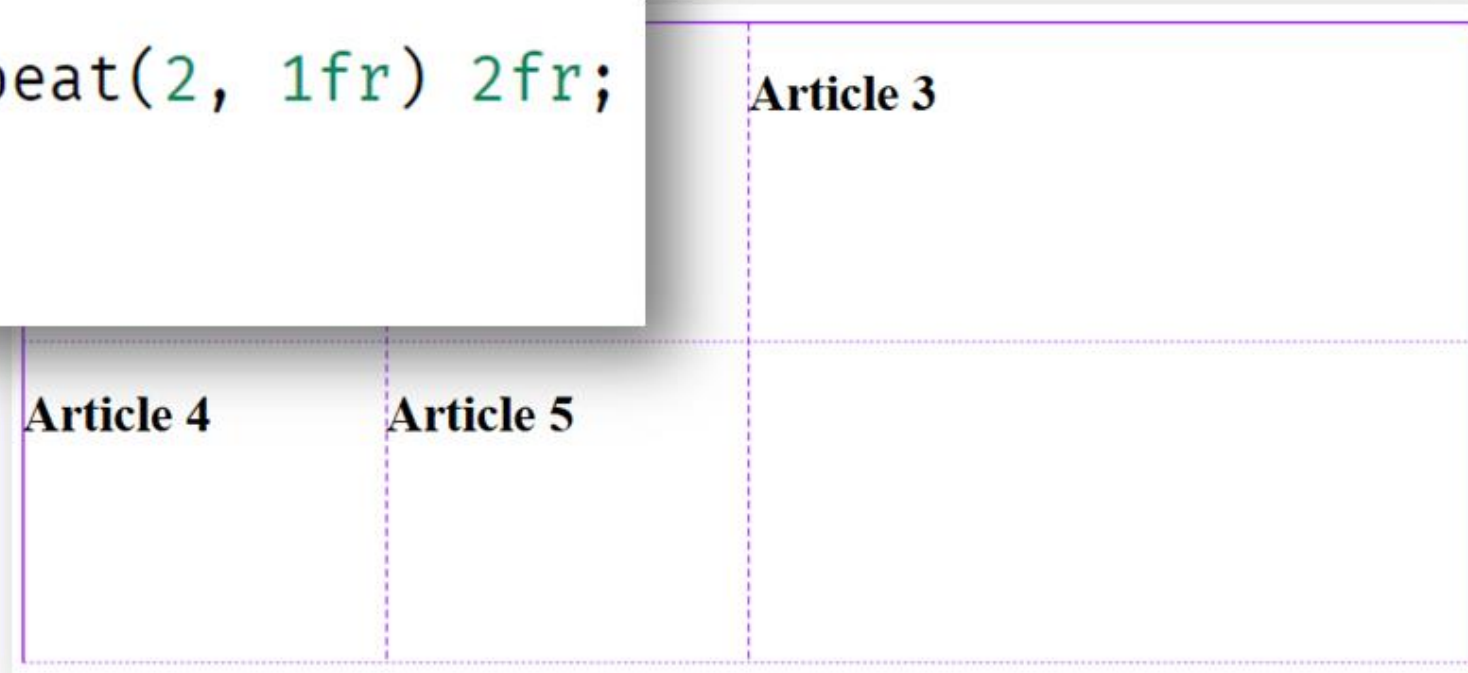
=

```
div#container {  
  display: grid;  
  grid-template-columns: 1fr 1fr 2fr;  
}
```

Implicit vs explicit grid

- Explicit grid = defined using **grid-template-columns** and/or **grid-template-rows**
- Implicit grid = when more grid tracks needed, created automatically
- Size of implicit tracks set using **grid-auto-rows** and **grid-auto-columns**

```
div#container {  
  display: grid;  
  grid-template-columns: repeat(2, 1fr) 2fr;  
  grid-auto-rows: 150px;  
}
```



minmax

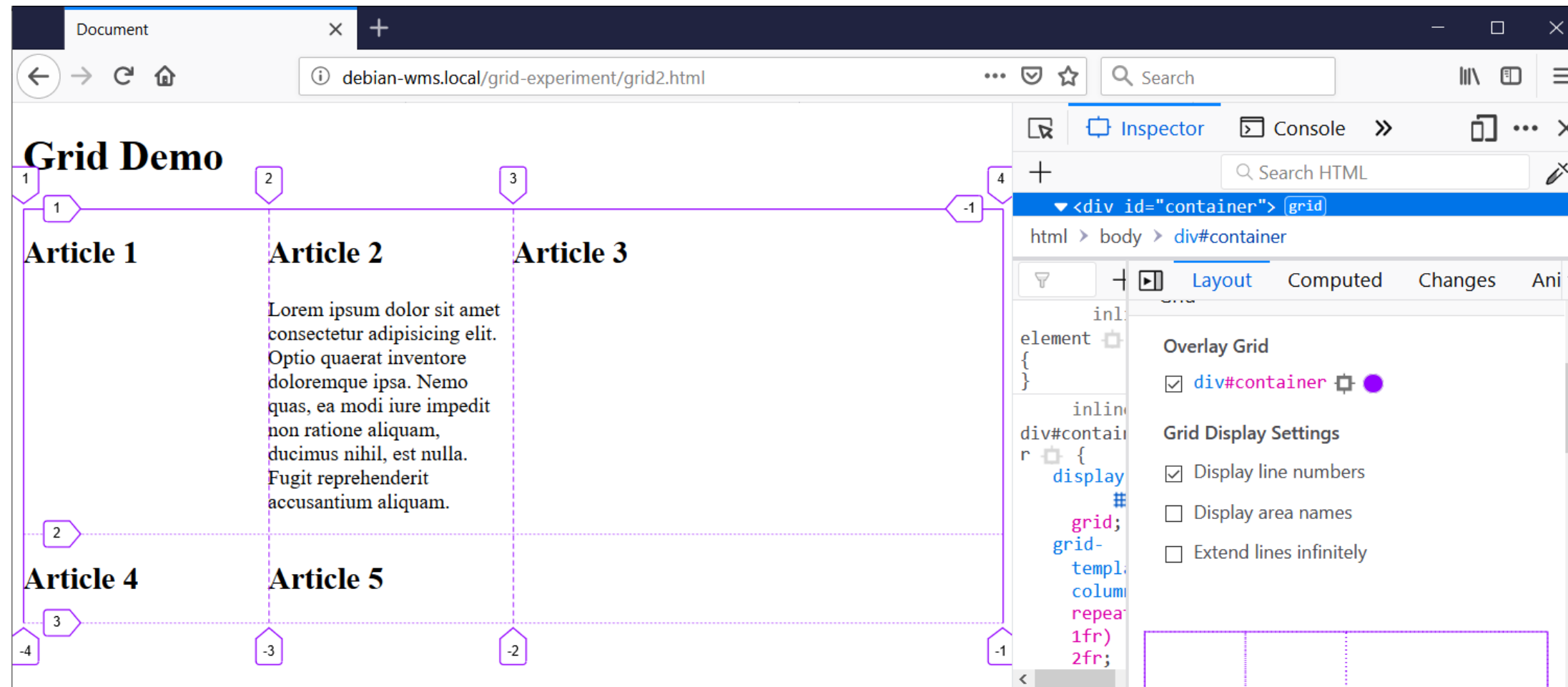
```
div#container {  
  display: grid;  
  grid-template-columns: repeat(2, 1fr) 2fr;  
  grid-auto-rows: minmax(50px, auto);  
}
```

