CSS Grid Layout

- Vocational Training (Grau Superior).
- Web applications development.
- Second year.
- Subject – Web interfaces development.
- Unit 2 – CSS
- Teacher: Vicent Tortosa.
- Official specification:
  - https://www.w3.org/TR/css-grid-1/

Introduction to CSS Grid Layout

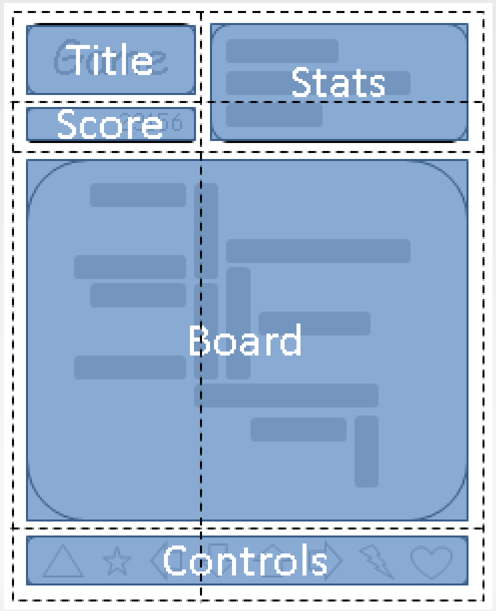
- Two dimensions layout based on grid. It comes to change how we design the css layout.
- We won’t use tables, nor floats, nor positions nor inline-blocks. All this were hacks to arrange the blocks as we wanted.
- Flexbox is easier than floats but it has only one dimension.
- Grid layout is created to solve all former problems.

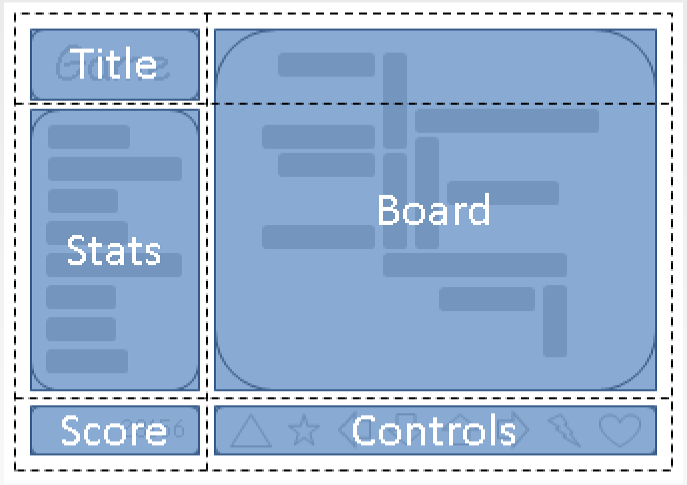
Introduction to CSS Grid Layout

- All current browsers support this specification http://caniuse.com/#feat=css-grid
- 960 Grid System was the first to implement a grid design
- On one hand, frameworks such as 960GS and Bootstrap implement many CSS rules that most of us won’t use. It makes the load of the web heavy.
- On the other hand, CSS Grid Layout is very easy to maintain and combine with other specifications such as `flex box` and `media queries`.

Introduction to CSS Grid Layout

- Both pictures show the same HTML but the applying CSS is different. Media queries are used to know the device features then apply one CSS or another. The first one shows the web in portrait mode and the second one shows the web in landscape mode.



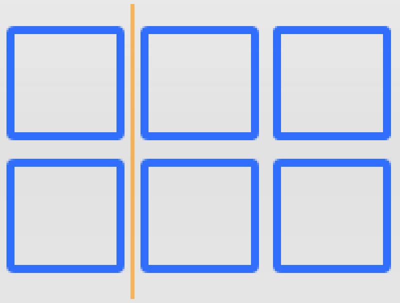


Grid layout Terminology

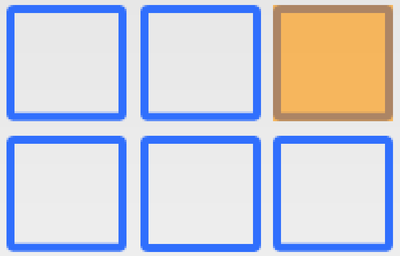
- `grid container`: The container that holds the entire CSS grid. It will be the element that has the `display: grid` or `display: inline-grid` property on it.
- `grid item`: Any element that is a direct child of a grid container. The child of the child won’t become grid item.
- `fr`: Unit of measure. MDN defines the *fr* unit as a unit which represents a fraction of the available space in the grid container.

Grid layout Terminology

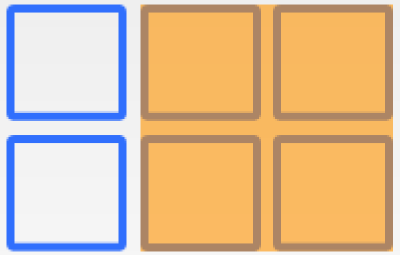
- `Grid line` The vertical and horizontal lines that divide the grid and separate the columns and rows.



- `Grid cell`. A single unit of a CSS grid.

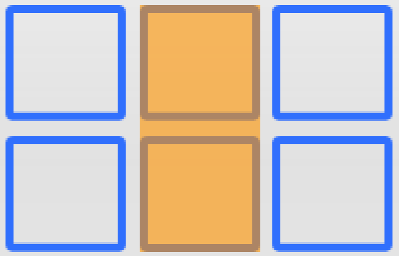


- `Grid area.` Rectangular space surrounded by four grid lines. A grid area can contain any number of grid cells.

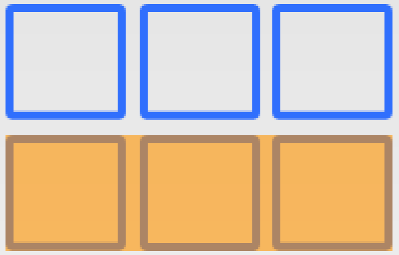


Grid layout Terminology

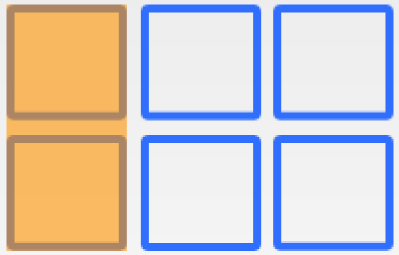
- `Grid track` The space between two grid lines. This space can be horizontal or vertical



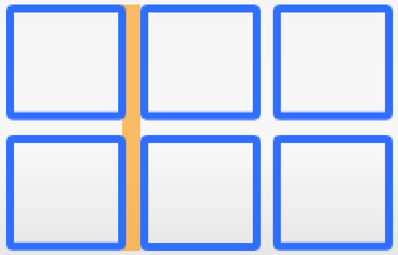
- `Grid row` A horizontal track of a grid.



