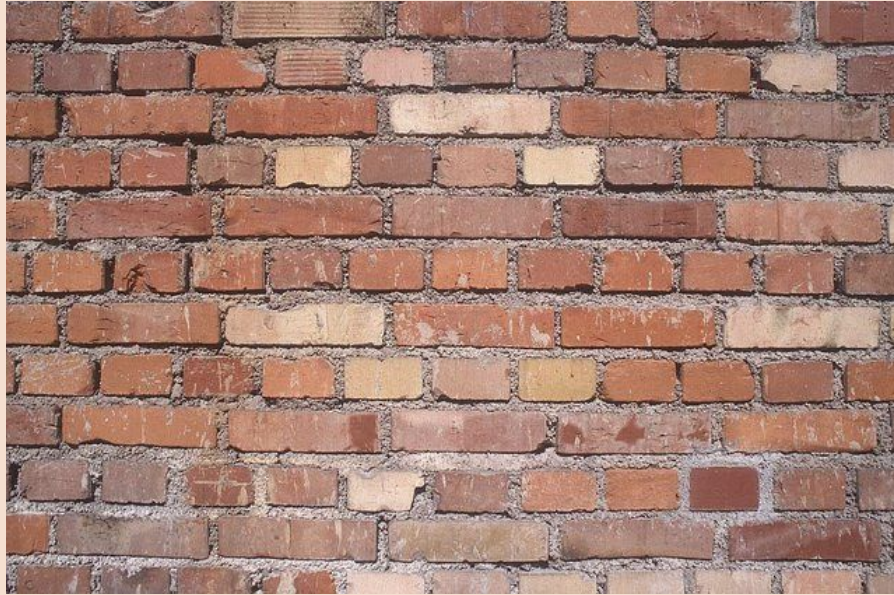


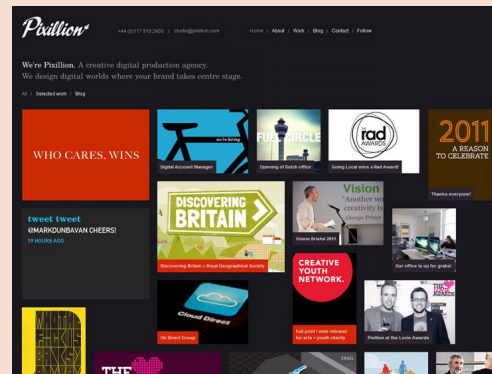
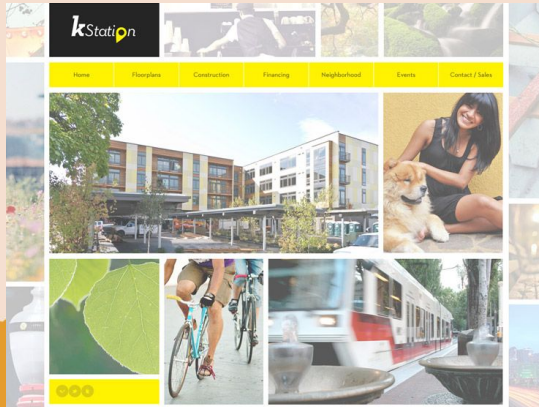
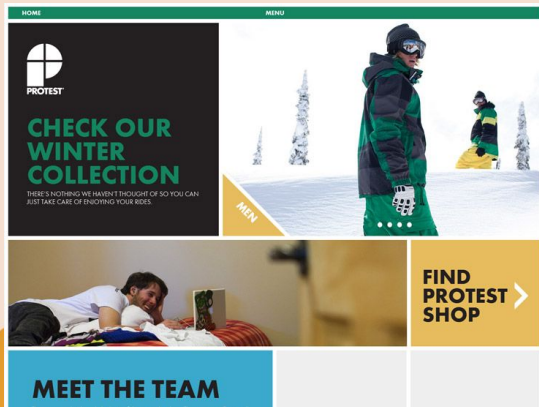
Bootcamp Grid Systems

