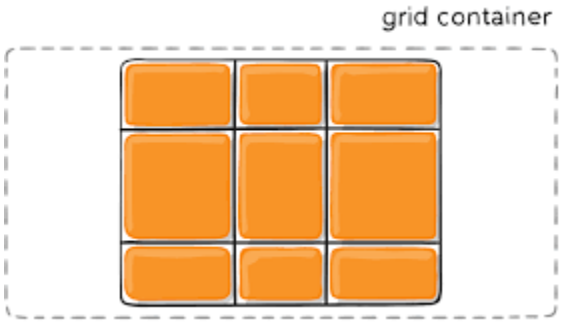
CSS Grid

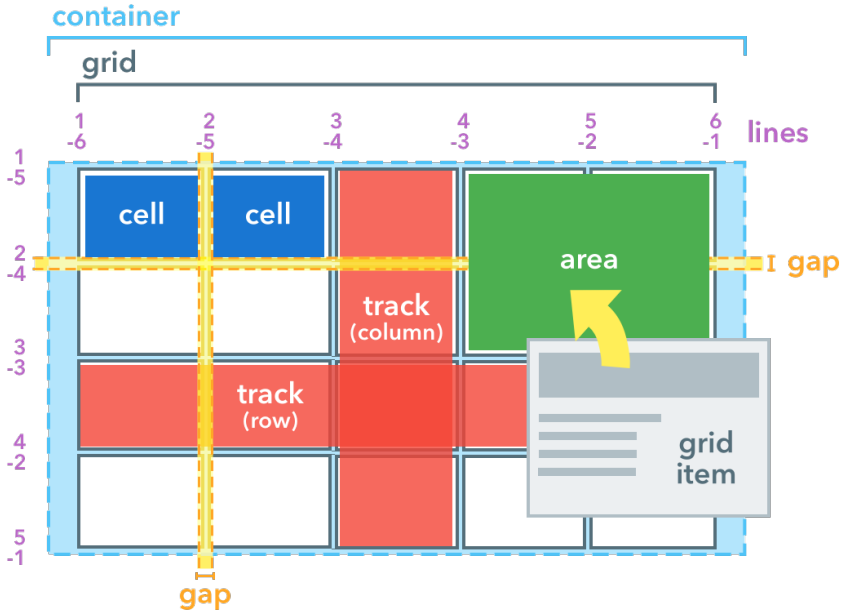
Six key terms:

1- Grid Container




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

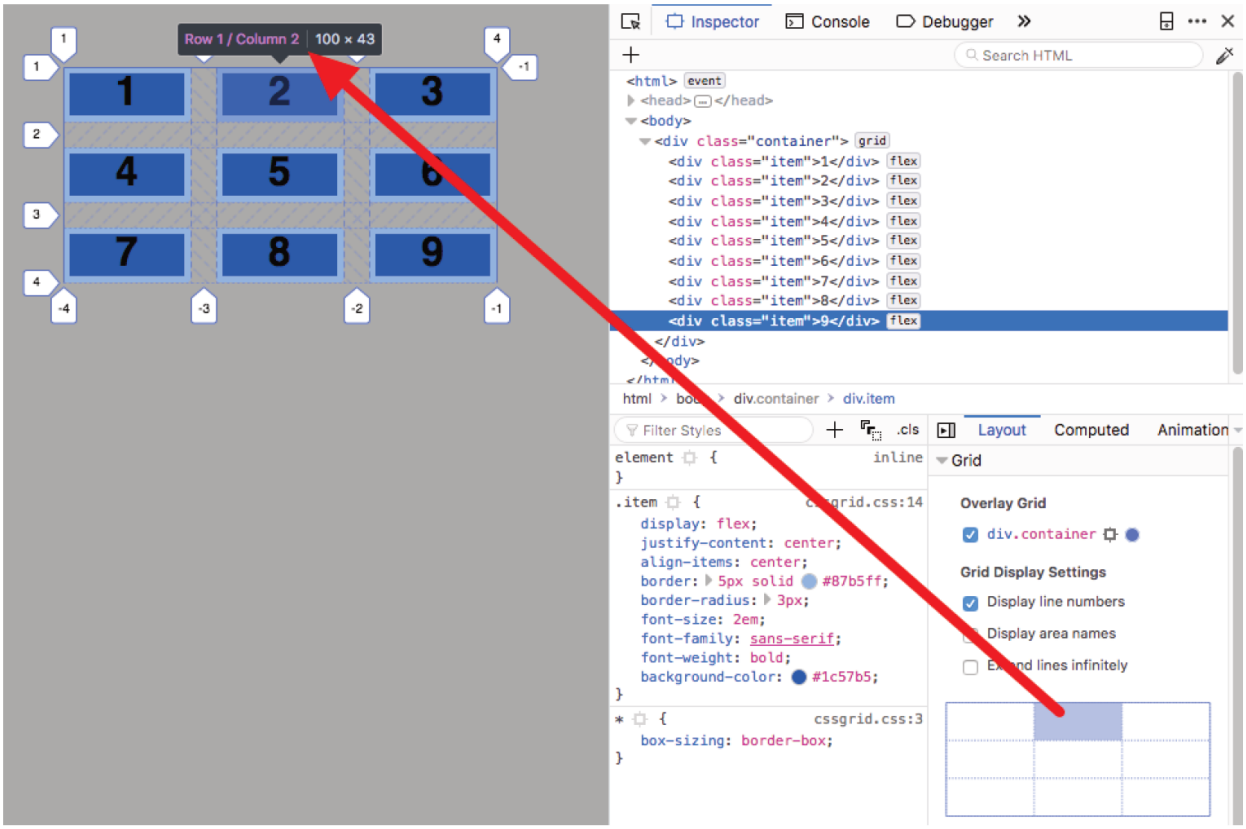
2- Grid Line
3- Grid Cell
4- Gride Area
5- Grid Track
6- Grid Gap