Auto-created rows will be minimum 50px in height and maximum auto (=to fit content)

Article 1	Article 2 Lorem ipsum dolor sit amet consectetur adipisicing elit. Optio quaerat inventore doloremque ipsa. Nemo quas, ea modi iure impedit non ratione aliquam, ducimus nihil, est nulla. Fugit reprehenderit accusantium aliquam.	Article 3
Article 4	Article 5	

Grid lines

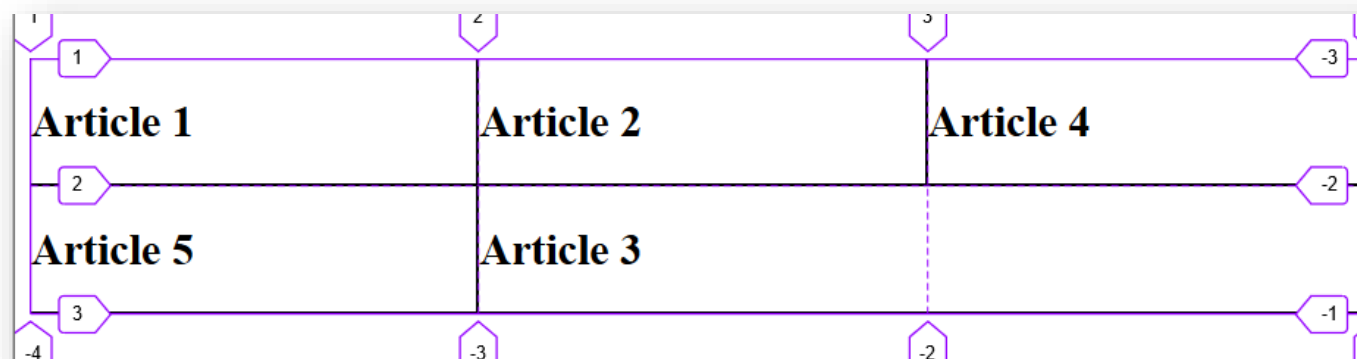
- When defining a grid, grid tracks are instantiated, not lines
- Lines are auto-created by grid system, based on tracks
- Can easily be visualized in the Firefox developer tools



Positioning against grid lines

- Using **grid-column-start**, **grid-column-end**, **grid-row-start** and **grid-row-end**

```
<div id="container">
  <article id="art1"><h2>Article 1</h2></article>
  <article id="art2"><h2>Article 2</h2></article>
  <article id="art3"><h2>Article 3</h2></article>
  <article id="art4"><h2>Article 4</h2></article>
  <article id="art5"><h2>Article 5</h2></article>
</div>
```



```
#art3 {
  grid-row-start: 2;
  grid-row-end: 2;
  grid-column-start: 2;
  grid-column-end: 4;
}
```

Positioning against grid lines

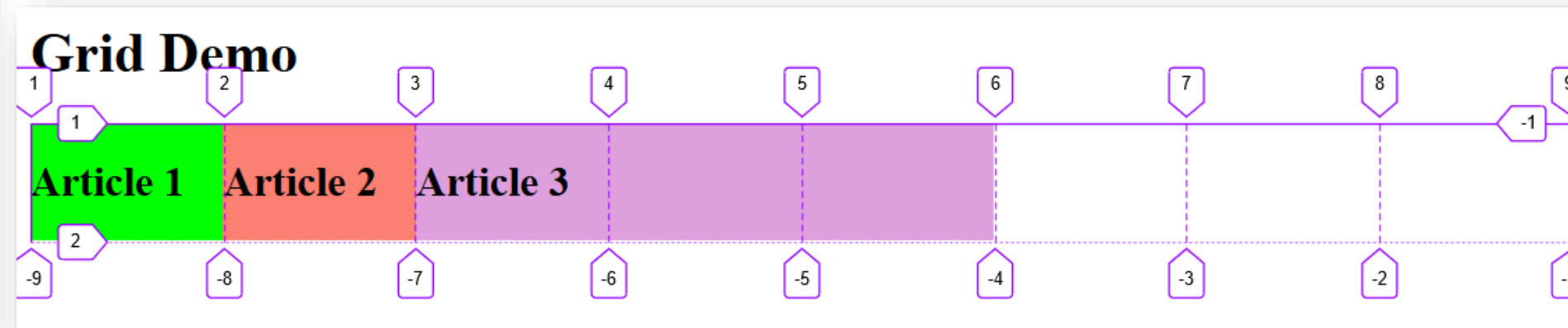
- Using **span**

```
<div id="container">
  <article id="art1"><h2>Article 1</h2></article>
  <article id="art2"><h2>Article 2</h2></article>
  <article id="art3"><h2>Article 3</h2></article>
</div>
```

```
#art1 {
  grid-area: 1 / 1 / 1 / 2;
  background-color: lime;
}

#art2 {
  grid-area: 1 / 2 / 1 / 3;
  background-color: salmon;
}

#art3 {
  grid-area: 1 / 3 / 1 / span 3;
  background-color: plum;
}
```