<https://github.com/HackerYou/bootcamp-notes/blob/master/css/grid-systems.md>

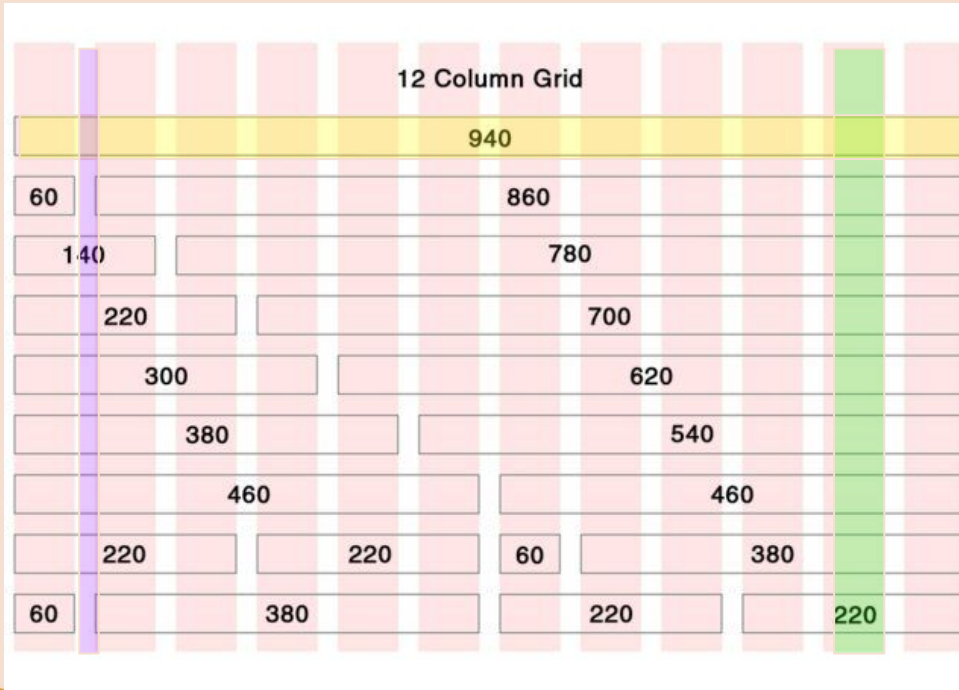
Grid System



>_ Juno



Web Design Grid System



Terminology

Column A vertical width that represents a section of content. Grid systems often use 12 or 16 columns within a row. Each row must contain a complete set of columns.

Row A 100% width that spans horizontally across the page. A row is made up of a complete set of columns.

Gutter The distance between each column. Allows equal margin between elements.

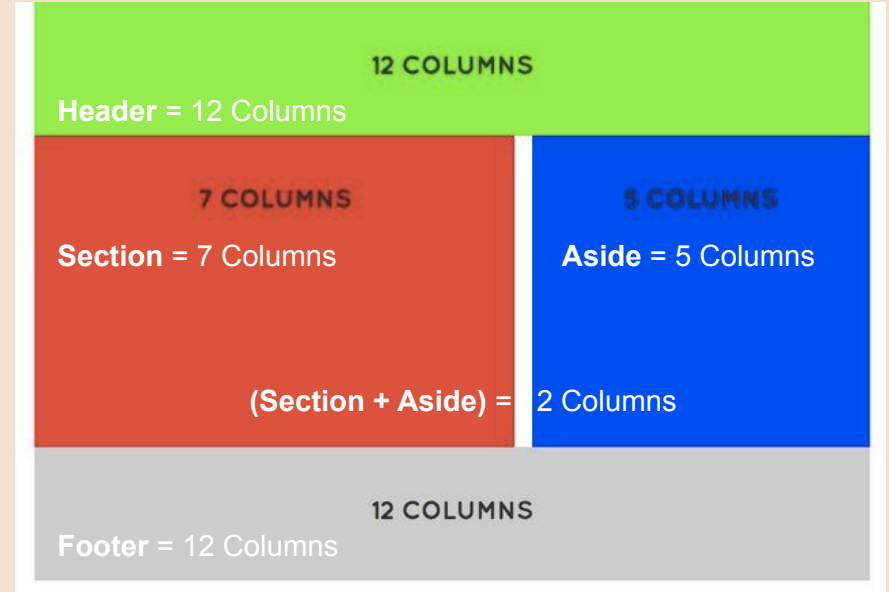
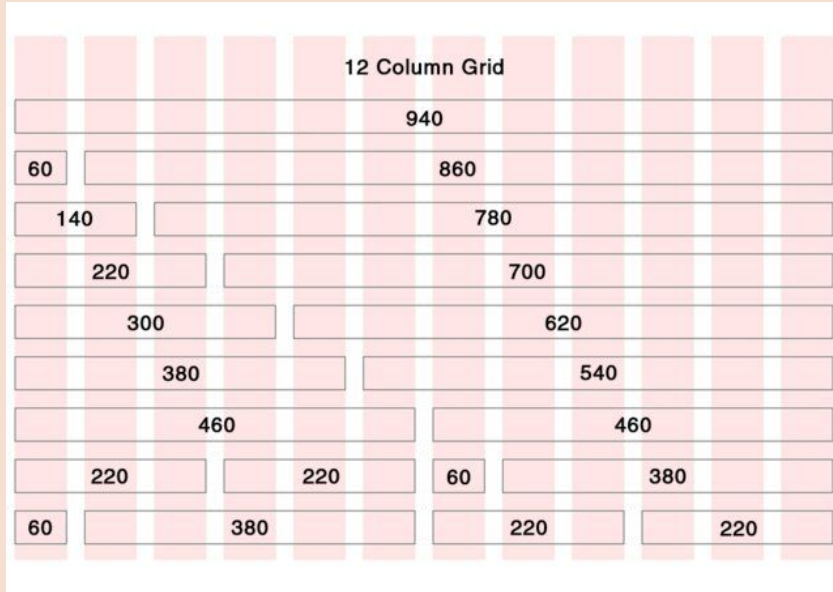
Fixed A definite width of each column. Not flexible.

Fluid A relative width of each column, Flexible and responsive. Typically used with a container div that has a max-width.

Presentational classes on elements to define width. (class="col-3")

Non-Presentational (e.g. semantic) classes based on content (class="header").