- `Grid column` A vertical track of a grid.

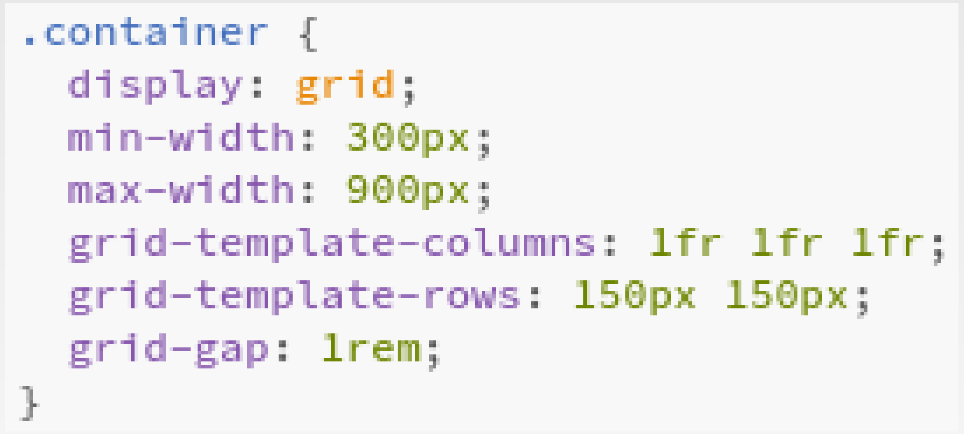


- `Gutter` The space between rows and columns in a grid.

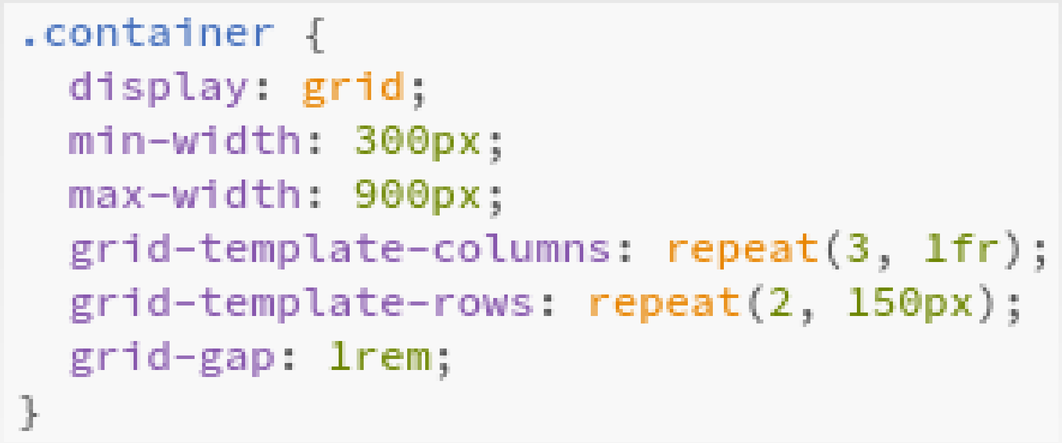


Your first Grid

1) Create a grid: `display: grid`
2) Define rows and columns
3) Add a gutter: `grid-gap: 1rem;`



```css
.container {
  display: grid;
  min-width: 300px;
  max-width: 900px;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 150px 150px;
  grid-gap: 1rem;
}
```

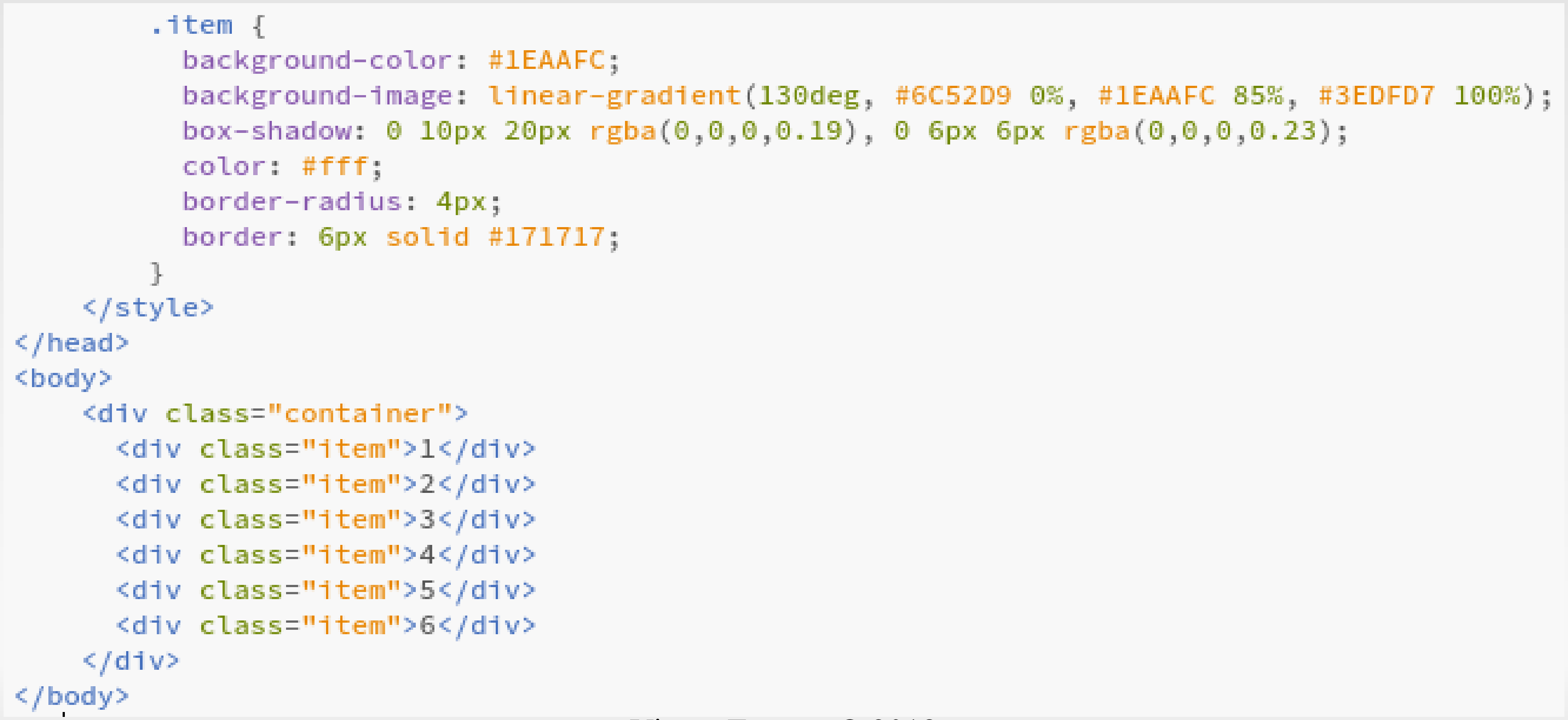


```css
.container {
  display: grid;
  min-width: 300px;
  max-width: 900px;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(2, 150px);
  grid-gap: 1rem;
}
```