Positioning against grid lines

- Demo: creating a layout for a basic web page: header, main, aside and footer

```
<div id="container">
  <header>
    <h1>Grid Demo</h1>
  </header>
  <main>
    <h2>Welcome to Web, Mobile and Security</h2>
    <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Repellat
  </main>
  <aside>
    <h3>Laravel Rocks!</h3>
    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ea unde di
  </aside>
  <footer>
    &copy; Howest University of Applied Sciences
  </footer>
</div>
```


Positioning against grid lines

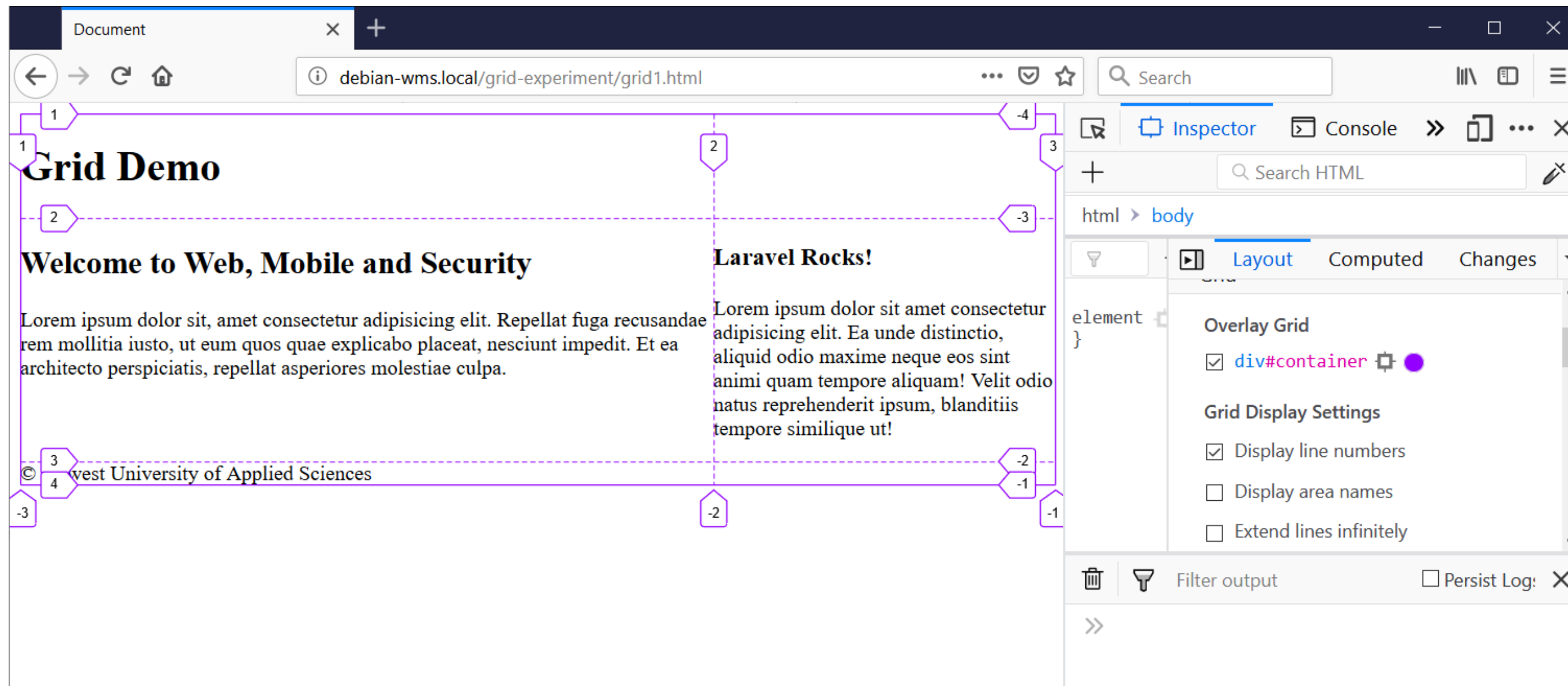
- Demo: creating a layout for a basic web page: header, main, aside and footer

```
div#container {  
  display: grid;  
  grid-template-rows: auto auto auto;  
  grid-template-columns: 67% 33%;  
}  
  
header {  
  grid-column-start: 1;  
  grid-column-end: 3;  
  grid-row-start: 1;  
  grid-row-end: 2;  
}  
  
main {  
  grid-column-start: 1;  
  grid-column-end: 2;  
  grid-row-start: 2;  
  grid-row-end: 3;  
}
```

```
aside {  
  grid-column-start: 2;  
  grid-column-end: 3;  
  grid-row-start: 2;  
  grid-row-end: 3;  
}  
  
footer {  
  grid-column-start: 1;  
  grid-column-end: 3;  
  grid-row-start: 3;  
  grid-row-end: 4;  
}
```



Positioning against grid lines


- Demo: creating a layout for a basic web page: header, main, aside and footer



grid-area: shorthand property

- Shorthand for grid-row-start, grid-column-start, grid-row-end, grid-column-end:

```
#art5 {           grs  gcs  gre  gce
  grid-area: 2 / 2 / 4 / 4;
  background-color:  rgba(255, 255, 0, 0.75);
  z-index: 2;
}

#art7 {           grs  gcs  gre  gce
  grid-area: 3 / 3 / 4 / 4;
  background-color:  lightsalmon;
  z-index: 1;
}
```

Naming lines

- Instead of working with number indices, use self-provided names

```
div#container {  
  display: grid;  
  grid-template-rows: [header-start] auto [header-end main-start]  
                      auto [main-end footer-start] auto [footer-end];  
  grid-template-columns: [main-start] 2fr [main-end aside-start]  
                        1fr [aside-end];  
}
```