You can define grid by
Target { display: grid}

See grid.html with grid1.css

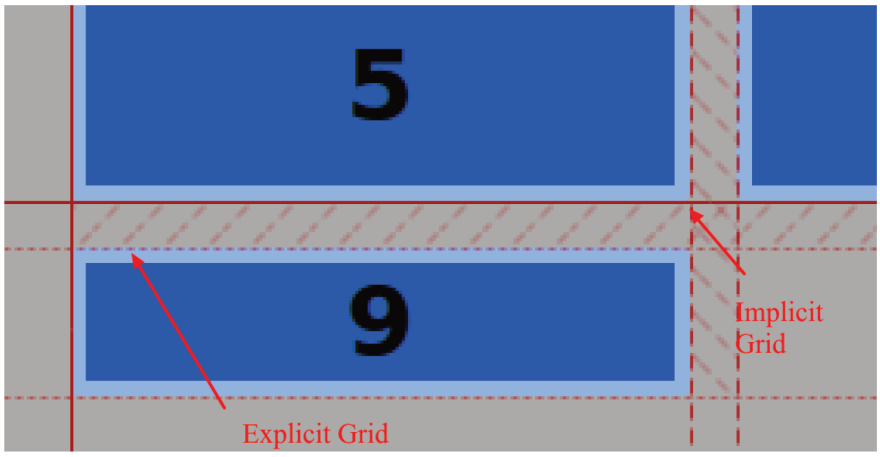
Developer tools in Firefox provides good information about Grid.



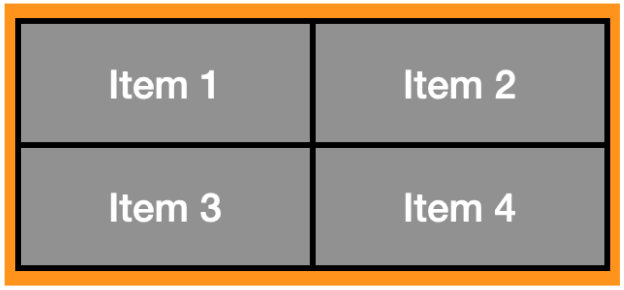
Explicit and Implicit Grid

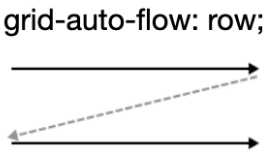
```
.container {
  display: grid;
  /* implicit grid */
  grid-template-columns: 100px 100px 100px;

  /* explicit grid */
  grid-template-rows: 60px 120px;
  grid-gap: 20px;
  }
```

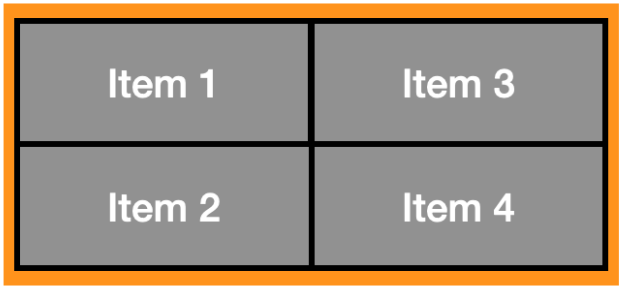
Check grid.html with grid2.css:

