CDI college

business • technology • healthcare

Web design and develpment II (CW2)

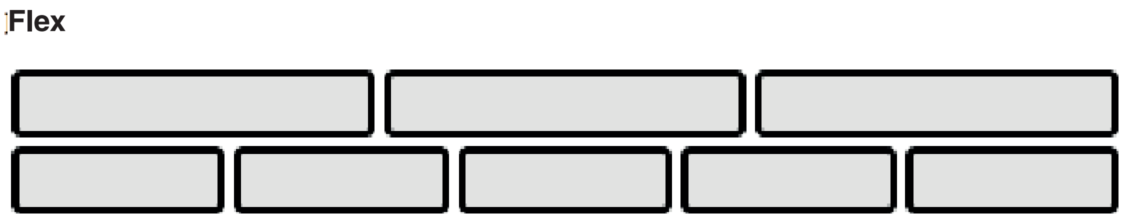
Class 3

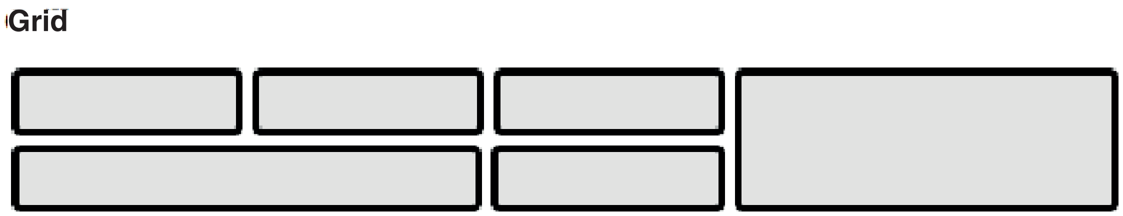
Grids

Since the very beginning of your web classes, you have been building grids. First using HTML containers, then with flex-boxes.

Before CSS was invented, web developers used tables to organize the pages elements, which was not an optimal solution. CSS allowed to create containers to better organize content,. Then Flex made it possible to position content in a much more flexible way.

Working a lot like flex-box, grid is a display defining a container into a grid so its items automatically position themselves upon developer's wishes, allowing to manage content in much more details and with more possibilities.





Simple grid

A container must first be defined as a grid..

The procedure is the following:

1. Apply **display: grid** to the container's CSS.

2. Indicate the number of rows and columns using the properties **grid-template-columns** and **grid-template-rows**, then define the wanted dimensions.

3. Indicates the gutter's width using **grid-gap**.

4. Place your items into the container.

« fr » measure unit

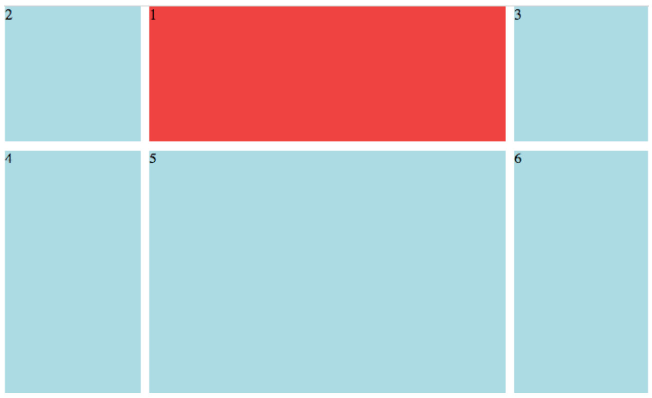
« fr » means a fraction of the grid and it is working a lot like flex-grow. In the example besides, the first and last columns width is 150px. The column in between indicates it uses the remaining space.

If every columns width would be equal to 1fr, this would mean they all use equal parts of the available space.

```
div.container {
        width: 100%;
        height: 100%;
        display: grid;
        grid-template-columns: 150px 1fr 150px;
        grid-gap: 10px 20px;
}

div.item {
        background-color : lightblue ;
}

<div class="container">
        <div class="item"> </div>
        <div class="item"> </div>
        <div class="item"> </div>
        <div class="item"> </div>
        <div class="item"> </div>
        <div class="item"> </div>
        <div class="item"> </div>
        <div class="item"> </div>
</div>
```

Positioning items in a grid

Using **grid-column** and **grid-row**, it is easy to indicates where to position an item. :



```
div.red {
        background-color: red;
        grid-column: 2;
        grid-row: 1
}
```

Gutters

Properties **grid-column-gap** and **grid-row-gap**. allow you to specify both horizontal and vertical gutters dimensions.

