Co-funded by the
Erasmus+ Programme
of the European Union

Code4SP Training Materials
Subchapter 3: CSS

WP3: Code4SP Training Materials

Prepared by: Social Hackers Academy

CSS GRID

Co-funded by the
Erasmus+ Programme
of the European Union

WHAT IS CSS GRID?

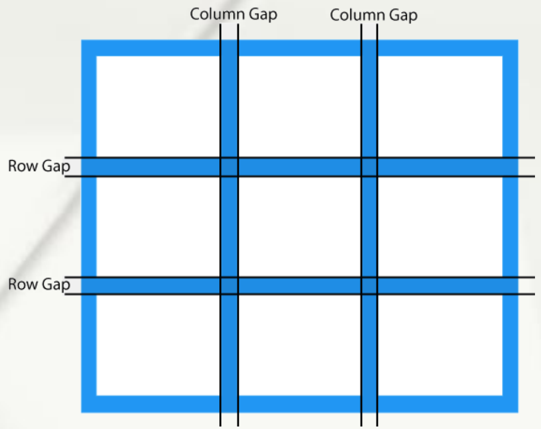
- **CSS grid layout** or **CSS grid creates** complex responsive web design layouts more easily and consistently across browsers. There have been other methods for controlling web page layout methods, such as tables, the box model, and CSS flex box.

- To get started you have to define a container element as a **grid** with **display: grid**, set the column and row sizes with **grid-template-columns** and **grid-template-rows**, and then place its child elements into the **grid** with **grid-column** and **grid-row**.

Co-funded by the Erasmus+ Programme of the European Union

WHAT IS CSS GRID?

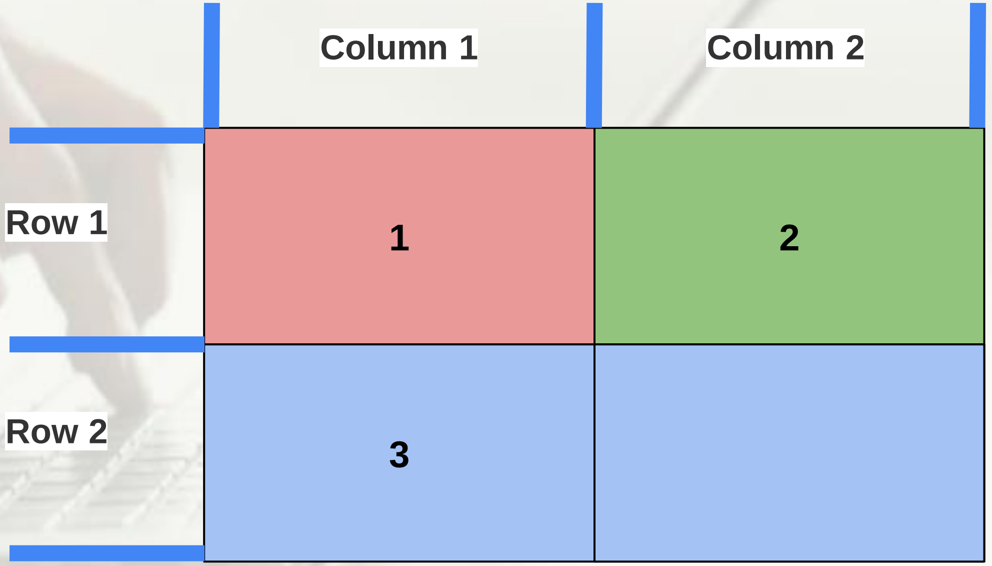
- CSS Grid based on rows and columns perspective, we will see how it works and what the orders that we should use.
- We can take this example on the next slide and let's build it using just grid, and discover the different ways to do it.



Co-funded by the Erasmus+ Programme of the European Union

Rows and Columns Perspective

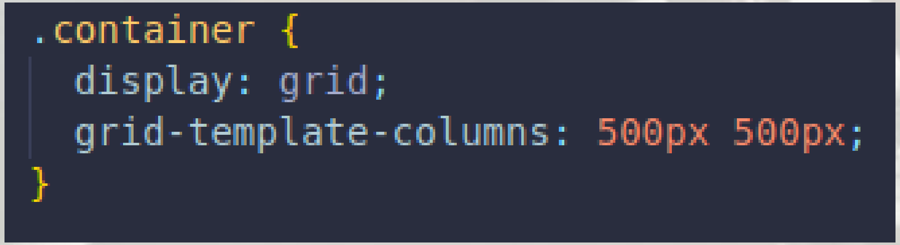
- As You can see here we have a design base on **columns** and **Rows**. For this design we have just two columns and two rows, we can made this design by puting the box 1 & 2 on the same div than we just give it an absolute width, and for the box 3 it will be outside of the previous div and it will have the same css commands. But using **CSS grid** it will be even much easier.