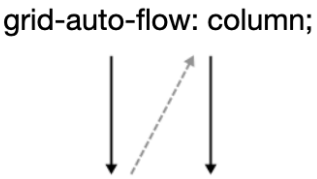


AUTOFLOW PROPERTY IN CSS GRID



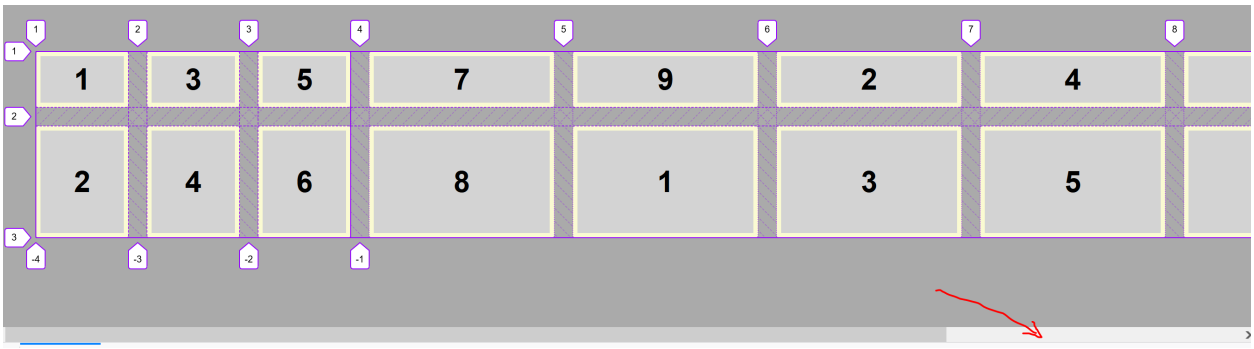






For column autoflow, there is a problem with scrolling if you have a lot of items in your grid

```
.container {
  display: grid;
  /* implicit grid */
  grid-template-columns: 100px 100px 100px;
  /* explicit grid */
  grid-template-rows: 60px 120px;
  grid-gap: 20px;
  grid-auto-flow: column;
}
```



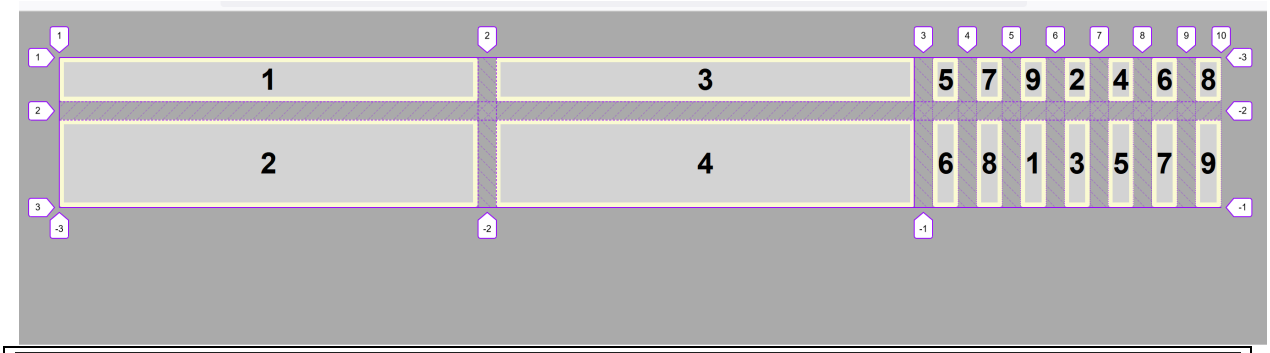
Check grid.html with grid3.css

As you see, the width of the grid is larger than display and need to be adjusted.

USE THE FR UNIT FOR LAYOUT

Using **fr** means fraction of remaining space.

```
.container {
  display: grid;
  /* relative sizes */
  /* fr stands for fraction of available space */
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 1fr 2fr;
  grid-gap: 20px;
  grid-auto-flow: column;
```




```
 /* try the following 15% is the percentage of container but fr means available
space */
  grid-template-columns: 2fr 1fr 150px 15%;
```

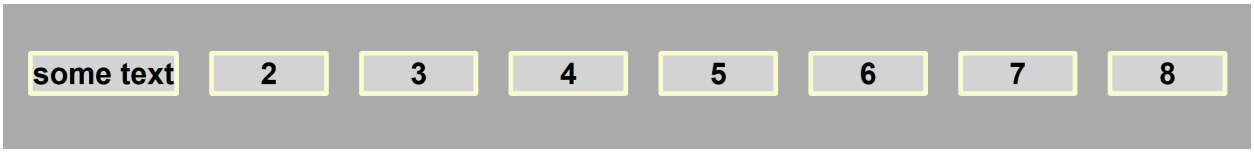
Use grid.html and grid4.css

THE AUTO KEYWORD

The auto keyword gives you the ability to adjust the width of the grid item to the max width of its
content. That means that element is responsible for sizing the column.

```
.container {
  display: grid;
```

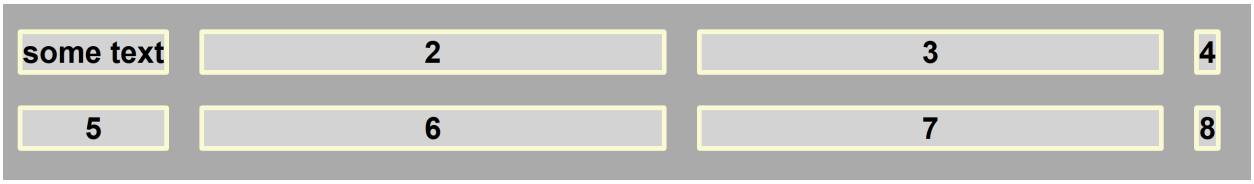
```
  grid-template-columns: auto 1fr 1fr 1fr 1fr 1fr 1fr 1fr;
  grid-gap: 2rem;
}
```



Use grid2.html and grid5.css

Let's add another row.

```
.container {
  display: grid;
  grid-template-columns: auto 1fr 1fr auto;
  grid-gap: 2rem;
}
```