Layout with a Grid System

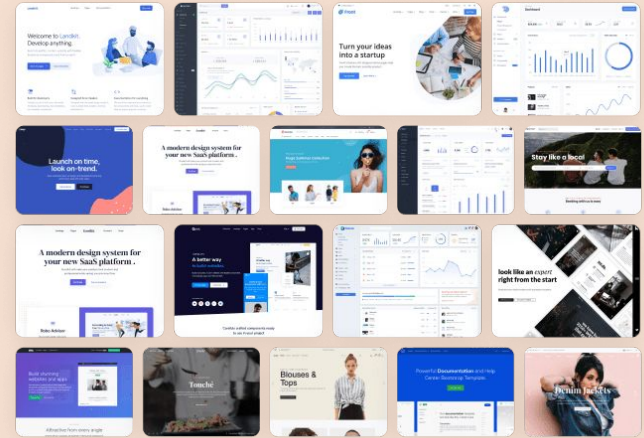


Most commonly used Grid Systems

- [Bootstrap](#) - (Fluid, Presentational, 12 Column)
- [Zurb Foundation](#) - (Fluid, Presentational, 12 Column)
- [Bourbon Neat](#) - (Fluid, Non-Presentational, 12 Column)
- [Semantic GS](#) - (Fluid, Non-Presentational, 12 Column)
- [960 Grid](#) - (Fixed, Presentational, 12 Column)
- [Profound Grid](#) - (Fluid, Non-Presentational, 12 Column)
- [Responsive Grid System](#) - (Fluid, Presentational, 12 Column)

You download or link the needed CSS files.

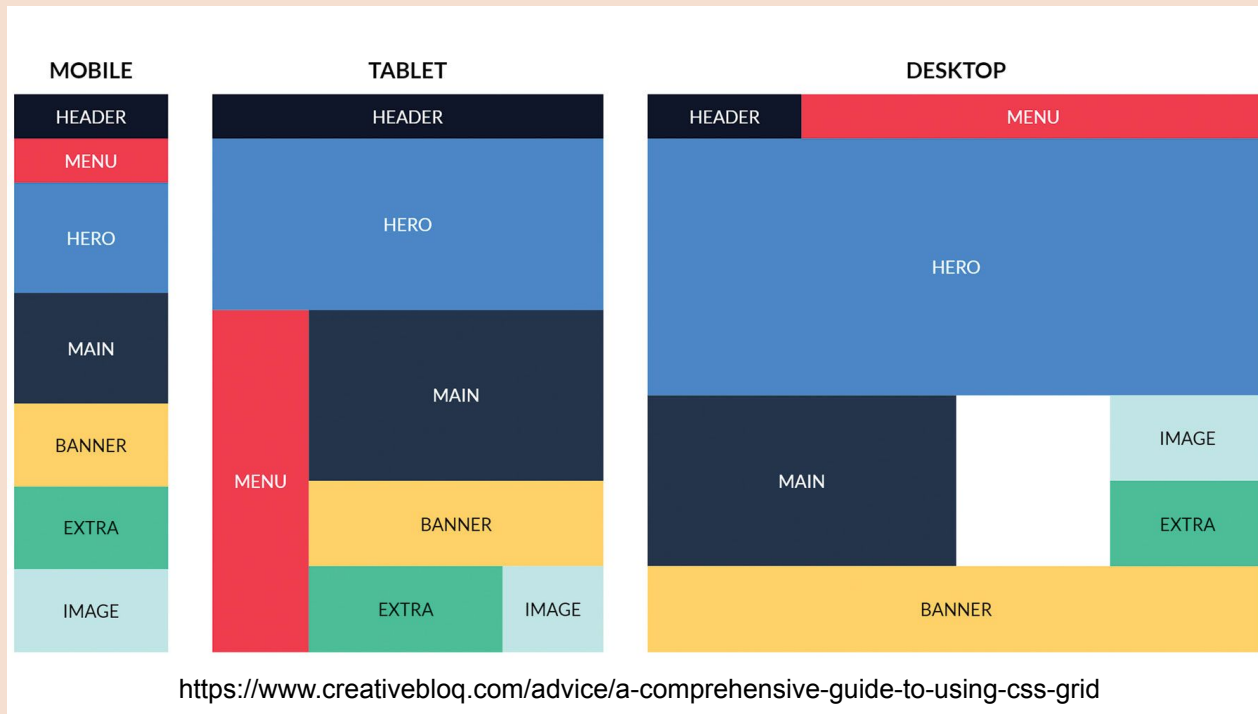
The frameworks are a collection of pre-written CSS that you apply through the use of HTML and classes.



Bootcamp CSS Grids

<https://github.com/HackerYou/bootcamp-notes/blob/master/css/css-grids.md>

CSS Grids



CSS Grids

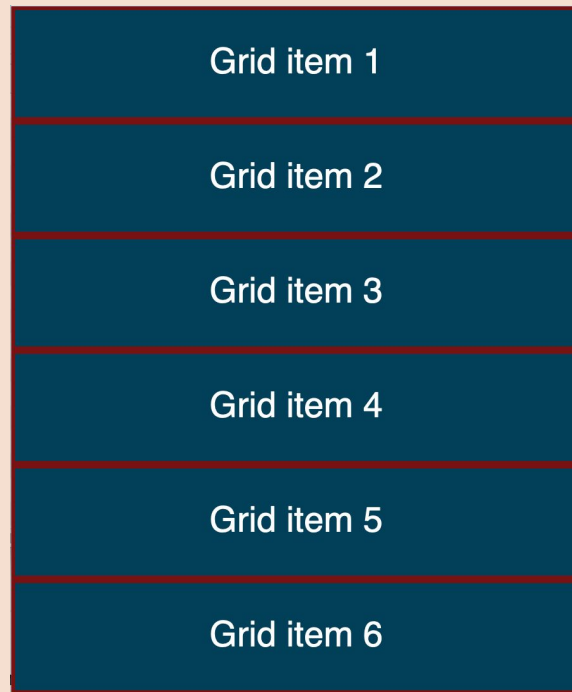
Two-dimensional system: columns x rows

ADDING TO Flexbox, floats, positioning: GRIDS!