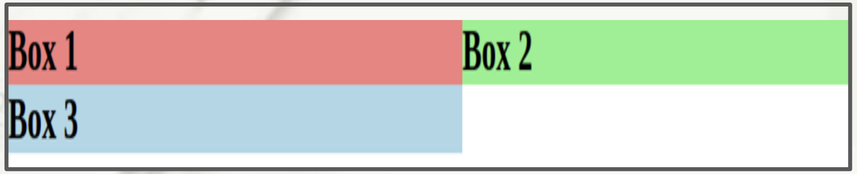
Co-funded by the Erasmus+ Programme of the European Union

Rows and Columns Perspective

- Using the following commands on the **CSS file** will give the result on the left. **grid-template -columns** it can takes more than value based on how mush columns you want to have, in this case i just pass on it two values and you can see that it make just two columns with **500px** of width even if there's an extra space at the left.



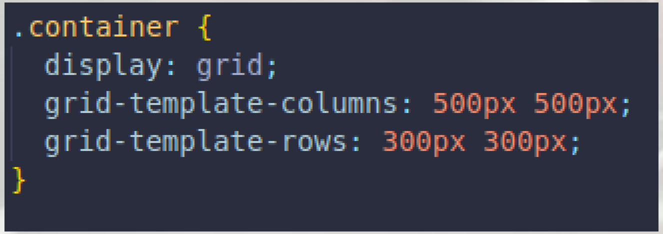
```
.container {
  display: grid;
  grid-template-columns: 500px 500px;
}
```



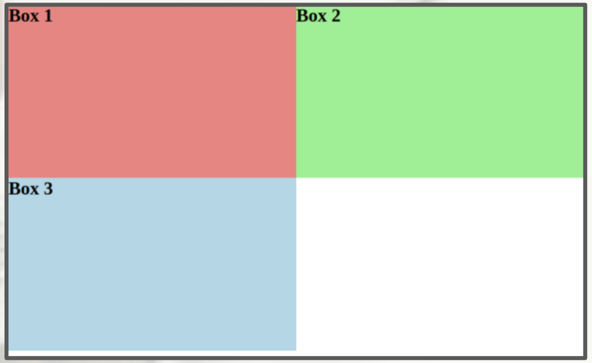
Co-funded by the
Erasmus+ Programme
of the European Union

Rows and Columns Perspective

- Also their is **grid-template -rows,** and us you can see below it makes the heights more bigger and this is because I had given **300px** to each row.



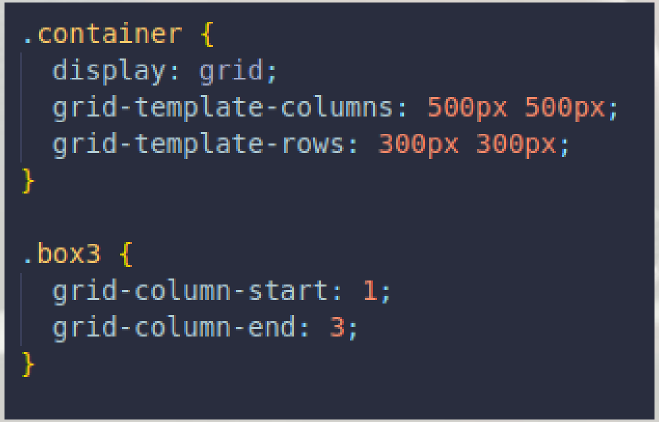
```css
.container {
  display: grid;
  grid-template-columns: 500px 500px;
  grid-template-rows: 300px 300px;
}
```



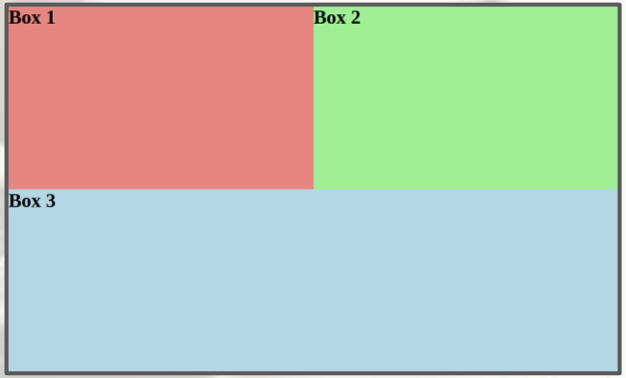
Co-funded by the Erasmus+ Programme of the European Union

Rows and Columns Perspective

- Now we can stretch **box 3** over the empty space under the **box 2** just by using the **grid-column-start / end** using this command we choose the start point and the end and **BOOM** we just made it.