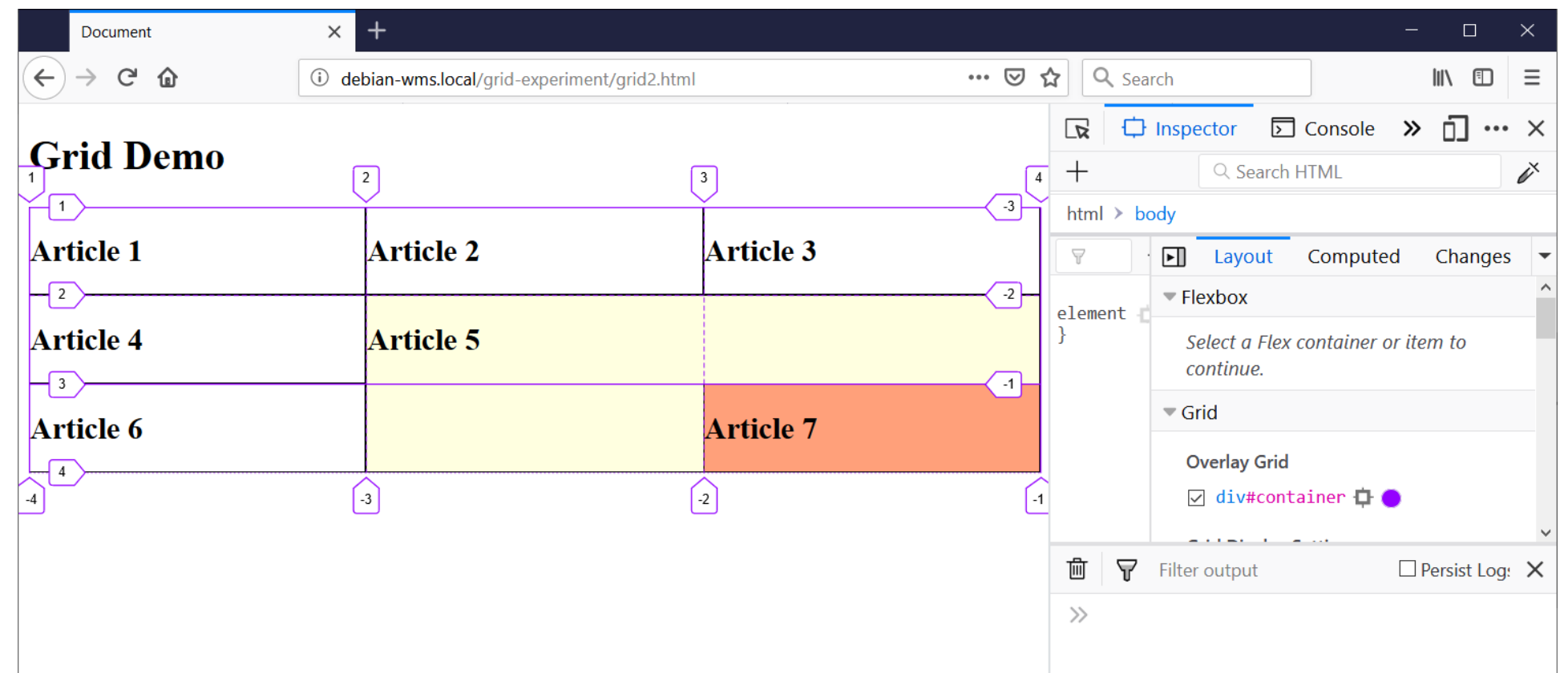
```
header {  
  grid-row-start: header-start;  
  grid-row-end: header-end;  
  grid-column-start: main-start;  
  grid-column-end: aside-end;  
  background: plum;  
}
```



Layering items

- Grid items can occupy the same cell

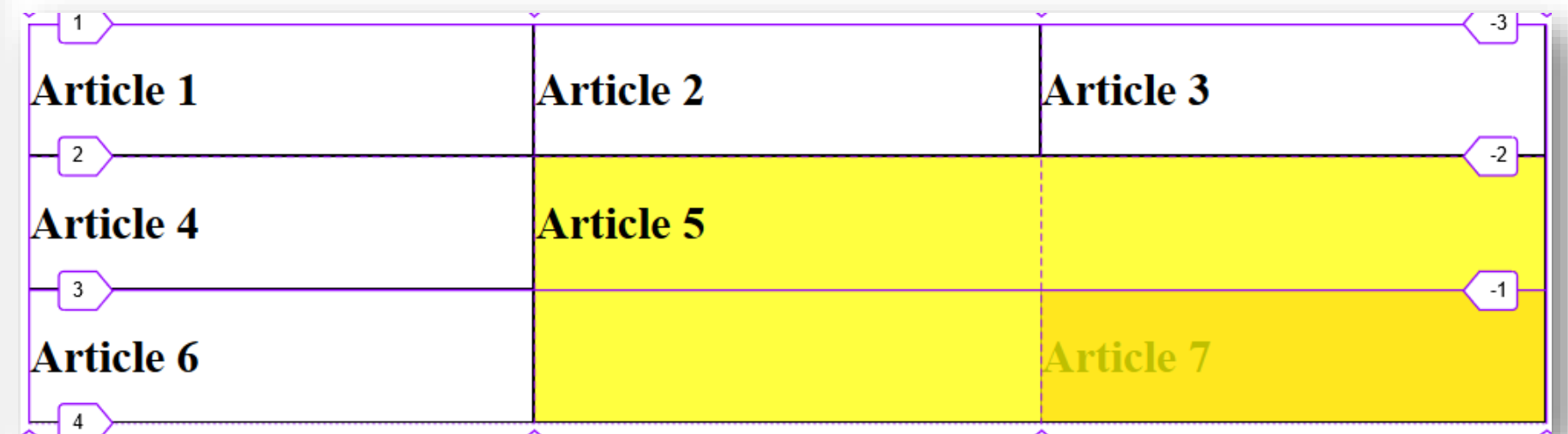
```
#art5 {  
  grid-row-start: 2;  
  grid-row-end: 4;  
  grid-column-start: 2;  
  grid-column-end: 4;  
  background-color: lightyellow;  
}  
  
#art7 {  
  grid-row-start: 3;  
  grid-row-end: 4;  
  grid-column-start: 3;  
  grid-column-end: 4;  
  background-color: lightsalmon;  
}
```



Layering items

- To change the overlap, use z-index:

```
#art5 {  
  grid-row-start: 2;  
  grid-row-end: 4;  
  grid-column-start: 2;  
  grid-column-end: 4;  
  background-color: yellow;   
  z-index: 2;  
}  
  
#art7 {  
  grid-row-start: 3;  
  grid-row-end: 4;  
  grid-column-start: 3;  
  grid-column-end: 4;  
  background-color: lightsalmon;  
  z-index: 1;  
}
```