(both configurations have the same effect)

Your first Grid II

- Let’s give some style to the *grid items* and the HTML.
- Se the result in 8_Tu_primer_GRID_Layout.html



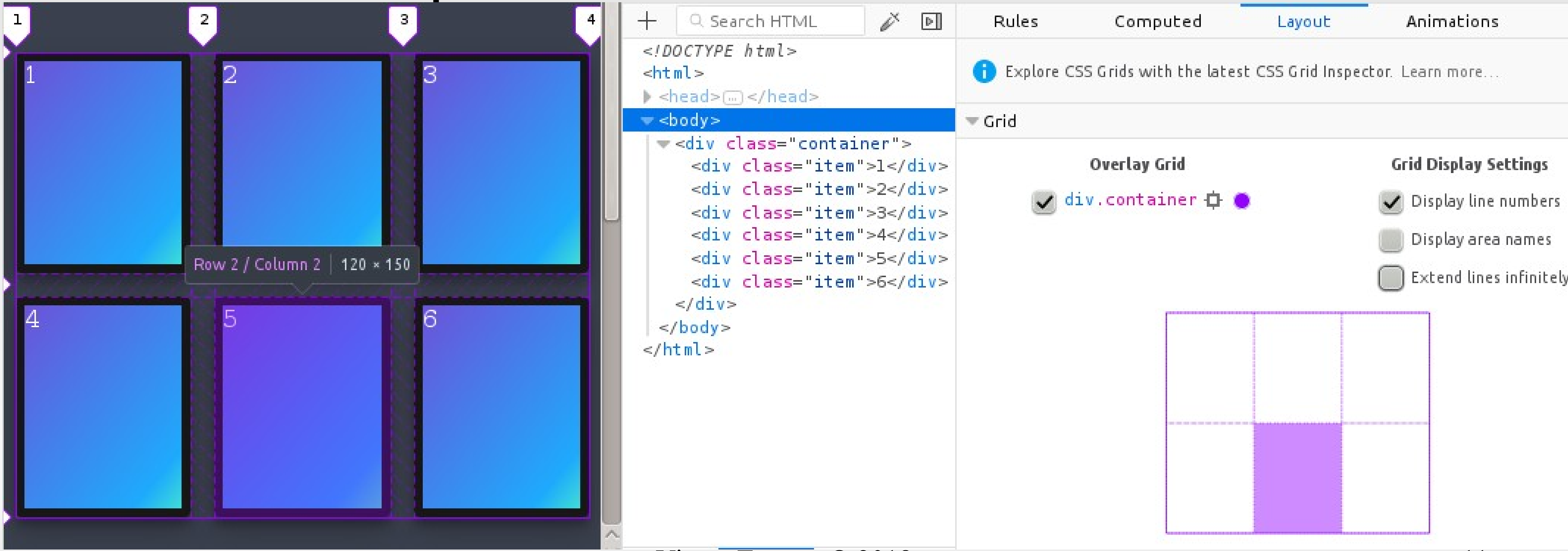
```
        .item {
          background-color: #1EAAFC;
          background-image: linear-gradient(130deg, #6C52D9 0%, #1EAAFC 85%, #3EDFD7 100%);
          box-shadow: 0 10px 20px rgba(0,0,0,0.19), 0 6px 6px rgba(0,0,0,0.23);
          color: #fff;
          border-radius: 4px;
          border: 6px solid #171717;
        }
    </style>
</head>
<body>
    <div class="container">
      <div class="item">1</div>
      <div class="item">2</div>
      <div class="item">3</div>
      <div class="item">4</div>
      <div class="item">5</div>
      <div class="item">6</div>
    </div>
</body>
```

Firefox Quantum: Developer Edition

- Before going on, it's a good idea to be familiar with a tool that will facilitate analysis and work with Grid layout.
- If in Firefox Developer Edition we activate the Inspect Element mode and in the right section we go to Layout, we will see that if we select the box that is the grid container, a grid is drawn that identifies our grid.

Firefox Quantum – Grid Overlay and Interactive Grid

- We can select the grid, the line numbers and the dimensions. Mouse over the outline to highlight parts of the grid on the pages and display size, area, and position information.


1
2
3
4
1
2
3
Row 2 / Column 2 | 120 × 150
4
5
6
Search HTML
Rules
Computed
Layout
Animations
Explore CSS Grids with the latest CSS Grid Inspector. Learn more…
Grid
Overlay Grid
Grid Display Settings