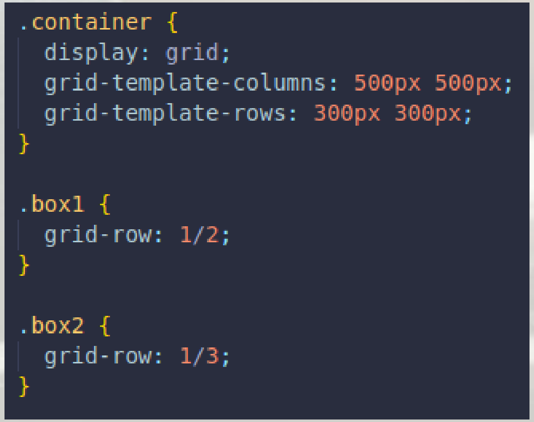
```
.container {
  display: grid;
  grid-template-columns: 500px 500px;
  grid-template-rows: 300px 300px;
}

.box3 {
  grid-column-start: 1;
  grid-column-end: 3;
}
```



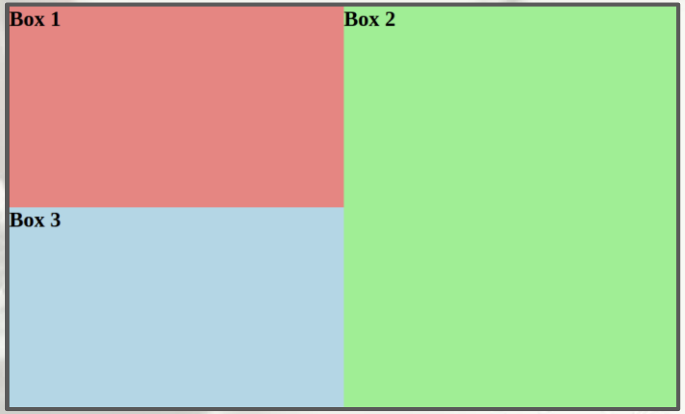
Co-funded by the
Erasmus+ Programme
of the European Union

Rows and Columns Perspective

- We can do the same for **rows**, using **grid-row** and follow it with the exact position that we want like you can see in the example below.



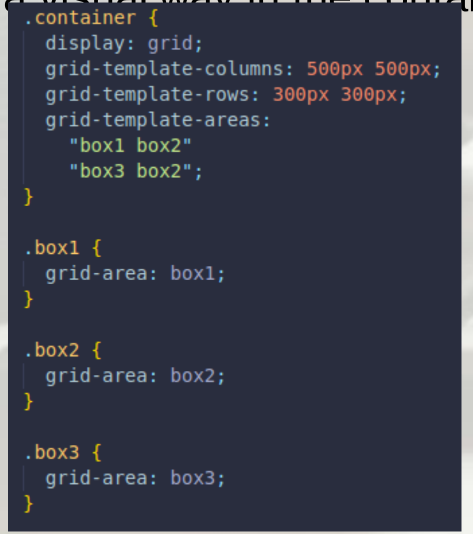
```
.container {
  display: grid;
  grid-template-columns: 500px 500px;
  grid-template-rows: 300px 300px;
}

.box1 {
  grid-row: 1/2;
}

.box2 {
  grid-row: 1/3;
}
```



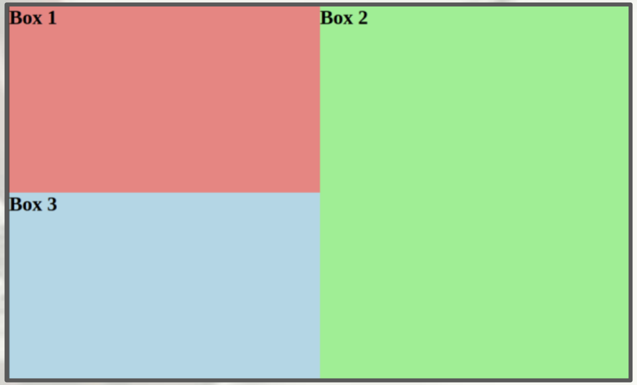
Co-funded by the Erasmus+ Programme of the European Union

Rows and Columns Perspective

- To make this even more easier we can use **grid-template-areas** and as you can see it give us the same result, by declaring the areas names in the children's and use it in a visual way in the container.