Use grid2.html and grid6.css

Repeat Notation

Instead of

```
.container {
grid-template-columns: 1fr 2fr 1fr 2fr;
}
```

```
.container {
  grid-template-columns: repeat(1fr 2fr);
}
```

Use grid2.html and grid7.css

Or you can also use multiplier that applies it only few times (no space after "repeat") (grid7.css)

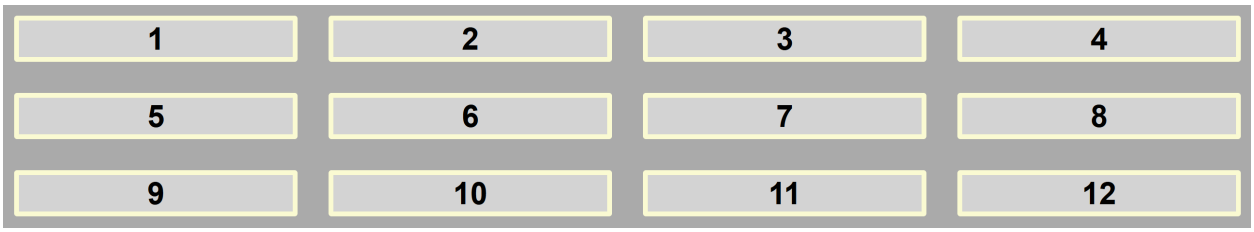
```
.container {
  grid-template-columns: repeat(2, 1fr 2fr);
}
```

SIZE GRID ITEMS WITH THE SPAN KEYWORD

Use grid3.html

Grid8.css

```
.container {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  grid-gap: 2rem;
}
```

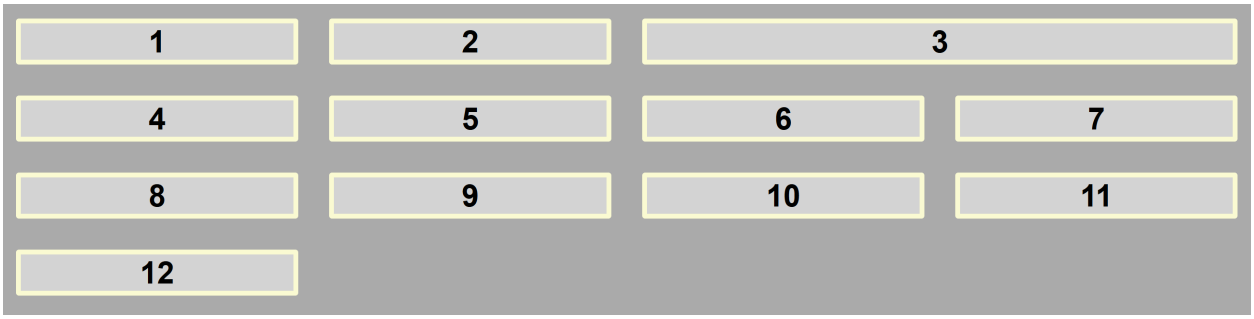


Use grid3.html and grid8.css

Resizing the Items

Use grid9.css

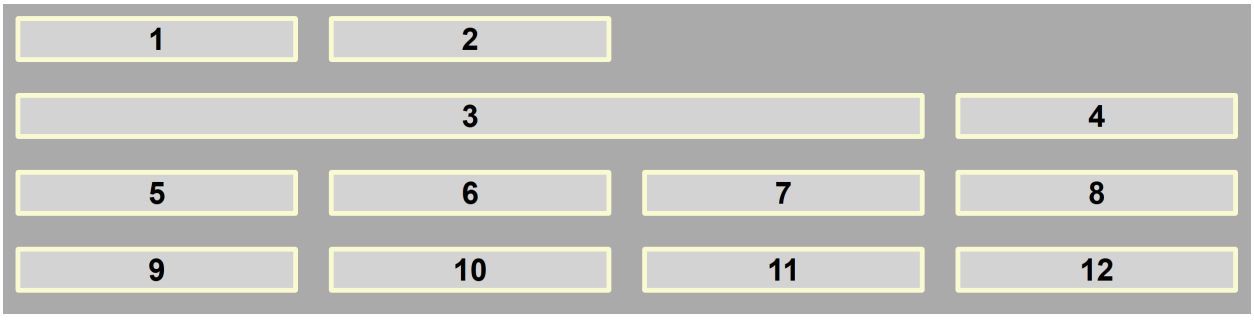
```
.item3 {
  grid-column: span 2;
}
```