1. Set **display** property on the **parent** element
2. Set **grid-template-columns** & **grid-template-rows** on the **parent** element

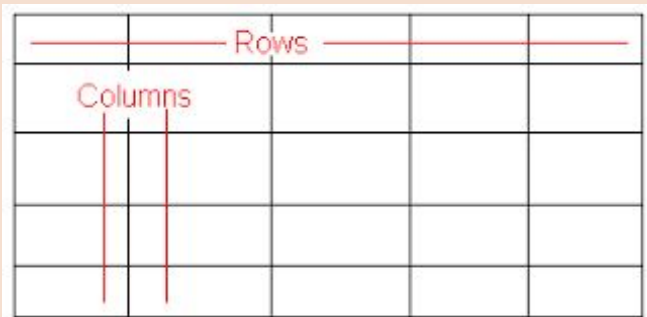
These properties take a unit of measurement, sometimes referred to as a *track size*, for each row or column you want to have. We will separate each row or column size with a space only, no commas.

To begin, let's download [gridStarter.zip](#).



CSS Grids

Two-dimensional system: columns x rows



```
.container {  
  display: grid;  
  grid-template-columns: 25% 50% 25%;  
  grid-template-rows: 150px 300px;  
}
```



CSS Grids :: Creating space (*gutters*)

```
gap: <grid-row-gap> <grid-column-gap>;
```

If you provide one value only, it will apply to both.

```
.container {  
  display: grid;  
  grid-template-columns: 25% 50% 25%;  
  grid-template-rows: 150px 300px;  
  gap: 40px;  
}
```



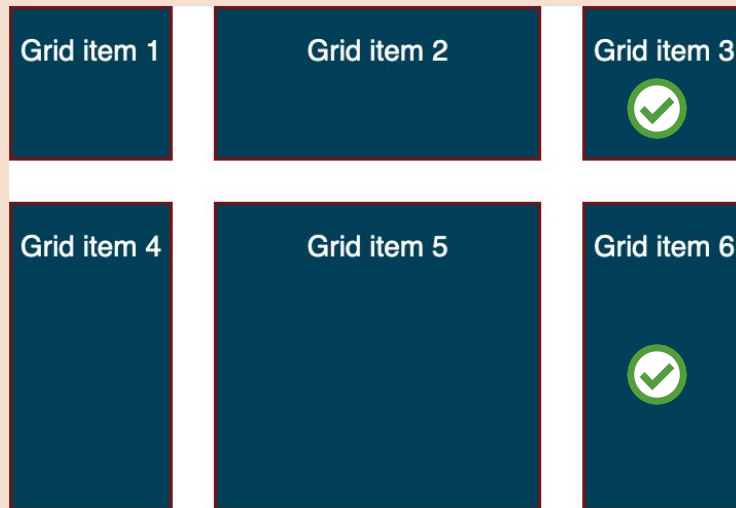
CSS Grids :: fr unit

fr unit: fraction of the free space in the grid container


```
.container {  
  display: grid;  
  grid-template-columns: 1fr 2fr 1fr;  
  grid-template-rows: 150px 300px;  
  gap: 40px;  
}
```

fr is determined **after** any non-flexible columns or rows are declared (similar to flex-grow and flex-shrink).

To test: `grid-template-columns: 1fr 300px 1fr;`



CSS Grids :: Explicit and implicit rows and columns

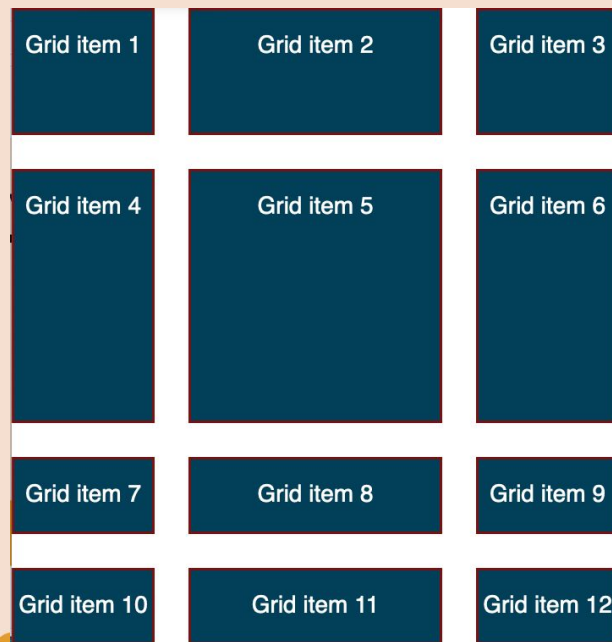
Adding more elements: remove the comments on the next 5 elements. ( /) (CTRL SHIFT /)

CSS Grids will automatically fit your new content in a new row.

The size of the row will be determined by the content that falls to that new row.

These new rows are called ***implicit rows***.

The rows we created initially are called ***explicit rows***.



CSS Grids :: Explicit and implicit rows and columns

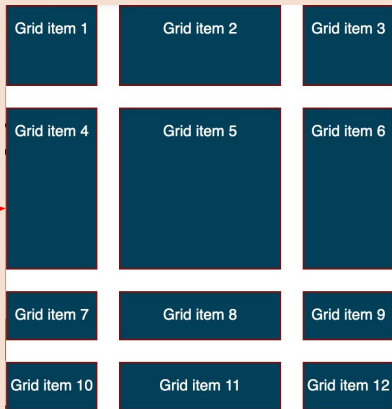
grid-auto-flow

Extra grid items get put in a new **row**, not a new **column** because the default value for the grid-auto-flow is row.