Shorthand :
grid-gap: *column row*;

```
grid-gap: 10px 20px;
```

The repeat() function

To create many equal columns, instead of writing all the measurments, you can use this simple function. Its first value is a multiplication factor, and the second value is a measurement.

```
grid-template-columns: repeat(3, 33.33%);
grid-template-rows: repeat(4, 100px);
grid-template-columns: repeat(3, 1fr);

grid-template-columns: 30px repeat(3, 1fr) 30px;
```

Using start and end numbers to position items in a grid

Here, you only have to indicate starting and ending row or column to indicate the position or span of an item in the grid.

```
grid-row-start:    2;
grid-row-end:      3;
grid-column-start: 2;
grid-column-end:   3;

or

grid-row:    2;
grid-column: 3 / 4;
```

Grid-area

Grid-area is a shorthand of *grid-row-start, grid-column-start, grid-row-end* and *grid-column-end* . All that's needed here is to write the ordre values one after the other.

grid-area: 2 / 2 / 3 / 3;

Row and columns span

It is also possible to specify the cell's start and end of an item. In the example beside, the item spans from the first to the third column's start.

grid-column: 1 / 3;

The property « span » may also be used to specify to an item to span on a certain number of cells. In the example beside, cell 2 spans over two columns.

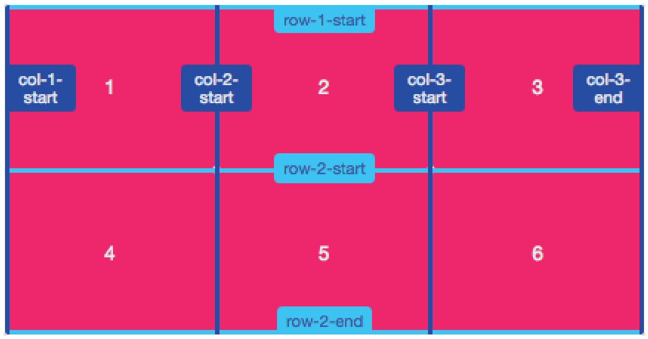
grid-column: 2 / span 2;

Naming grid lines

Grid lines can be associated to names as values of property « grid-template-rows » and « grid-template-columns » writing them between brace brackets.

grid-template-rows:
[row-1-start] 1fr [row-2-start] 1fr [row-2-end];

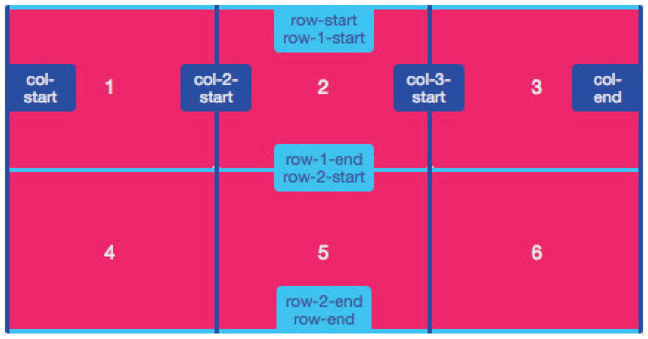
grid-template-columns:
[col-1-start] 1fr [col-2-start] 1fr [col-3-start] 1fr [col-3-end];



Many names can be associated to a unique grid line. It then needs tu be put between brace brackets and separated using a space.

grid-template-rows:
[row-start row-1-start] 1fr [row-1-end row-2-start] 1fr [row-2-end row-end];

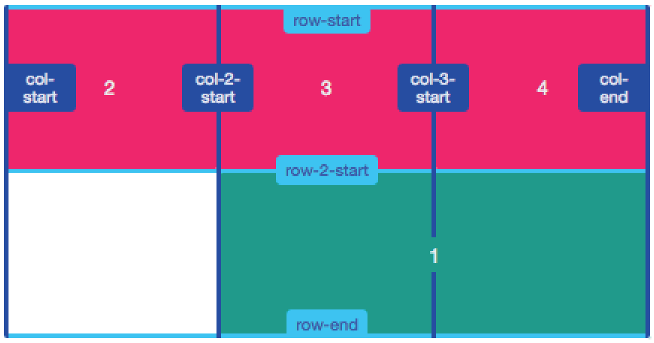
grid-template-columns:
[col-start] 1fr [col-2-start] 1fr [col-3-start] 1fr [col-end];