Use grid3.html and grid9.css

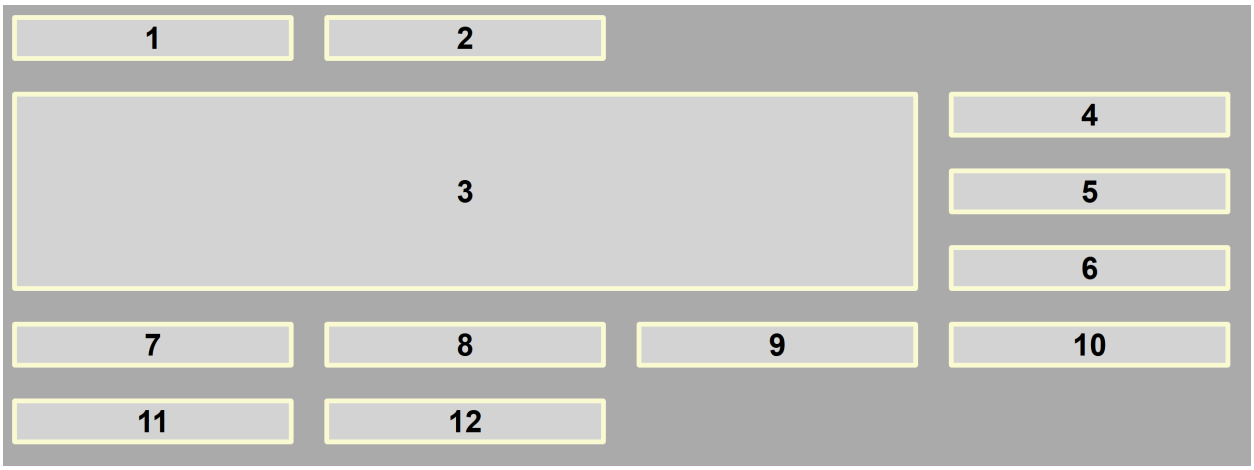
Also try:

```
.item3 {
  grid-column: span 3;
}
```

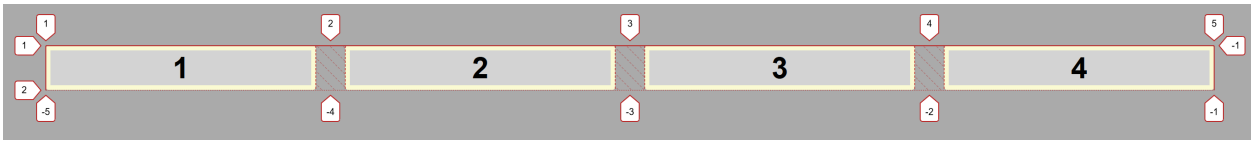


Now try:

```
.item3 {
  grid-column: span 3;
  grid-row: span 3;
}
```

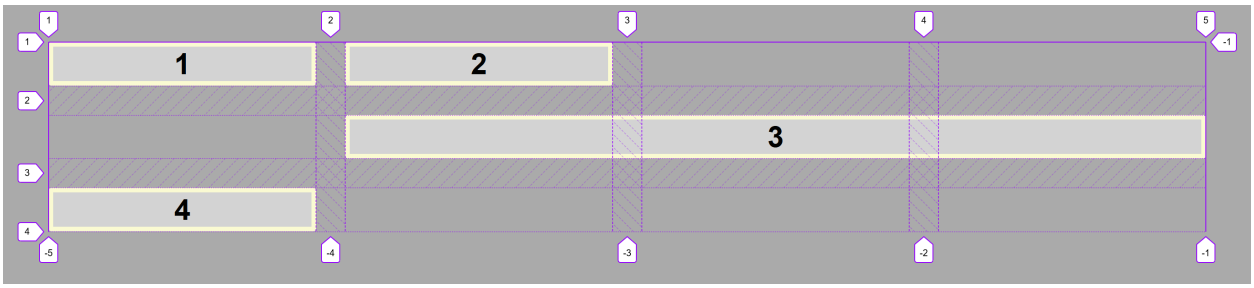


USE LINE NUMBERS IN CSS GRID



Now try

```
.item3 {
  grid-column-start: 2;
  grid-column-end: 5;
}
```



This is equal to:

```
.item3 {
  grid-column-start: 2;
  grid-column-end: -1;
}
```

You can shorten the code:

```
.item3 {
  grid-column: 2/5;
  /* grid-column-start: 2;
  grid-column-end: 5; */
```

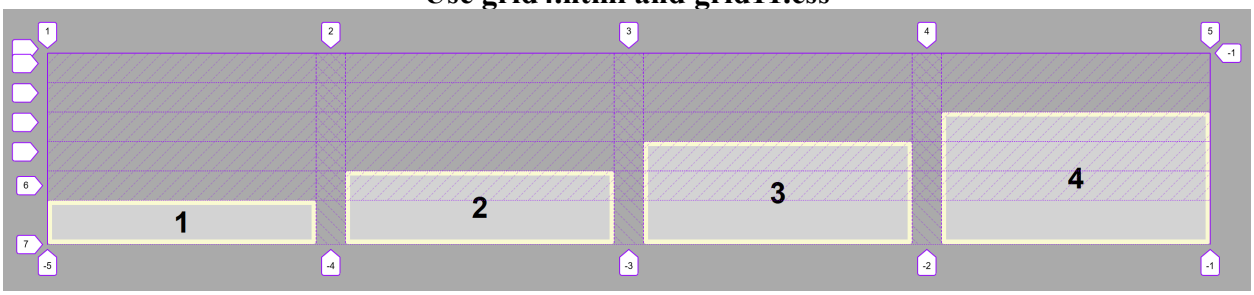
```
}
```

Use grid4.html and grid10.css

Also try the following combination:

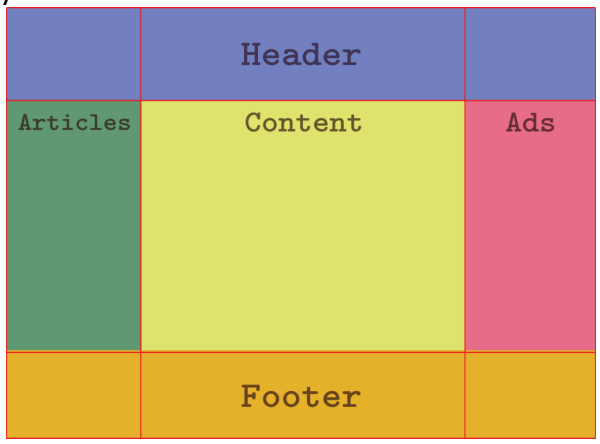
```
.item1 {
  grid-column: 1 / 2;
  grid-row: 6 / 7;
}

.item2 {
  grid-column: 2 / 3;
  grid-row: 5 / 7;
}

.item3 {
  grid-column: 3 / 4;
  grid-row: 4 / 7;
}

.item4 {
  grid-column: 4 / 5;
  grid-row: 3 / 7;
}
```

Use grid4.html and grid11.css

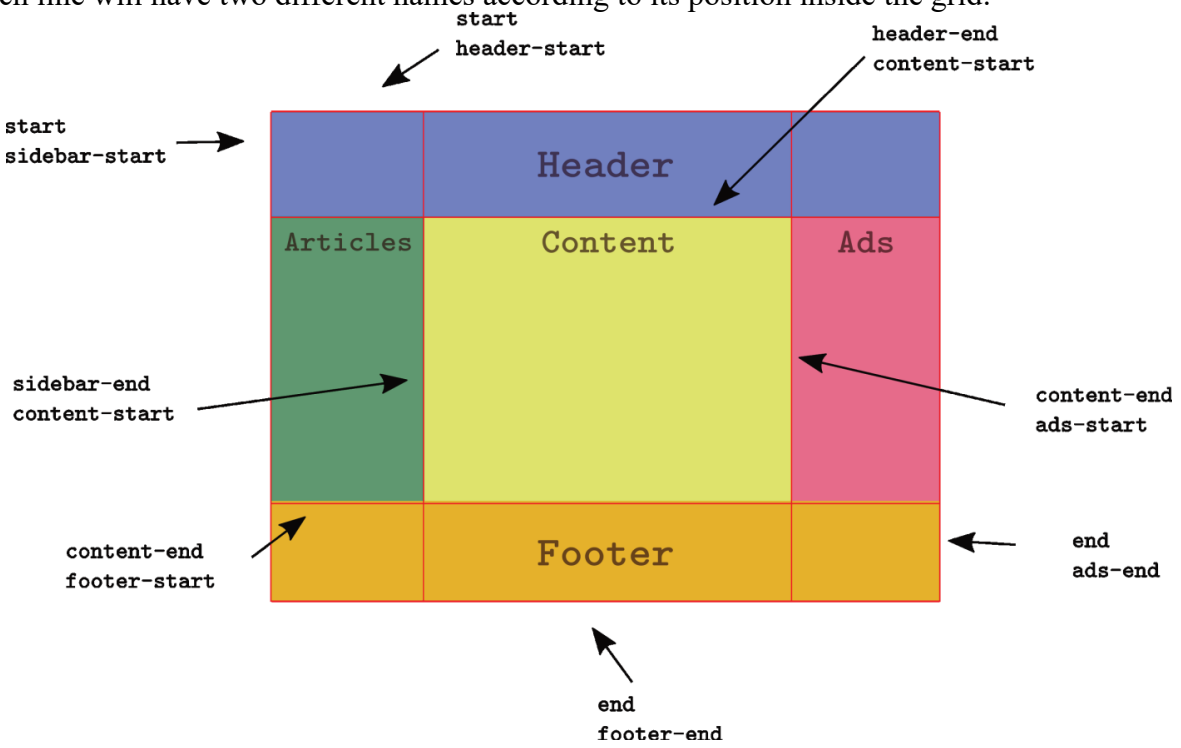


USE LINE NAMES IN CSS GRID

You learned how to place items according to the numbered lines in CSS Grid.
Here, it is explained how to name the lines that compose CSS Grid.
In order to use line names, you have to declare them first. The notation for declaring a line is as follows:
[first-line]
The names of the lines are declared inside the grid-template-columns and grid-template-rows properties. A line can have two or more different names with this notation:
[first-line start-line]
Consider the following layout:



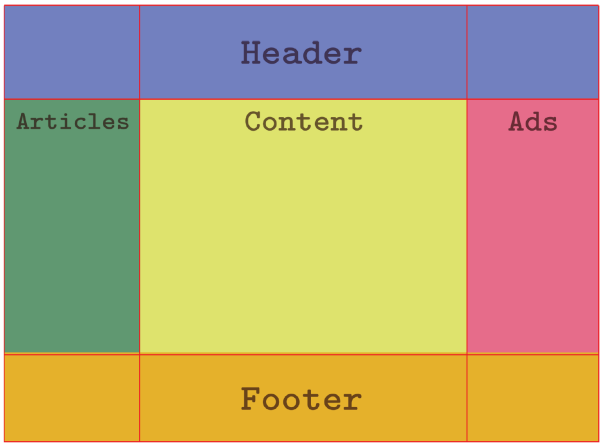
The grid is composed of four lines from top to bottom and four lines from left to right. We can now define names for these lines and then declare them in the CSS file.
You can name the lines whatever you want but make sure you choose relevant names, in order to work more easily.
Each line will have two different names according to its position inside the grid:



Use grid5.html and grid12.css

USE Grid Templates IN CSS GRID

If you try to make a layout like below:



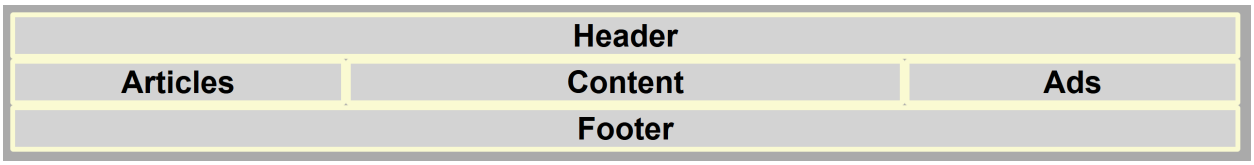
Use grid6.html and grid13.css

Using template as:

Header Header Header
Ariticles Content Ads
Footer Footer Footer

```
.container {
  display: grid;
  grid-template-columns: 3fr 5fr 3fr;
  grid-template-rows: auto auto auto;
  grid-template-areas: "header header header"
                       "articles content ads"
                       "footer footer footer";
}

.header {
  grid-area: header;
}

.articles {
  grid-area: articles;
}

.content {
  grid-area: content;
}

.ads {
  grid-area: ads;
}
```

```
.footer {
  grid-area: footer;
}
```



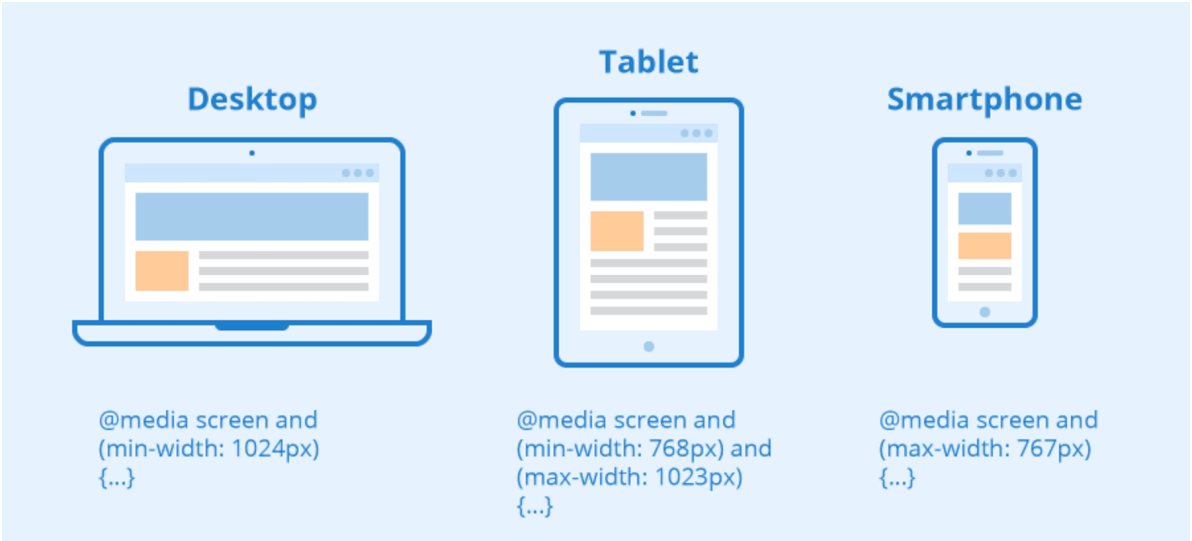
Use grid5.html and grid13.css

WORKING WITH MEDIA QUERIES

So far, you've assigned each one of the items to their respective areas. You may want to create a responsive webpage so to edit the grid-template-areas property to match the layout to different screen sizes:

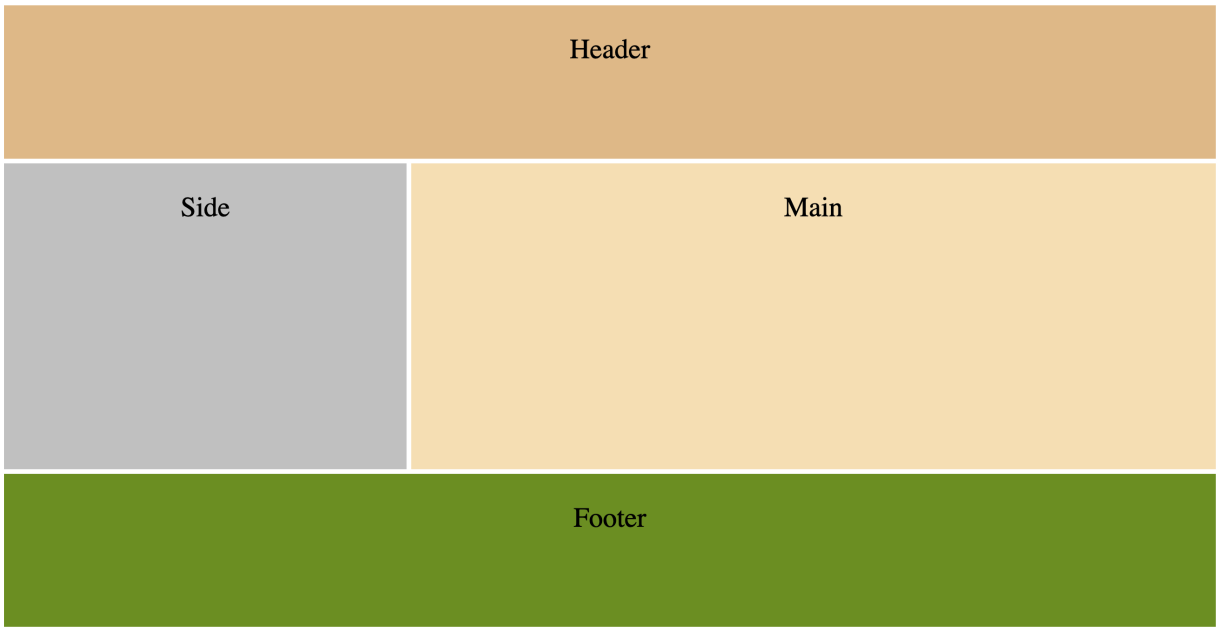
```
container {
  display: grid;
  grid-template-columns: 3fr 5fr 3fr;
  grid-template-rows: auto auto auto;
}

@media (max-width: 767px) {
  .container {
    grid-template-columns: 1fr;
    grid-template-areas:
    'header'
    'articles'
    'content'
    'footer';

    .ads {
      display: none;
    }

    margin: auto;
    /* width: fit-content; */
    border: 2px solid black;
  }
}

@media (min-width: 768px) and (max-width: 3024px) and (orientation: landscape) {
  .container {
    grid-template-areas:
    'header header header'
    'articles content ads'
    'footer footer footer';
  }
```

See grid6.html and grid14.css



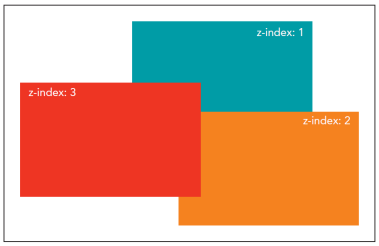
Grid Exercise 1:

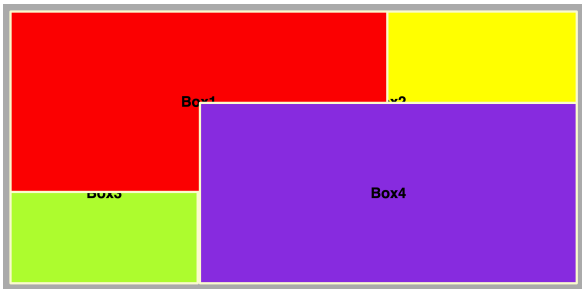
Create the following page using Grid



Stacking Elements

Figure 3–77 Using the z-index property to stack elements

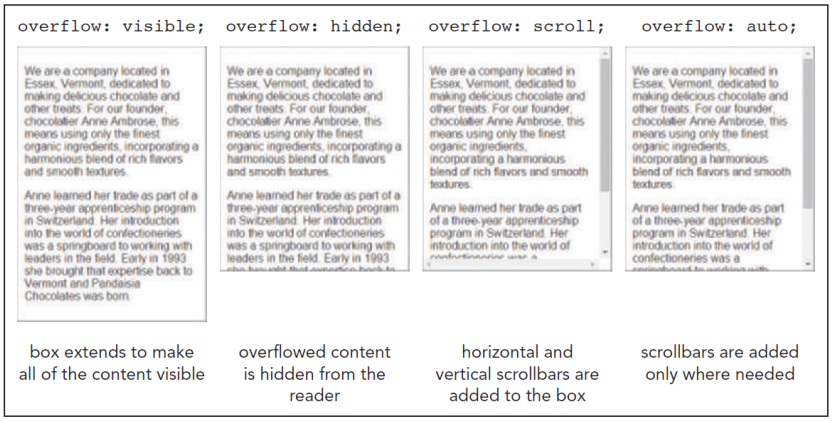




Check grid7.html and grid15.css

Overflow Control

Figure 3–73 Values of the overflow property



Clipping an Element

Figure 3–76 Clipping an image