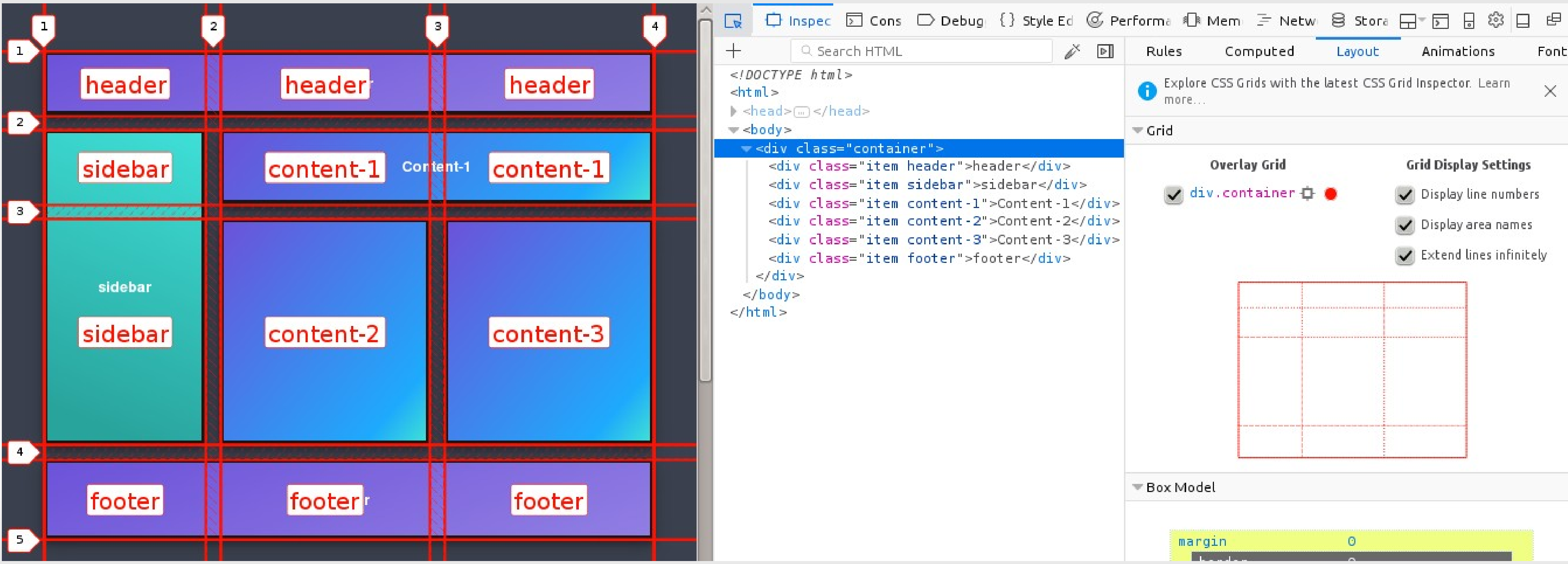
```
<!DOCTYPE html>
<html>
<head>…</head>
<body>
  <div class="container">
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
    <div class="item">4</div>
    <div class="item">5</div>
    <div class="item">6</div>
  </div>
</body>
</html>
```

✓ div.container ●

✓ Display line numbers
Display area names
Extend lines infinitely

Firefox Quantum – Display Grid Area

- “Display area names” shows the bounding areas and the associated area name in every cell. We'll learn more about how to set a grid area name in a bit.



```
<!DOCTYPE html>
<html>
  <head>…</head>
  <body>
    <div class="container">
      <div class="item header">header</div>
      <div class="item sidebar">sidebar</div>
      <div class="item content-1">Content-1</div>
      <div class="item content-2">Content-2</div>
      <div class="item content-3">Content-3</div>
      <div class="item footer">footer</div>
    </div>
  </body>
</html>
```

Explore CSS Grids with the latest CSS Grid Inspector. Learn more…

Overlay Grid

✓ div.container

Grid Display Settings

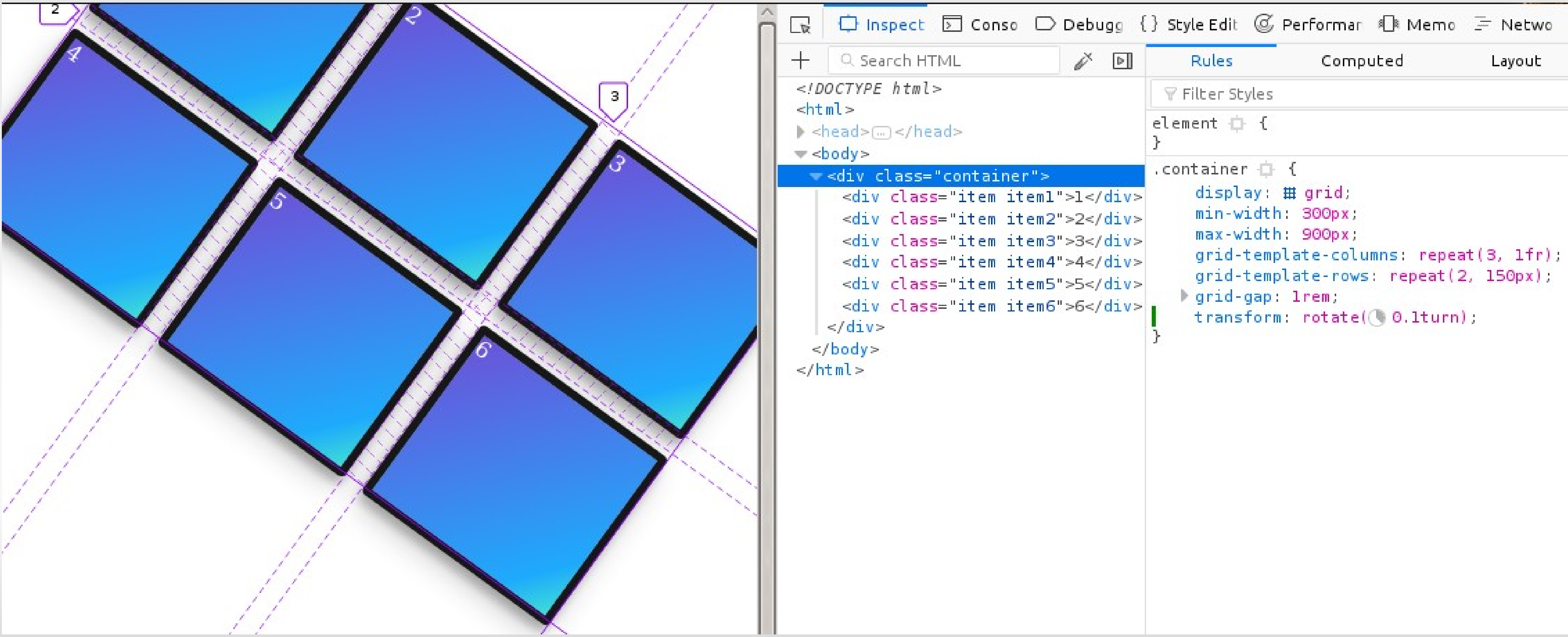
✓ Display line numbers

✓ Display area names

✓ Extend lines infinitely

Firefox Quantum – Visualize Transformations

- The Grid Inspector is capable of visualizing transformations applied to the grid container. This lets you accurately see where the grid lines are on the page for any grids that are translated, skewed, rotated, or scaled.