Named grid areas

- Alternative to positioning using row and column lines
- Step 1: name the areas of your (preferably) semantic HTML layout using the **grid-area** property

```
header {  
  grid-area: hd;  
  background-color: lime;  
}  
  
main {  
  grid-area: mn;  
  background-color: lightyellow;  
}  
  
aside {  
  grid-area: as;  
  background-color: plum;  
}  
  
footer {  
  grid-area: ft;  
  background-color: salmon;  
}
```

Named grid areas

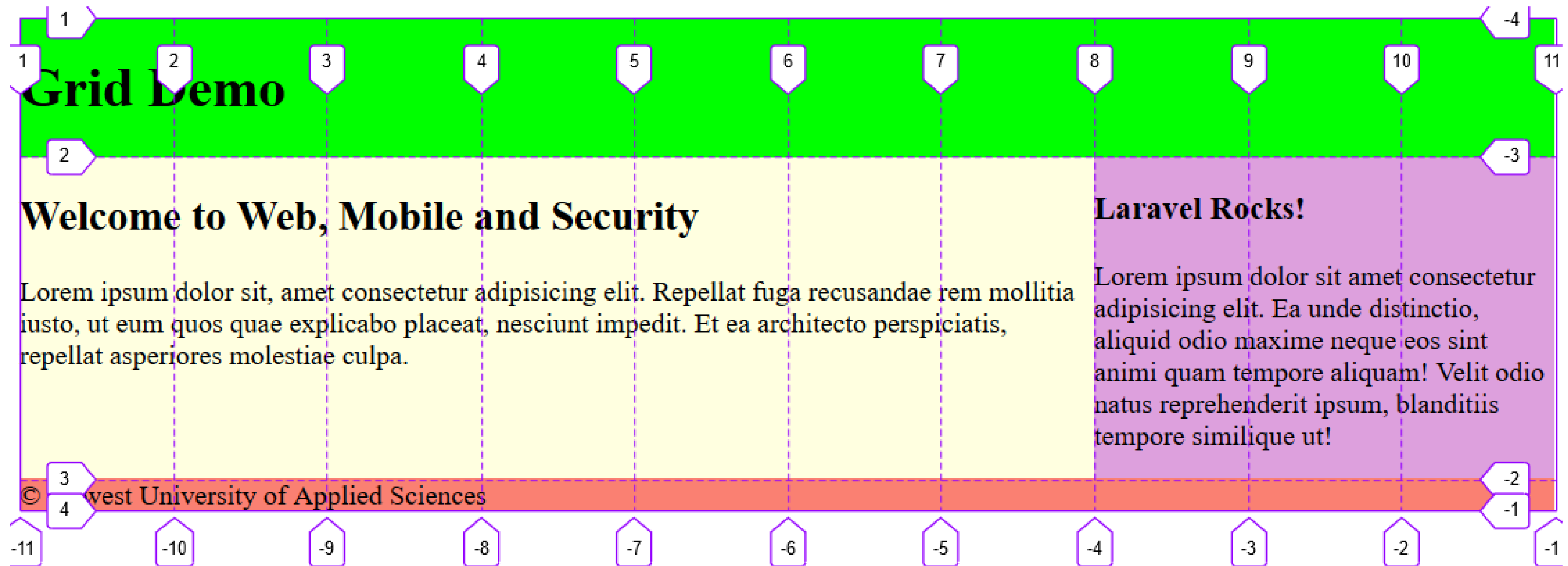
- Step 2: specify row per row, column per column
- Reference the areas using the names provided in step 1

```
div#container {  
  display: grid;  
  grid-template-rows: auto auto auto;  
  grid-template-columns: repeat(10, 1fr);  
  grid-template-areas:  
    "hd hd hd hd hd hd hd hd hd hd"  
    "mn mn mn mn mn mn mn as as as"  
    "ft ft ft ft ft ft ft ft ft ft"  
}
```

row per row,
column per
column

Named grid areas

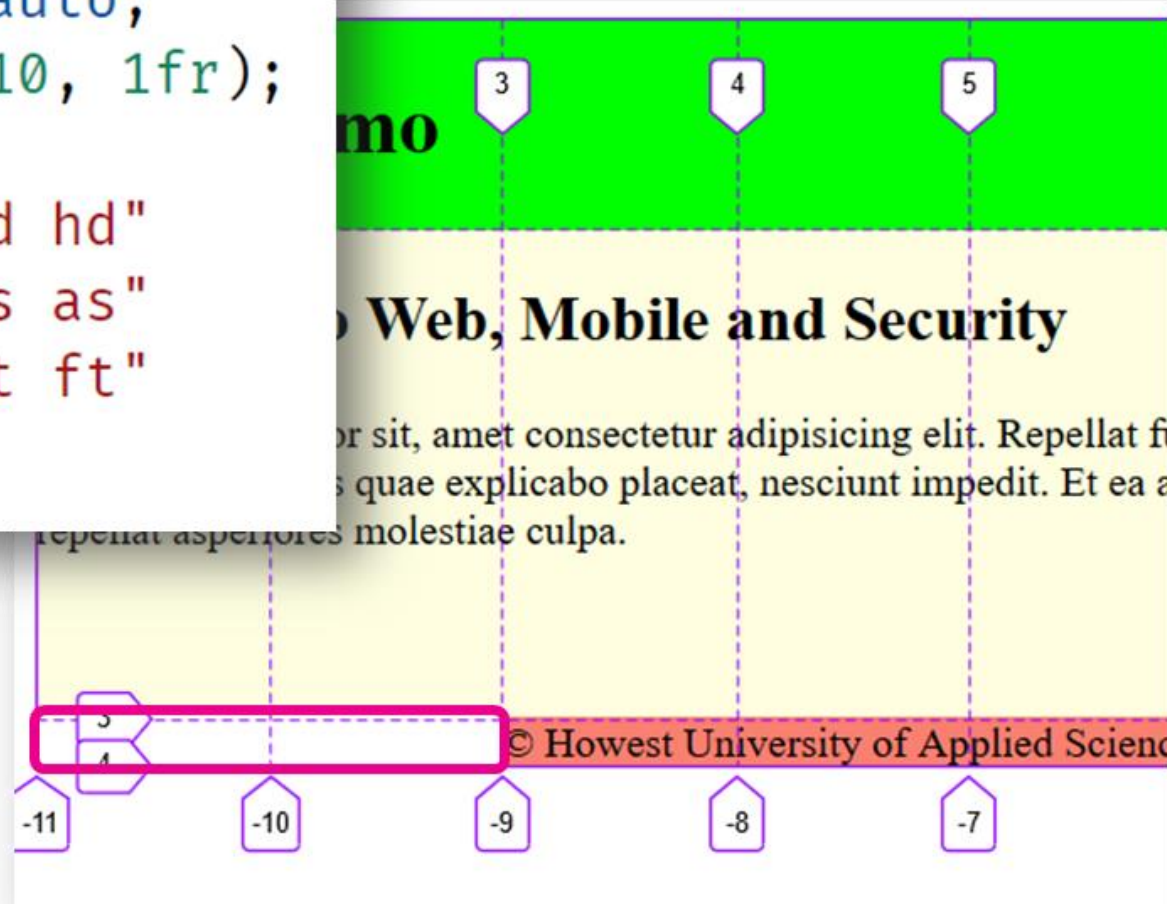
- Result:



Named grid areas

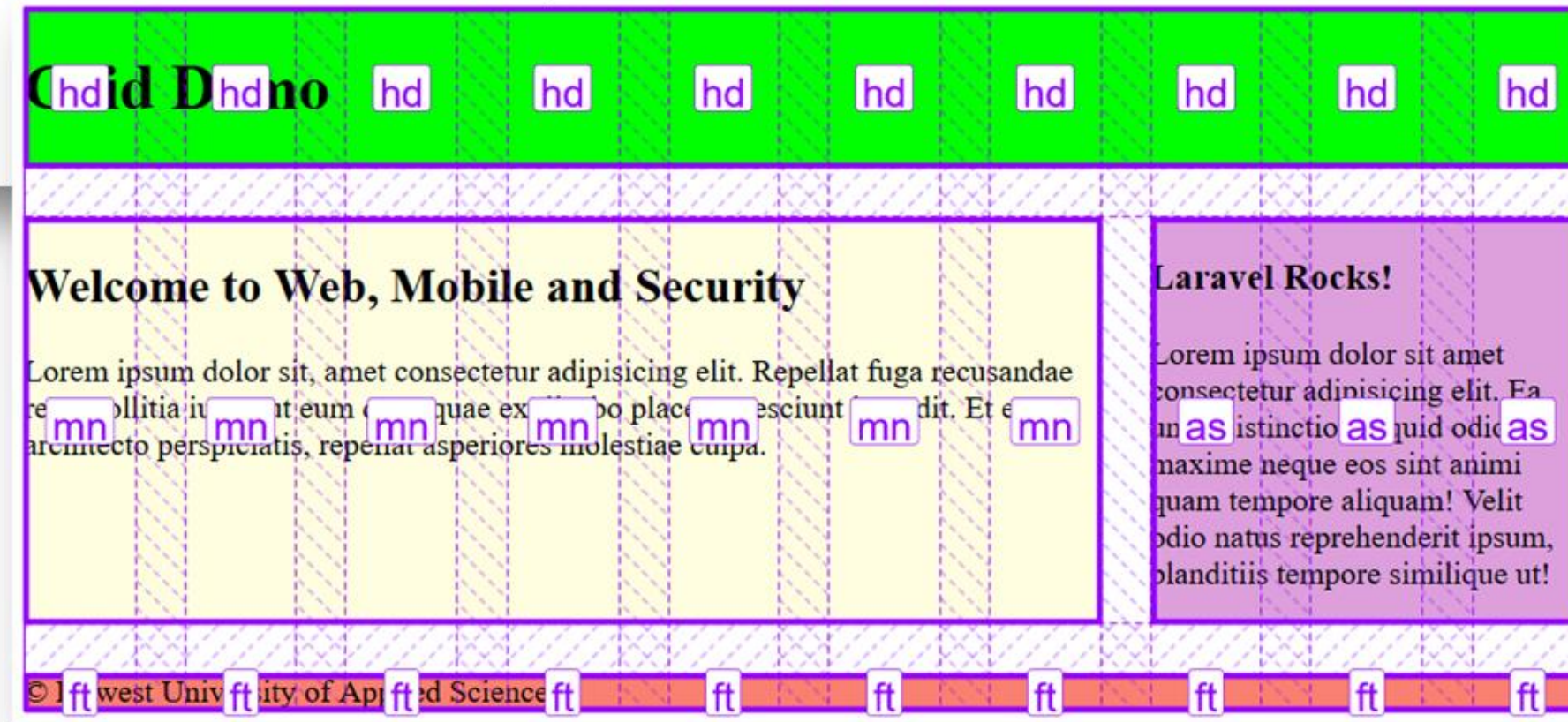
- Leaving cells empty → use the . (dot)

```
div#container {  
  display: grid;  
  grid-template-rows: auto auto auto;  
  grid-template-columns: repeat(10, 1fr);  
  grid-template-areas:  
    "hd hd hd hd hd hd hd hd hd hd"  
    "mn mn mn mn mn mn mn as as as"  
    ". . ft ft ft ft ft ft ft ft"  
}
```