- **row**: default value
- **column**: any implicit items will create new columns
- **dense**: attempts to fill in holes earlier in the grid.



Using grid-auto-flow: dense; will change the order of your grid items, which may create **accessibility challenges**. If the order of your content matters to the user's understanding, don't use this value.



CSS Grids :: Explicit and implicit rows and columns

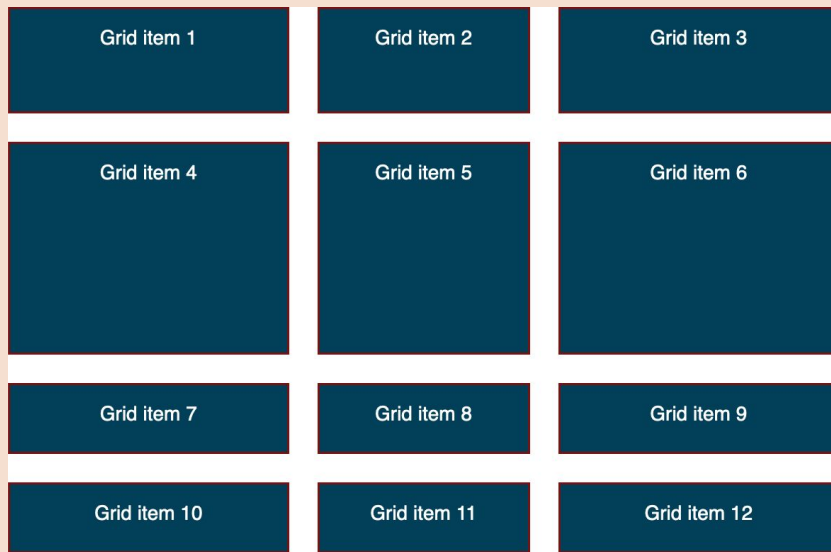
grid-auto-rows: controls implicit rows

It is possible to provide multiple values for multiple implicit rows.



Firefox does not currently support multiple values, so wait until the support is more robust to use it in production.

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 2fr 1fr;  
  grid-template-rows: 150px 300px;  
  gap: 40px;  
  grid-auto-rows: 100px;  
}
```



CSS Grids :: Explicit and implicit rows and columns

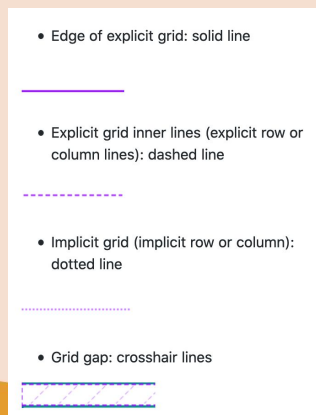
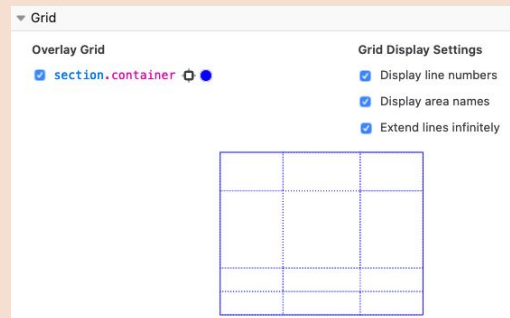
grid-auto-columns: controls implicit columns when grid-auto-flow is set to column.

⚠ Again, it is possible to provide multiple values for multiple implicit rows, but again Firefox does not currently support multiple values, so wait until the support is more robust to use it in production.

```
.container {  
  display: grid;  
  grid-template-columns: 1fr 300px 1fr;  
  grid-template-rows: 150px 300px;  
  gap: 40px;  
  grid-auto-flow: column;  
  grid-auto-columns: 150px;  
}
```



CSS Grids :: Dev Tools



CSS Grids :: Grid Item Placement

grid-column-start and **grid-column-end** / **grid-row-start** and **grid-row-end**:

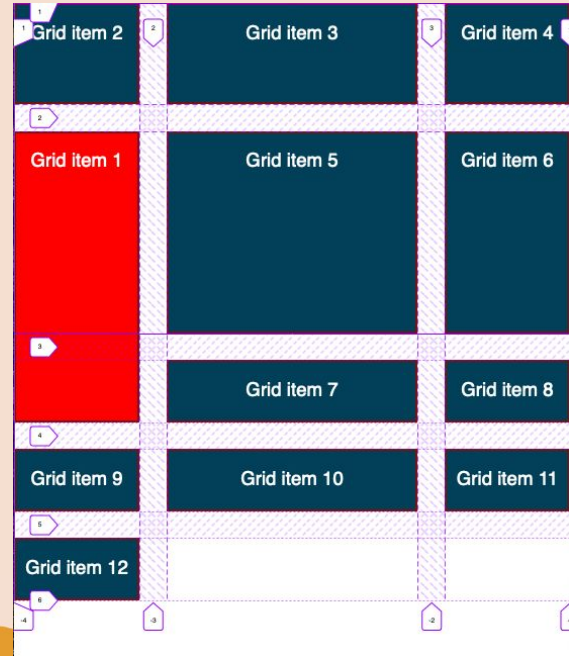
Place any child elements in specific grid locations.

-**start** properties refer to the grid line where you want your element to start

-**end** properties refer to the grid line where you want element to end (not inclusive).

Using numbers:

```
.gridItem1 {  
  grid-row-start: 2;  
  grid-row-end: 4;  
  background-color: red;  
}
```



CSS Grids :: Grid Item Placement

grid-column-start and **grid-column-end** / **grid-row-start** and **grid-row-end**:

Place any child elements in specific grid locations.

-**start** properties refer to the grid line where you want your element to start

-**end** properties refer to the grid line where you want element to end (not inclusive).

Using **span <number>**:

```
.gridItem2 {  
  grid-column-start: span 3;  
  background-color: yellow;  
}
```



CSS Grids :: Grid Item Placement

grid-column-start and **grid-column-end** / **grid-row-start** and **grid-row-end**:

Place any child elements in specific grid locations.

-**start** properties refer to the grid line where you want your element to start

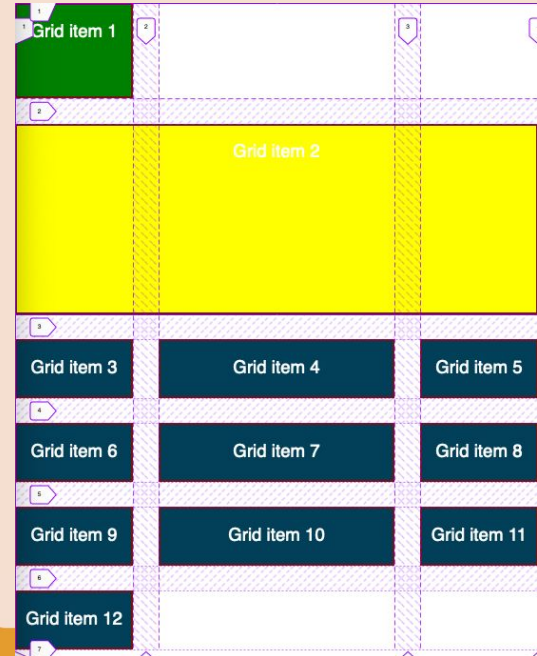
-**end** properties refer to the grid line where you want element to end (not inclusive).

Using auto: will default to the automatic span of one

```
.gridItem1 {  
  /* grid-row-start: 2;  
  grid-row-end: 4; */  
  /* background-color: red; */  
  grid-row-start: auto;  
  background-color: green;  
}
```

Additional learning topics:

- custom grid line name
- span < grid line name >

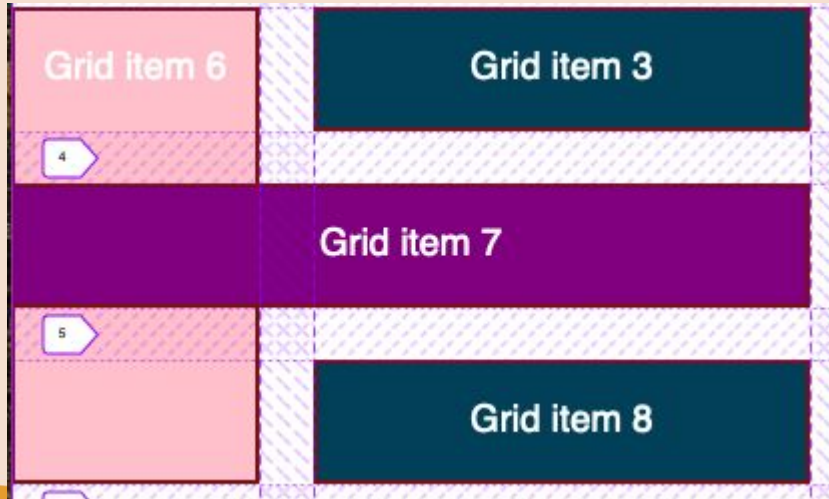


CSS Grids :: Overlapping with Grid

By default grid will shift content to avoid items overlapping.

If an overlapping grid item is in your design, then every item in the grid will need to have an **explicit grid-...-start** and **grid-...-end** values.

```
.gridItem6 {  
  grid-row-start: 3;  
  grid-row-end: 6;  
  grid-column-start: 1;  
  background-color: pink;  
}  
  
.gridItem7 {  
  grid-row-start: 4;  
  grid-column-start: 1;  
  grid-column-end: 3;  
  background-color: purple;  
}
```



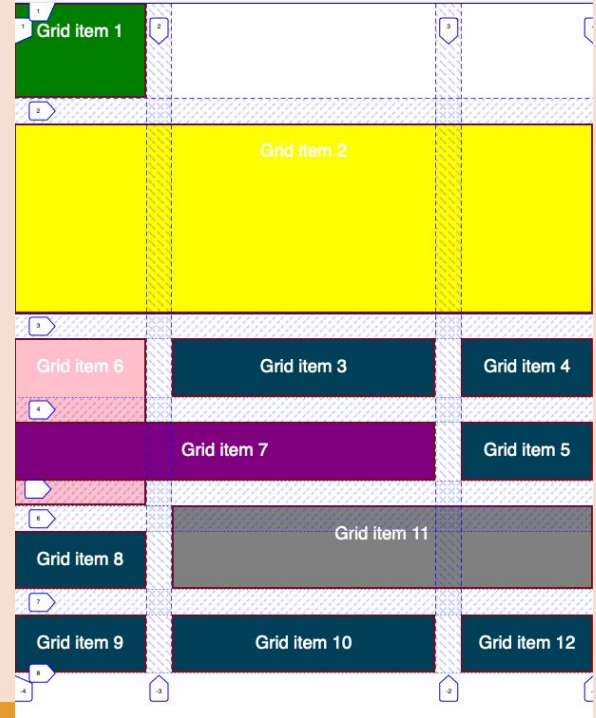
CSS Grids :: Shorthand placement

- **grid-column:** <grid-column-start> / <grid-column-end>
- **grid-row:** <grid-row-start> / <grid-row-end>
- **grid-area:** <grid-row-start> / <grid-column-start> / <grid-row-end> / <grid-column-end>