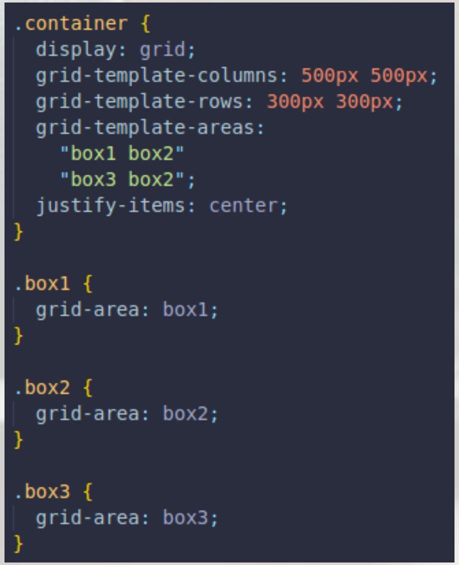
```
.container {
  display: grid;
  grid-template-columns: 500px 500px;
  grid-template-rows: 300px 300px;
  grid-template-areas:
    "box1 box2"
    "box3 box2";
}

.box1 {
  grid-area: box1;
}

.box2 {
  grid-area: box2;
}

.box3 {
  grid-area: box3;
}
```



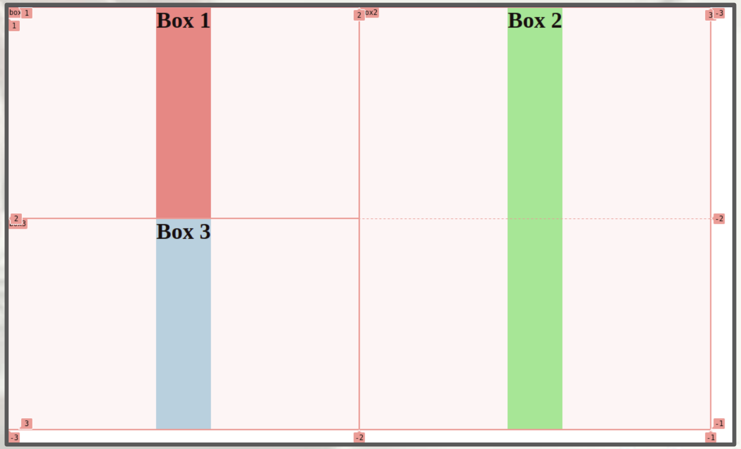
Co-funded by the Erasmus+ Programme of the European Union

Rows and Columns Perspective

- And also we can change all children’s position by using **justify-items** then we can choose it position like **center**,**start**,**end** and **stretch**



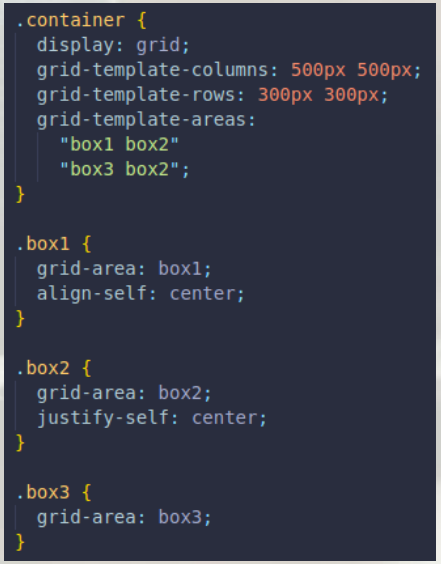
```
.container {
  display: grid;
  grid-template-columns: 500px 500px;
  grid-template-rows: 300px 300px;
  grid-template-areas:
    "box1 box2"
    "box3 box2";
  justify-items: center;
}

.box1 {
  grid-area: box1;
}

.box2 {
  grid-area: box2;
}

.box3 {
  grid-area: box3;
}
```



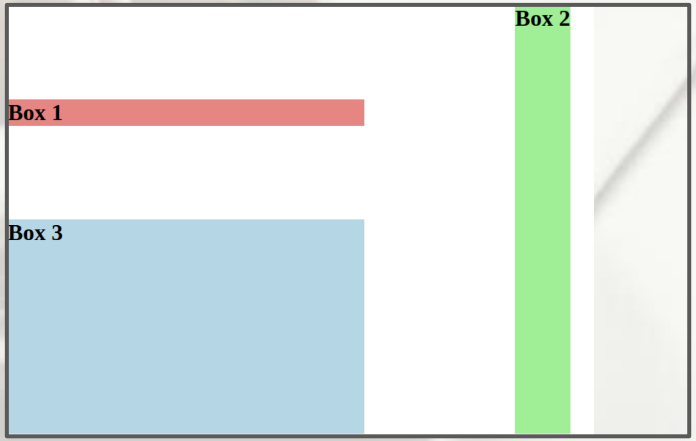
Co-funded by the
Erasmus+ Programme
of the European Union

Rows and Columns Perspective

- We can also use the previous command for specific element, all we have to do is use **align-self** or **justify-self** in the child that we want to applied on.



```
.container {
  display: grid;
  grid-template-columns: 500px 500px;
  grid-template-rows: 300px 300px;
  grid-template-areas:
    "box1 box2"
    "box3 box2";
}

.box1 {
  grid-area: box1;
  align-self: center;
}

.box2 {
  grid-area: box2;
  justify-self: center;
}

.box3 {
  grid-area: box3;
}
```



Co-funded by the
Erasmus+ Programme
of the European Union