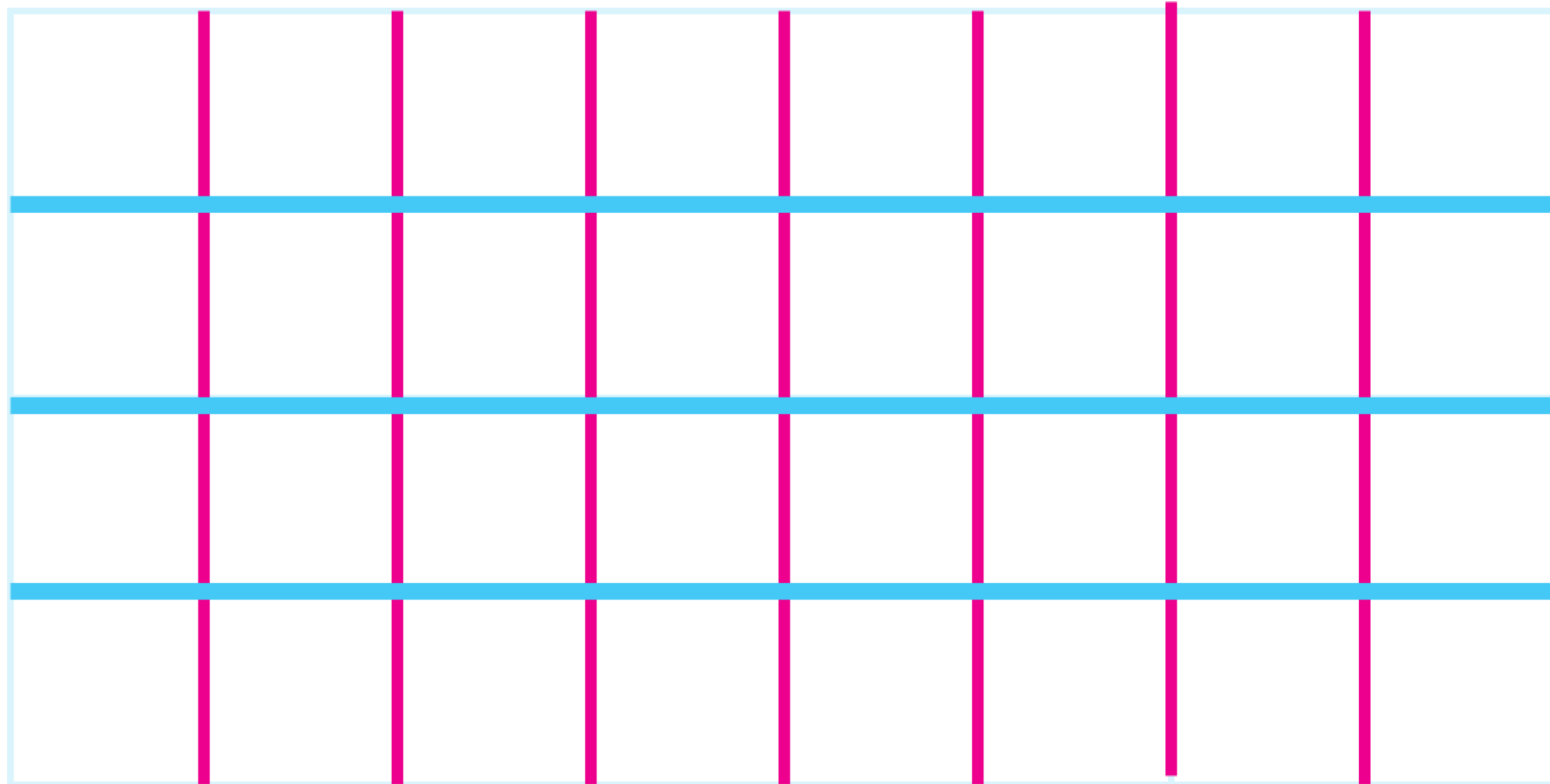
Row/column gaps

```
div#container {  
  display: grid;  
  grid-template-rows: auto auto auto;  
  grid-template-columns: repeat(10, 1fr);  
  grid-template-areas:  
    "hd hd hd hd hd hd hd hd hd hd"  
    "mn mn mn mn mn mn mn as as as"  
    "ft ft ft ft ft ft ft ft ft ft ft";  
  row-gap: 25px;  
  column-gap: 25px;  
}
```



Box alignment (cfr. Flexbox)

- Grid has 2 axes: **block axis (Y)** **inline/row axis (X)**



Aligning items on block axis (I)

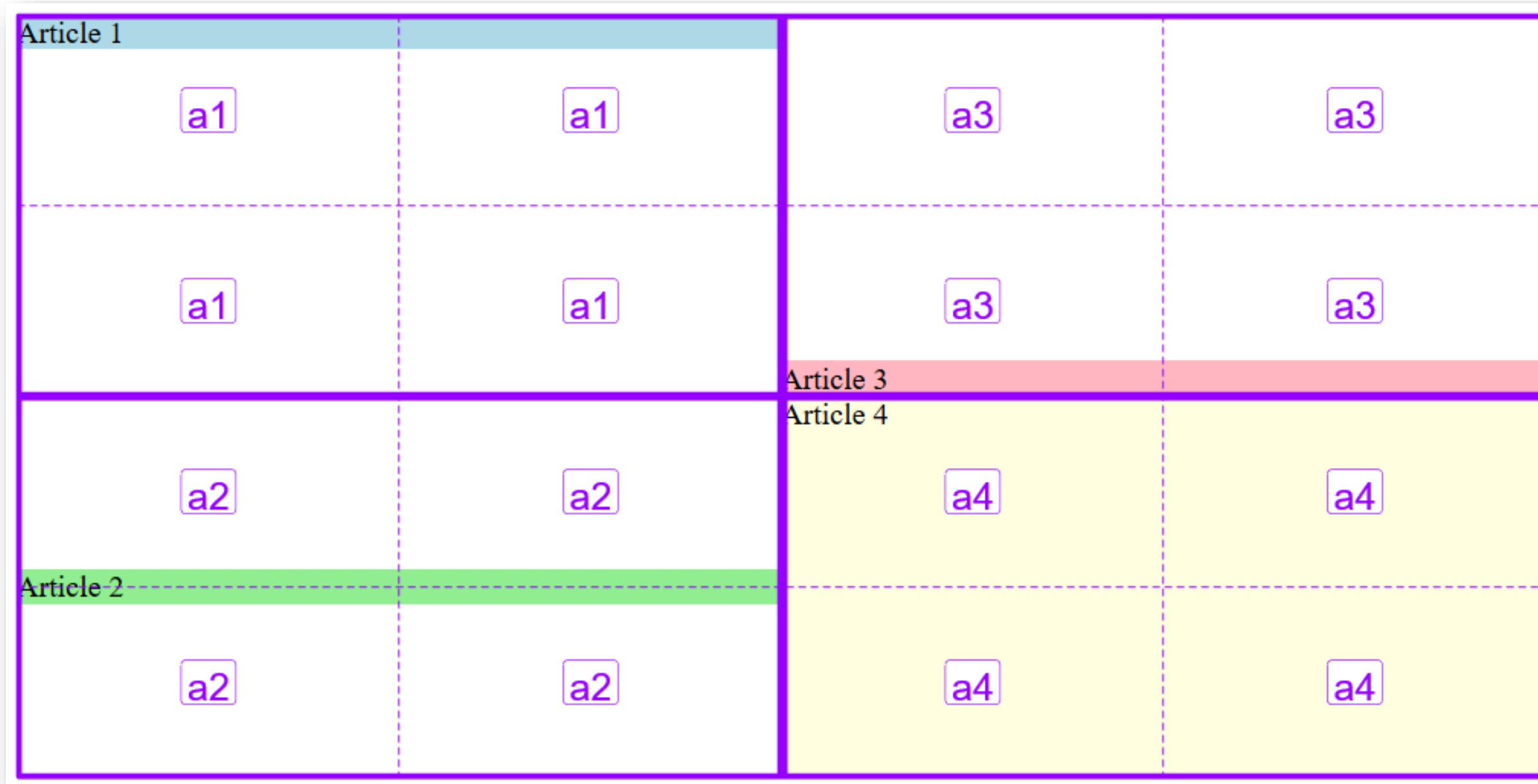
- Use **align-self** (and **align-items**)

```
#container {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  grid-template-rows: repeat(4, 100px);  
  grid-template-areas:  
    "a1 a1 a3 a3"  
    "a1 a1 a3 a3"  
    "a2 a2 a4 a4"  
    "a2 a2 a4 a4";  
}
```

```
#art1 {  
  grid-area: a1;  
  background-color: lightblue;  
  align-self: start;  
}  
  
#art2 {  
  grid-area: a2;  
  background-color: lightgreen;  
  align-self: center;  
}  
  
#art3 {  
  grid-area: a3;  
  background-color: lightpink;  
  align-self: end;  
}  
  
#art4 {  
  grid-area: a4;  
  background-color: lightyellow;  
}
```