```
.gridItem1 {  
  grid-column: 1 / -1;  
}
```

```
.gridItem1 {  
  grid-column: 1 / span 3;  
}
```

```
.gridItem11 {  
  grid-area: 5 / 2 / 7 / 4;  
  background-color: gray;  
}
```



CSS Grids :: Repeat keyword

repeat: used inside of **grid-template-columns** and **grid-template-rows** to repeat the value for a number of rows or columns. Requires at least 2 values:

- a positive number (number of columns or rows to create) OR a keyword (**auto-fill*** or **auto-fit***)
- size of the columns or rows to be created. (pixels or fr units)

```
/* Three columns of 50px each */  
grid-template-columns: repeat(3, 50px);  
/* same as saying: */  
grid-template-columns: 50px 50px 50px;
```

```
/* Two rows of 1fr each */  
grid-template-rows: repeat(2, 1fr);  
/* same as saying: */  
grid-template-rows: 1fr 1fr;
```

More than one
value to repeat:

```
grid-template-columns: repeat(4, 1fr 2fr);  
/* same as saying: */  
grid-template-columns: 1fr 2fr 1fr 2fr 1fr 2fr 1fr 2fr;
```

CSS Grids :: Bonus – Adding text and pictures

```
.gridItem {  
  text-align: center;  
  color: white;  
  border: 3px solid #791313;  
  background: #023f58;  
  /* overflow: auto; */  
}
```

Grid item 1 Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque tempora iste, corporis distinctio porro ducimus quas doloremque aut sed earum?

Grid item 3

Grid item 5

Grid item 6

Grid item 7



```
.gridItem4 {  
  grid-column-start: 2;  
  grid-column-end: 2;  
  grid-row-start: 2;  
  grid-row-end: 4;  
}
```

```
img {  
  width: 100%;  
  height: 100%;  
  object-fit: cover;  
  margin: 0;  
}
```

```
<div class="gridItem gridItem4">  
    
</div>
```

CSS Grids :: Grid Garden

GRID GARDEN


< Level 1 of 28 >

Welcome to Grid Garden, where you write CSS code to grow your carrot garden! Water only the areas that have carrots by using the `grid-column-start` property.

For example, `grid-column-start: 3;` will water the area starting at the 3rd vertical grid line, which is another way of saying the 3rd vertical border from the left in the grid.

```
1 #garden {
2   display: grid;
3   grid-template-columns: 20% 20% 20% 20%;
4   20%;
5   grid-template-rows: 20% 20% 20% 20%;
6 }
7
8 #water {
9   grid-column-start: 3;
10 }
11
12
13
14
```

Next



GRID GARDEN


< Level 16 of 28 >

If typing out both `grid-column` and `grid-row` is too much for you, there's yet another shorthand for that. `grid-area` accepts four values separated by slashes: `grid-row-start, grid-column-start, grid-row-end, grid-column-end`, followed by `grid-column-end`.

One example of this would be `grid-area: 1 / 1 / 3 / 6;`.