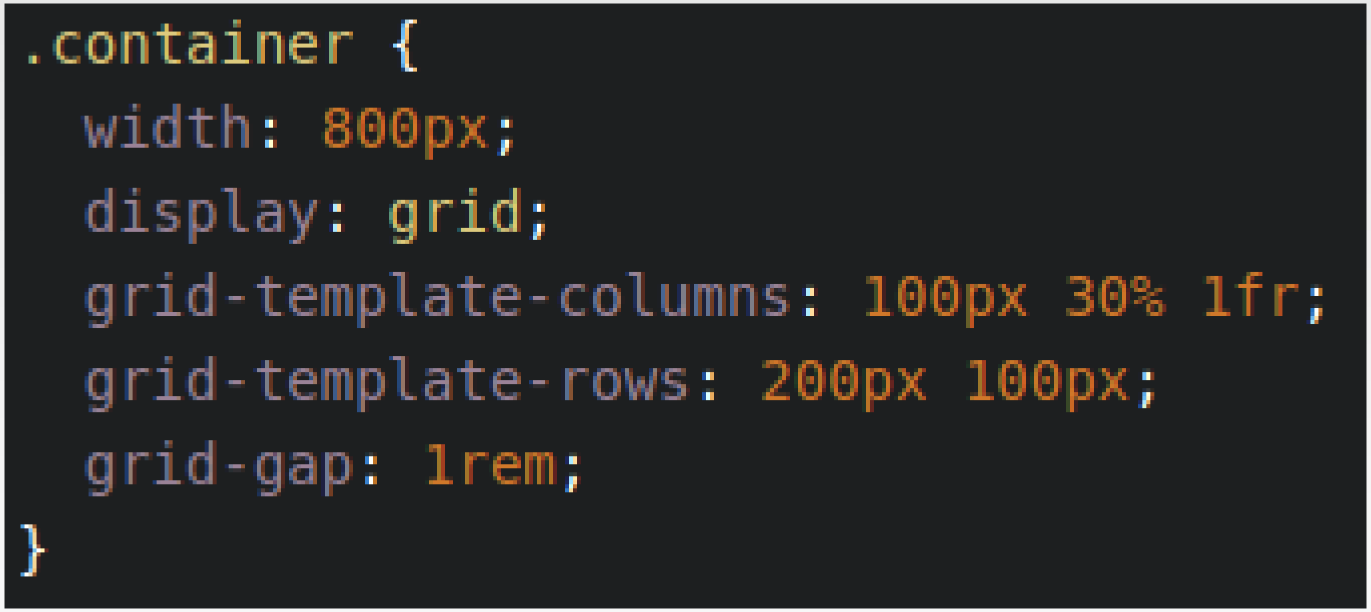


```
<!DOCTYPE html>
<html>
 <head>…</head>
 <body>
   <div class="container">
     <div class="item item1">1</div>
     <div class="item item2">2</div>
     <div class="item item3">3</div>
     <div class="item item4">4</div>
     <div class="item item5">5</div>
     <div class="item item6">6</div>
   </div>
 </body>
</html>
```

```
element {
}
.container {
    display: grid;
    min-width: 300px;
    max-width: 900px;
    grid-template-columns: repeat(3, 1fr);
    grid-template-rows: repeat(2, 150px);
  ▸ grid-gap: 1rem;
    transform: rotate(0.1turn);
}
```

Mixin units

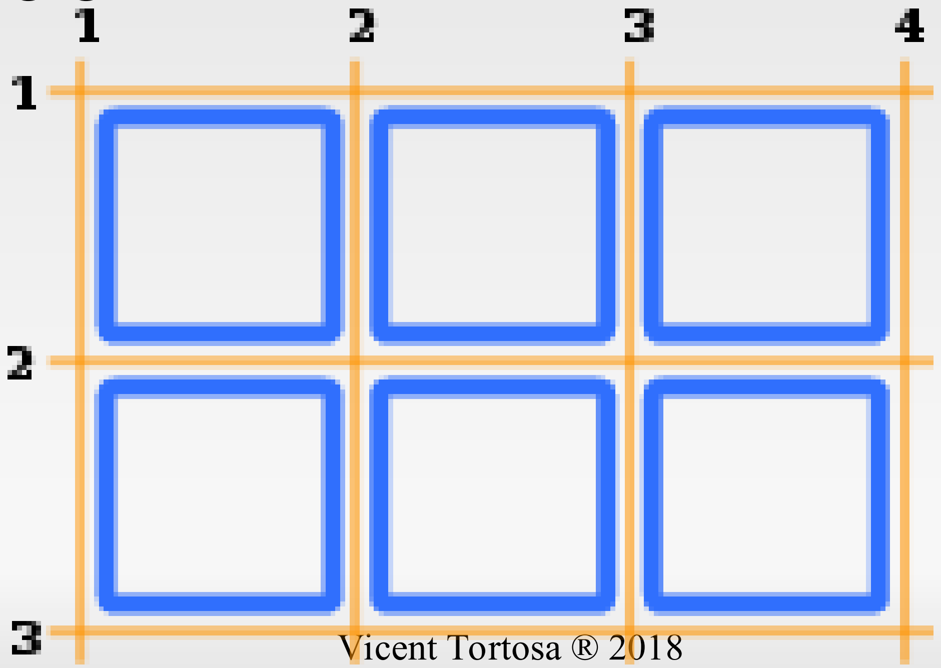
- When declaring track sizes, you can use fixed sizes with units such as *px* and *em*. You can also use flexible sizes such as percentages or the *fr* unit. The real magic of CSS Grid Layout, however, is the ability to mix these units.



```css
.container {
  width: 800px;
  display: grid;
  grid-template-columns: 100px 30% 1fr;
  grid-template-rows: 200px 100px;
  grid-gap: 1rem;
}
```

Position *items* I

- There are several ways to place items, but we will start with a basic example.
- Each item within this grid will be placed automatically in the default order.
- If we wish to have greater control, we can position items on the grid using grid line numbers.



Position *items* II

- We position the item1 in the second column in the second row. We'll use the grid lines to position it.

```
.item1 {
  grid-row-start: 2;
  grid-row-end: 3;
  grid-column-start: 2;
  grid-column-end: 3;
}
```

```
.item1 {
  grid-row: 2 / 3;
  grid-column: 2 / 3;
}
```