Positioning items using grid lines names

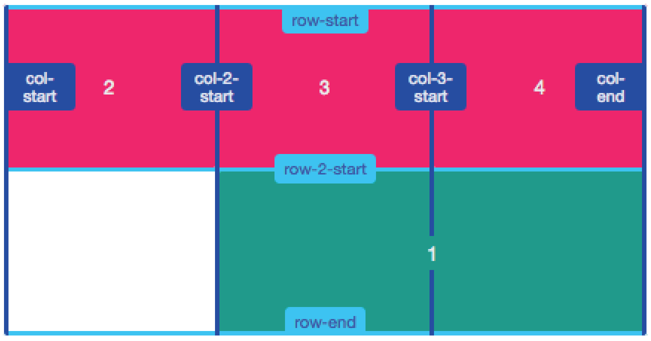
When names are associated to grid lines, it becomes possible to position items using these names.

```
grid-row-start: row-2-start;
grid-row-end: row-end;
grid-column-start: col-2-start;
grid-column-end: col-end;
```



Properties ***grid-row*** an ***grid-column*** can also be used as well as the function *repeat()*.

```
grid-row:    row-2-start / row-end;
grid-column: col-2-start / col-end;
```



Although, the use of the function *repeat()* assigns the same name to many lines.

However, the number representing the order of these automatically generated lines is implicit and can therefor be used for items positioning.

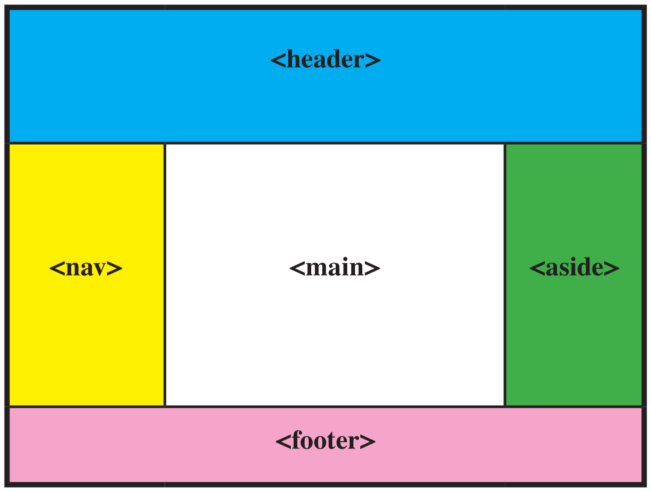
```
grid-template-rows:
repeat(3, [row-start] 1fr [row-end]);

grid-template-columns:
repeat(3, [col-start] 1fr [col-end]);

grid-row:    row-start 2 / row-end 3;
grid-column: col-start / col-start 3;
```

Positioning items using grid-area

Just like for grid lines, the cells formed by rows and columns can be named (between quote marks and separated by spaces) and then be used when positioning items.

So, a grid containing three rows and three columns could be created where names would be associated to each of the nine cells.

In order to position items in the grid below, see the example beside.



To modify the items positioning, it is only needed to change the properties values (grid-template-areas) or simply associate the selectors to other names.

It is also possible to use preceding methods such as :

- grid-row-start / grid-row-end
- grid-row / grid-rcolumn

```
grid-template-areas: "top top top"
                     "left center right"
                     "bottom bottom bottom";

grid-template-rows: 100px 1fr 50px;
grid-template-columns: 300px 1fr 200px;


header {
        grid-area: top;
}

nav {
        grid-area: left;
}

main {
        grid-area: center;
}

aside {
        grid-area: right;
}

footer {
        grid-area: bottom;


grid-row-start: top;
grid-row-end: top;
grid-column-start: top;
grid-column-end: top;

grid-row: bottom;
grid-column: bottom;
```

Rows and columns automatic generation

To automatically create extra rows or columns, the properties ***grid-auto-rows*** or ***grid-auto-columns*** can be used with a height and a width.

```
grid-template-rows: 70px;
grid-template-columns: repeat(2, 1fr);
grid-auto-rows: 140px;
grid-auto-columns: 1fr;
```

Items alignment into grids

Just like for flex, grids allow you to align items vertically and horizontally.

Horizontal :
justify-items

Vertical :
align-items

Supported values :

- auto
- normal
- start
- end
- center
- stretch
- baseline
- first baseline
- last baseline
- space-evenly