Aligning items on block axis (|)

- Use **align-self** (and **align-items**)



Aligning items on row axis (–)

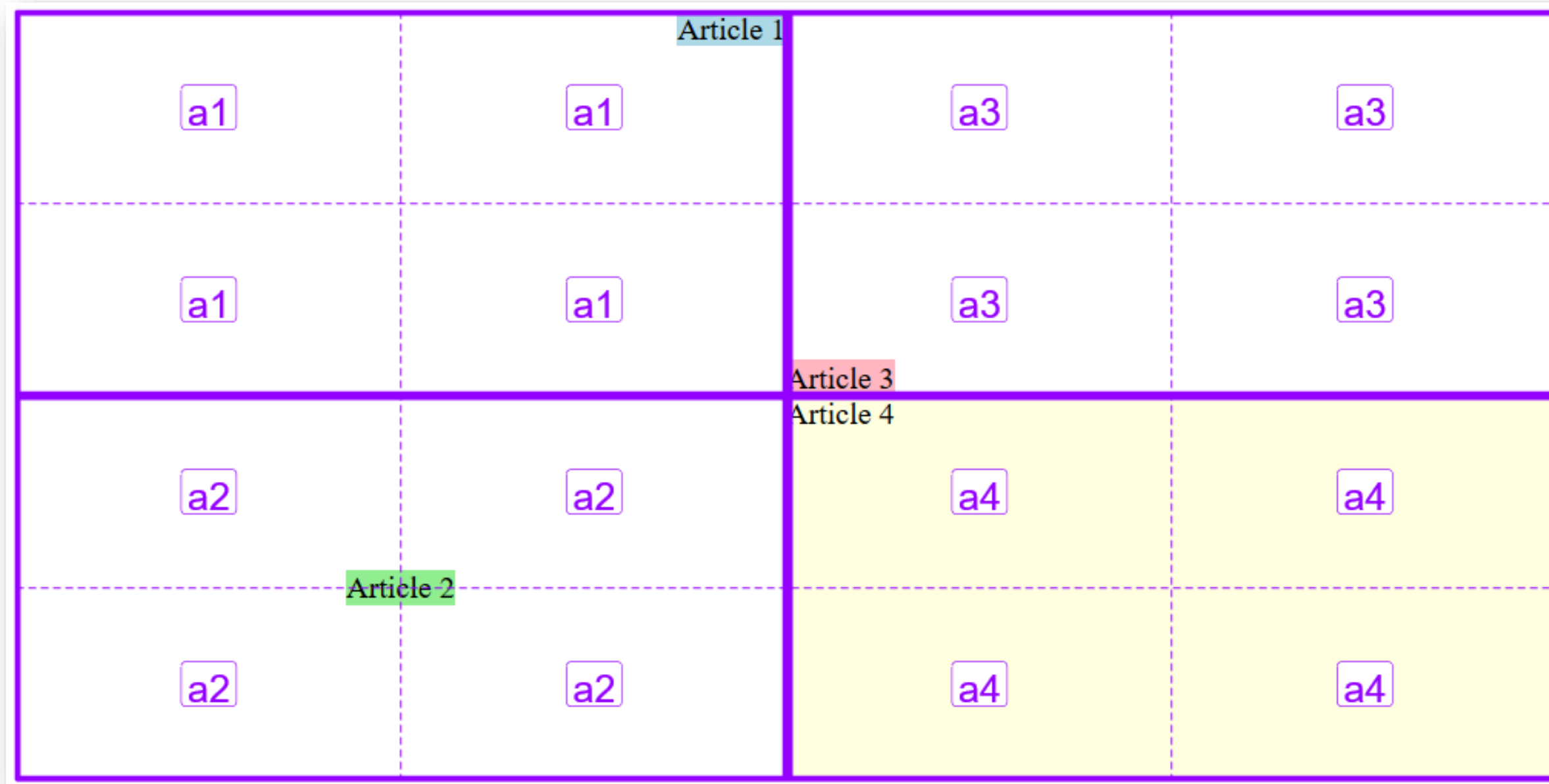
- Use **justify-self** (and **justify-items**)

```
#container {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  grid-template-rows: repeat(4, 100px);  
  grid-template-areas:  
    "a1 a1 a3 a3"  
    "a1 a1 a3 a3"  
    "a2 a2 a4 a4"  
    "a2 a2 a4 a4";  
}
```

```
#art1 {  
  grid-area: a1;  
  background-color: lightblue;  
  align-self: start;  
  justify-self: end;  
}  
  
#art2 {  
  grid-area: a2;  
  background-color: lightgreen;  
  align-self: center;  
  justify-self: center;  
}  
  
#art3 {  
  grid-area: a3;  
  background-color: lightpink;  
  align-self: end;  
  justify-self: start;  
}
```

Aligning items on row axis (–)

- Use **justify-self** (and **justify-items**)



Questions?