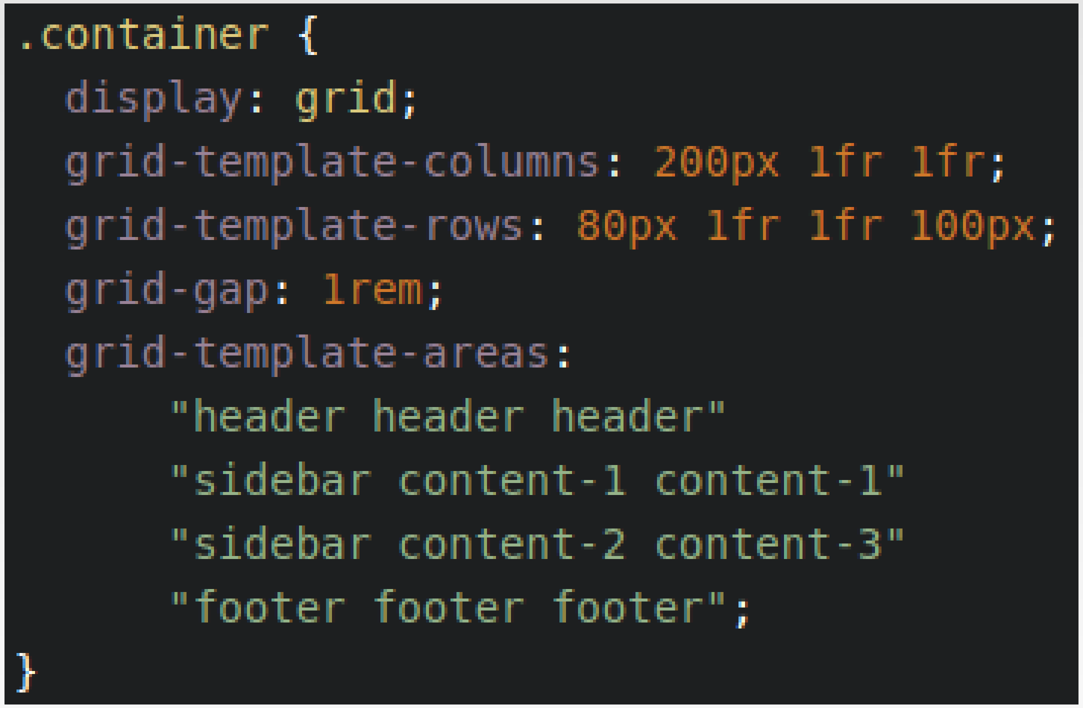
Both definitions are the same, second one is in shorthand mode.

Basic Layout

- Now you know how the CSS grid works. Is time to study some samples.
- Have a look to the code:
  - 11_GRID_Basic_Layout.html
  - 11_GRID_Basic_Layout.css

Template areas I

- Another method for positioning items is to use named grid areas with the grid-template-areas and grid-area properties. Sample:

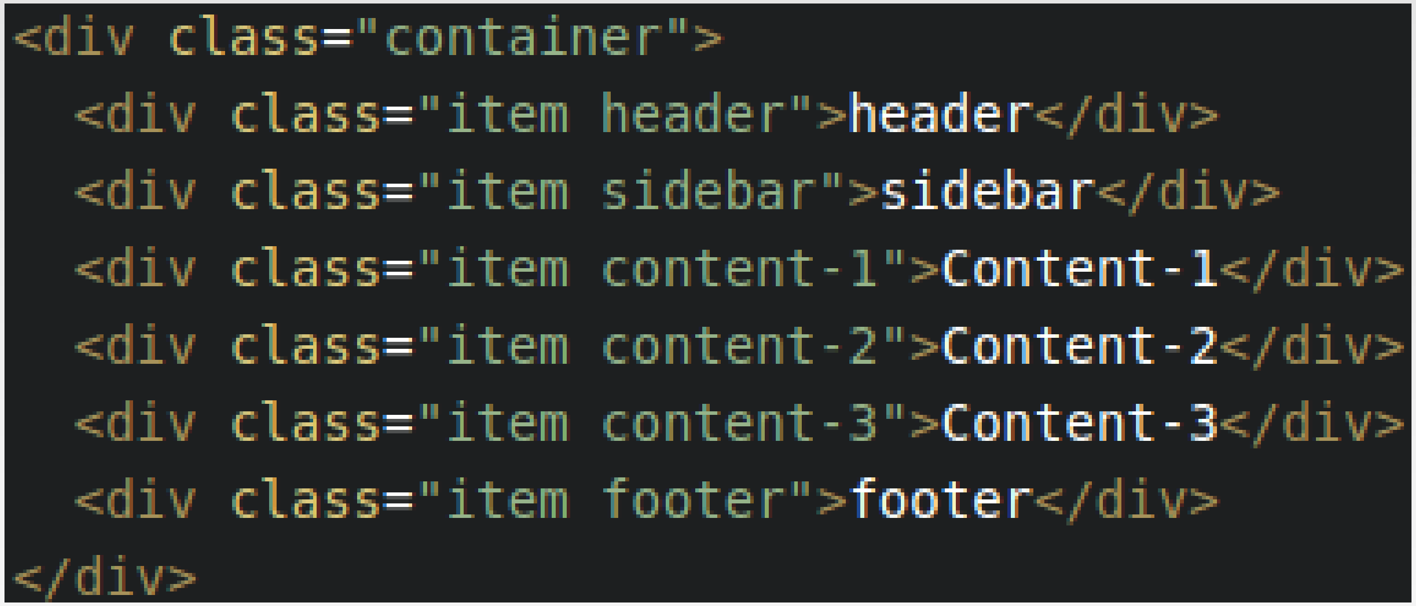


```
.container {
  display: grid;
  grid-template-columns: 200px 1fr 1fr;
  grid-template-rows: 80px 1fr 1fr 100px;
  grid-gap: 1rem;
  grid-template-areas:
      "header header header"
      "sidebar content-1 content-1"
      "sidebar content-2 content-3"
      "footer footer footer";
}
```

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

Template areas II

- Our HTML has blocks of which can have classes or not. In this example all blocks are *divs* ergo we need classes to identify them:

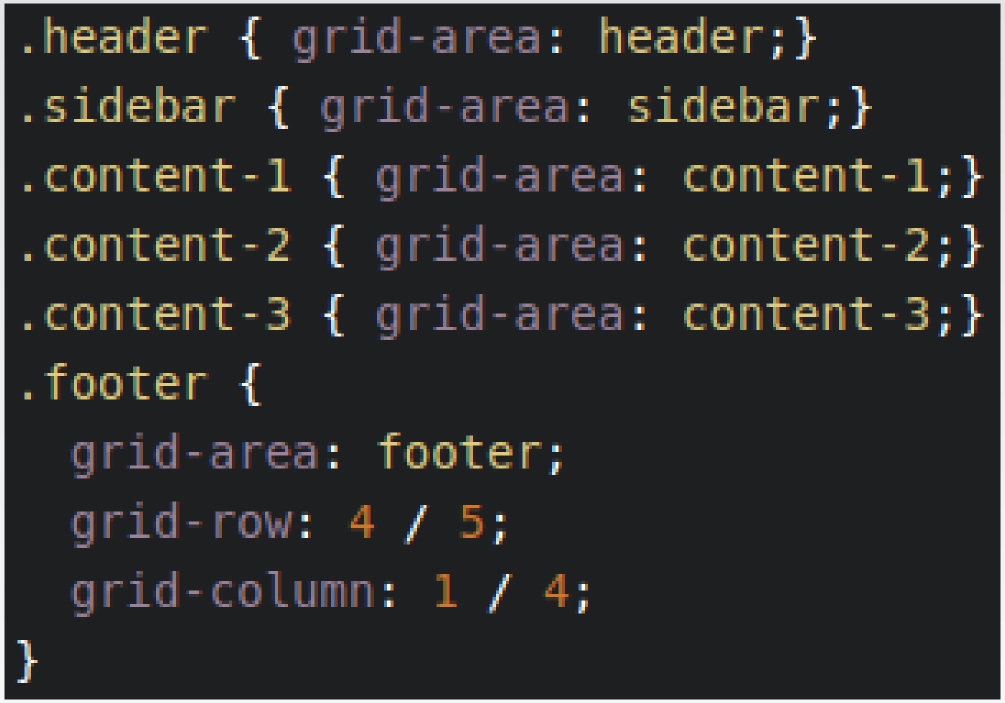


```
<div class="container">
  <div class="item header">header</div>
  <div class="item sidebar">sidebar</div>
  <div class="item content-1">Content-1</div>
  <div class="item content-2">Content-2</div>
  <div class="item content-3">Content-3</div>
  <div class="item footer">footer</div>
</div>
```

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

Template areas III

- We define classes where we can then assign the areas to each grid item using the grid-area property.

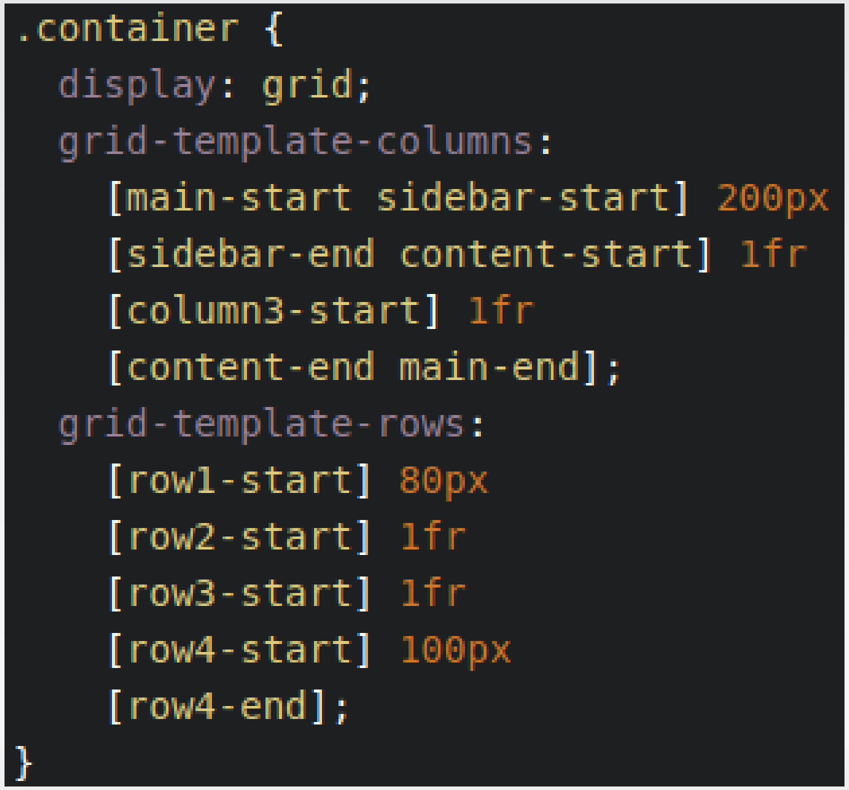


```css
.header { grid-area: header;}
.sidebar { grid-area: sidebar;}
.content-1 { grid-area: content-1;}
.content-2 { grid-area: content-2;}
.content-3 { grid-area: content-3;}
.footer {
  grid-area: footer;
  grid-row: 4 / 5;
  grid-column: 1 / 4;
}
```

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

Named lines I

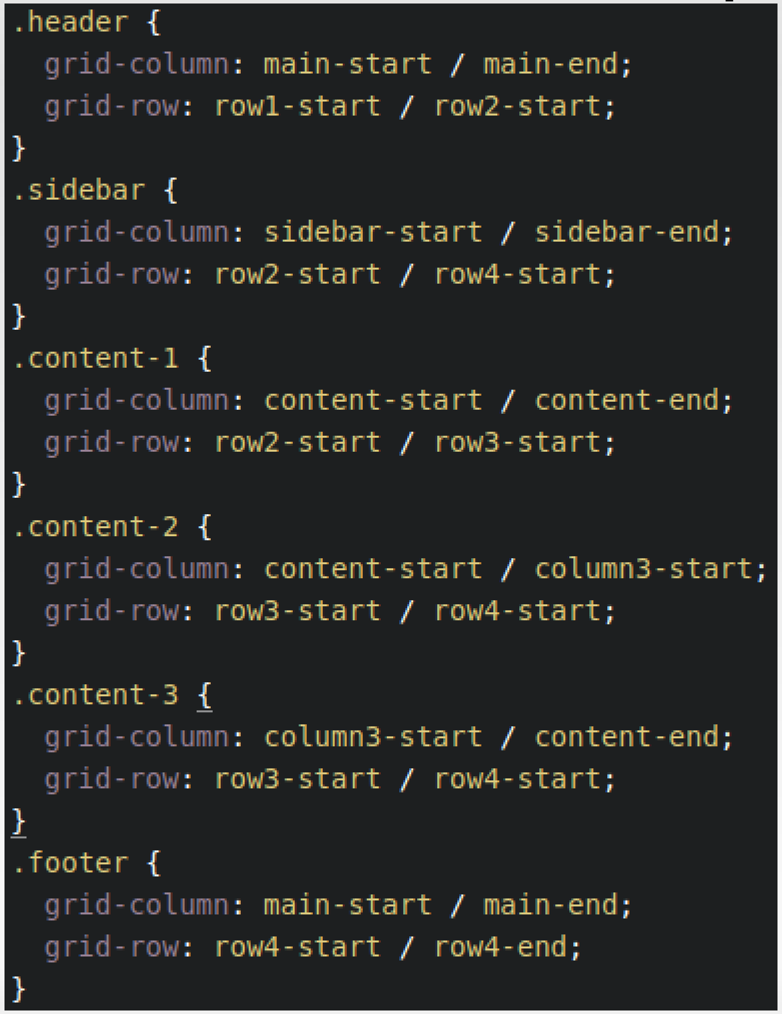
- We can also name some or all of those grid lines when defining a grid. This allows us to use those names instead of grid lines. To name a grid line, simply add the name in square brackets:



```
.container {
  display: grid;
  grid-template-columns:
    [main-start sidebar-start] 200px
    [sidebar-end content-start] 1fr
    [column3-start] 1fr
    [content-end main-end];
  grid-template-rows:
    [row1-start] 80px
    [row2-start] 1fr
    [row3-start] 1fr
    [row4-start] 100px
    [row4-end];
}
```

Named lines II

- We can use those names when placing items.



```css
.header {
  grid-column: main-start / main-end;
  grid-row: row1-start / row2-start;
}
.sidebar {
  grid-column: sidebar-start / sidebar-end;
  grid-row: row2-start / row4-start;
}
.content-1 {
  grid-column: content-start / content-end;
  grid-row: row2-start / row3-start;
}
.content-2 {
  grid-column: content-start / column3-start;
  grid-row: row3-start / row4-start;
}
.content-3 {
  grid-column: column3-start / content-end;
  grid-row: row3-start / row4-start;
}
.footer {
  grid-column: main-start / main-end;
  grid-row: row4-start / row4-end;
}
```

Examples

- Have a look to the code and the behaviour of the CSS and HTML.
- 8_Tu_primer_GRID_Layout.html
- 9_GRID_mezclando_unidades.html
- 10_GRID_posicionando_grid_items.html
- 11_GRID_Basic_Layout.html
- 12_GRID_Areas.html
- 13_GRID_Named_lines.html

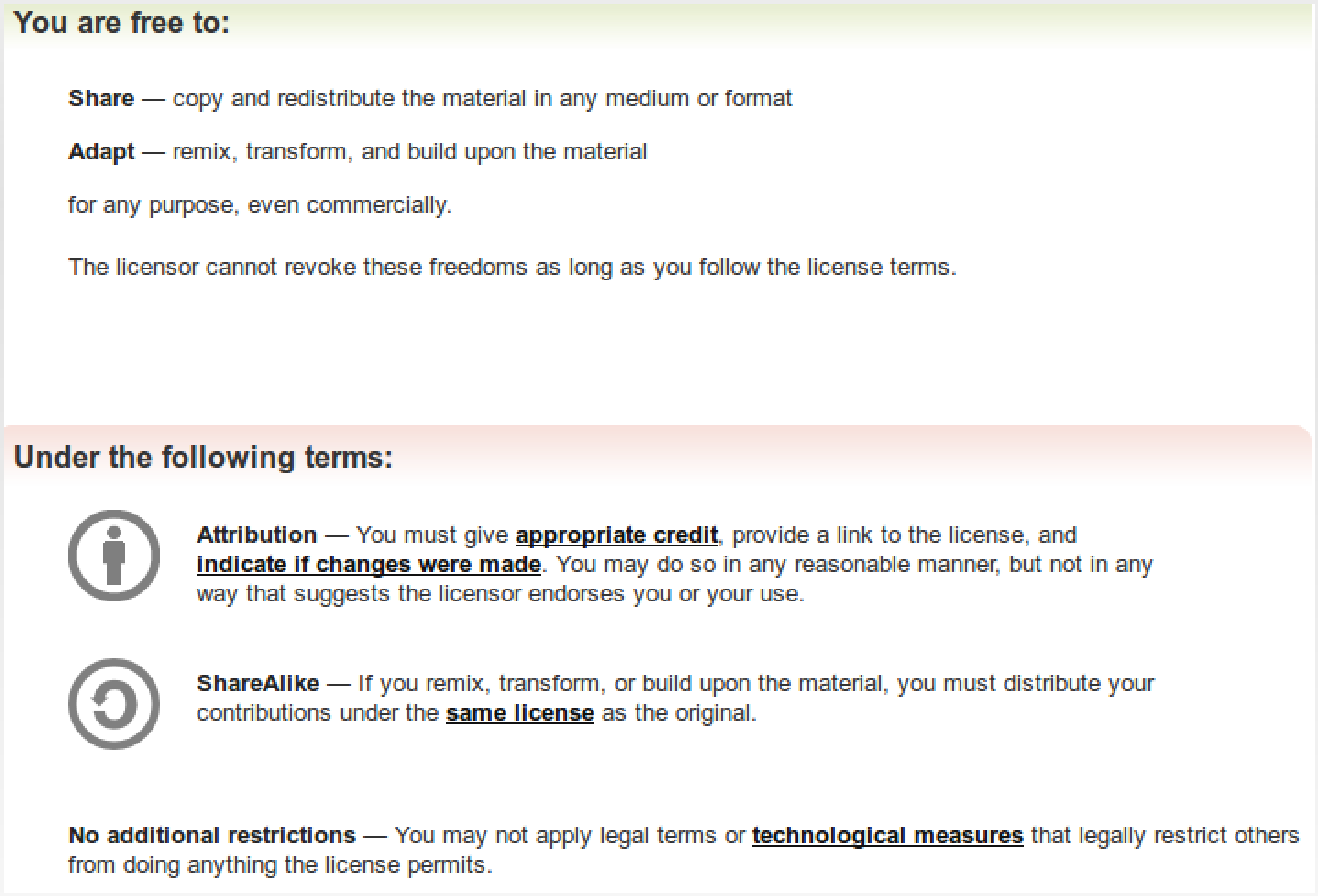
Learn more about

- This presentation is limited to studying how to make a layout, how to arrange the boxes and how to order them.
- It remains to be seen how to integrate *flexbox* and *grid layout*, in the examples you can see in the code how these two types of layouts complement each other.
- It remains to see the justification and alignment of the elements within the grid layout, which is very similar to how justification and alignment works in *flexbox*.

Webography and examples

- https://www.w3.org/TR/css-grid-1/
- https://mozilladevelopers.github.io/playground/
- https://css-tricks.com/snippets/css/complete-guide-grid/
- https://hacks.mozilla.org/2017/10/an-introduction-to-css-grid-layout-part-1/
- https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout

Creative Commons License Attribution-ShareAlike – CC BY-SA



You are free to:

Share — copy and redistribute the material in any medium or format

Adapt — remix, transform, and build upon the material

for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give **appropriate credit**, provide a link to the license, and **indicate if changes were made**. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the **same license** as the original.

No additional restrictions — You may not apply legal terms or **technological measures** that legally restrict others from doing anything the license permits.

http://creativecommons.org/licenses/by-sa/4.0/