```
1 #garden {
2   display: grid;
3   grid-template-columns: 20% 20% 20% 20%;
4   20%;
5   grid-template-rows: 20% 20% 20% 20%;
6 }
7
8 #water {
9   grid-area: 1 / 2 / 4 / 6;.
10 }
11
12
13
14
```

Next

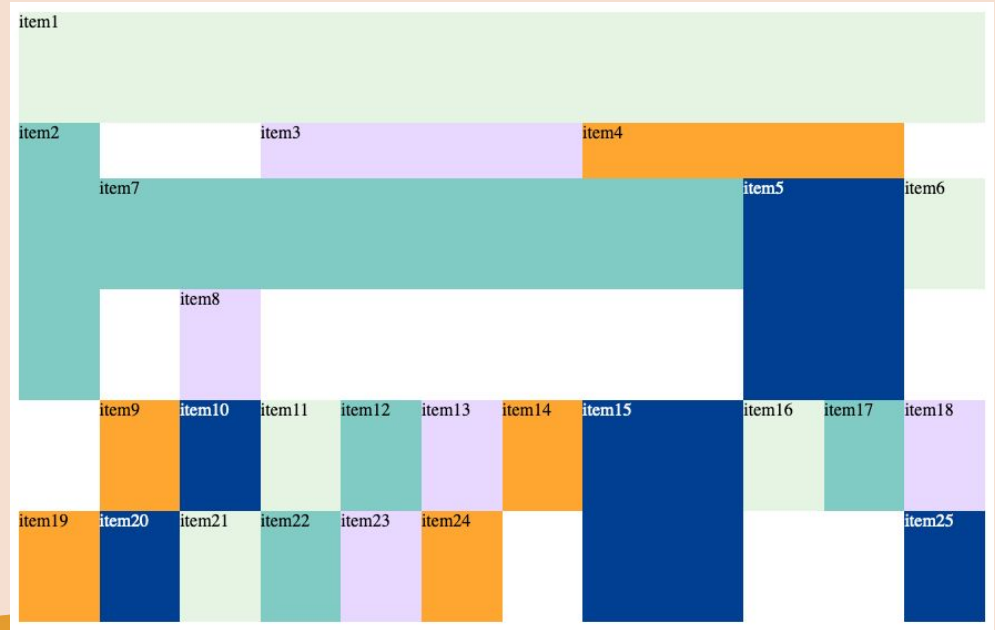


CSS Grids :: Exercise

Try this exercise: [grid-placement-exercise-start.html](https://junocollge.com/grid-placement-exercise-start.html). The answer key is available [here](https://junocollge.com/grid-placement-exercise-answer-key.html).

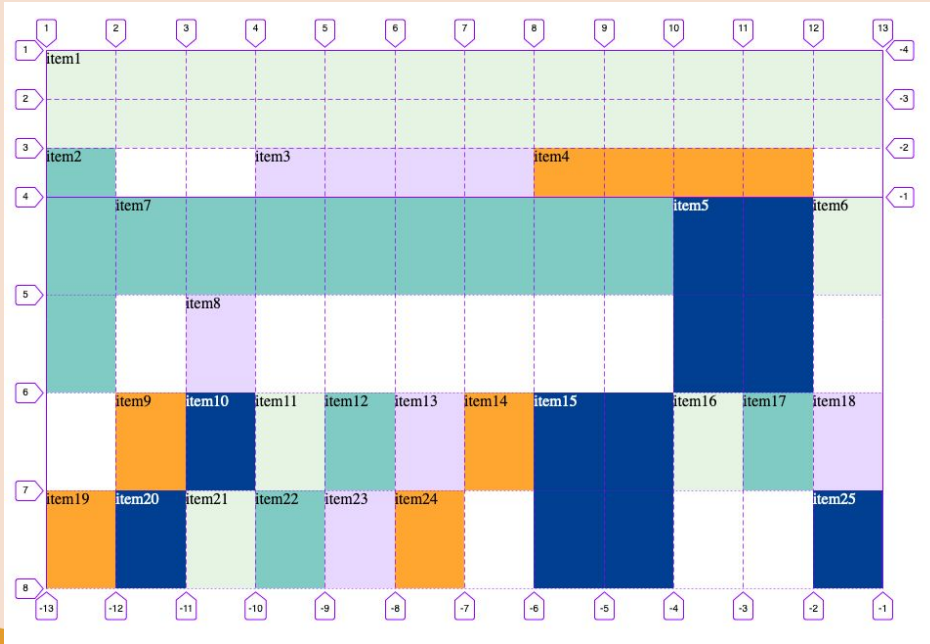


- Breakout rooms in pairs
- One shares the screen
- Create a plan
- Alternate suggestions for each box



CSS Grids :: Exercise

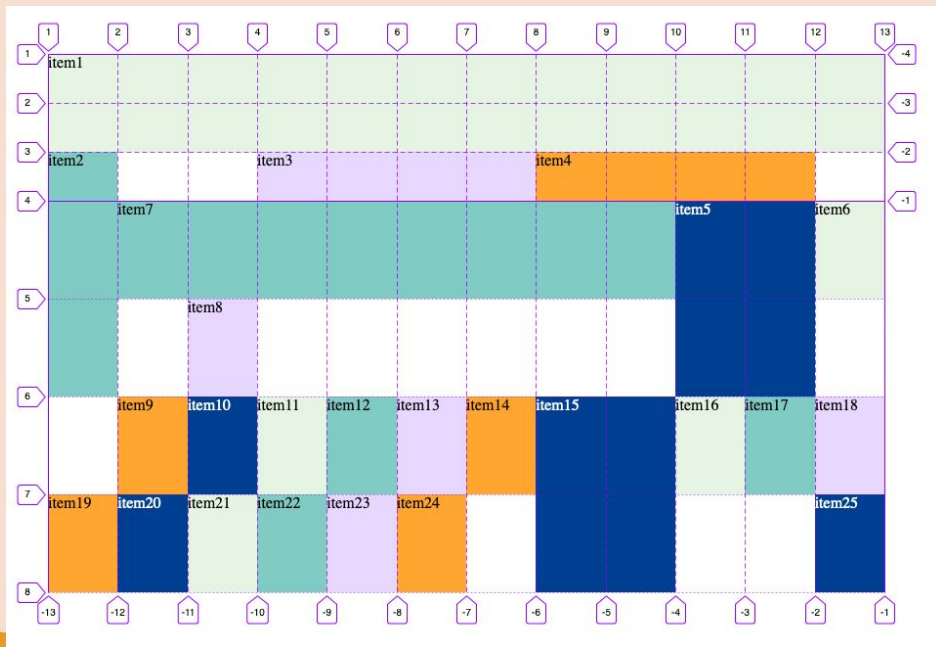
Try this exercise: [grid-placement-exercise-start.html](https://junocollge.com/grid-placement-exercise-start.html). The answer key is available [here](#).



```
/* create a grid with:  
  12 columns of even space  
  3 explicit rows that are 50px tall  
  implicit rows that are 100px tall  
*/  
  
.grid {  
  margin: 100px;  
  display: grid;  
  grid-template-columns: repeat(12, 1fr);  
  grid-template-rows: repeat(3, 50px);  
  grid-auto-rows: 100px;  
}
```

CSS Grids :: Exercise

Try this exercise: [grid-placement-exercise-start.html](https://junedev.com/grid-placement-exercise-start.html). The answer key is available [here](https://junedev.com/grid-placement-exercise-answer.html).



```
/* make the first item span the whole width of
the grid, and be 2 columns tall */
.item1 {
  grid-column: 1 / -1;
  grid-row: span 2;
}

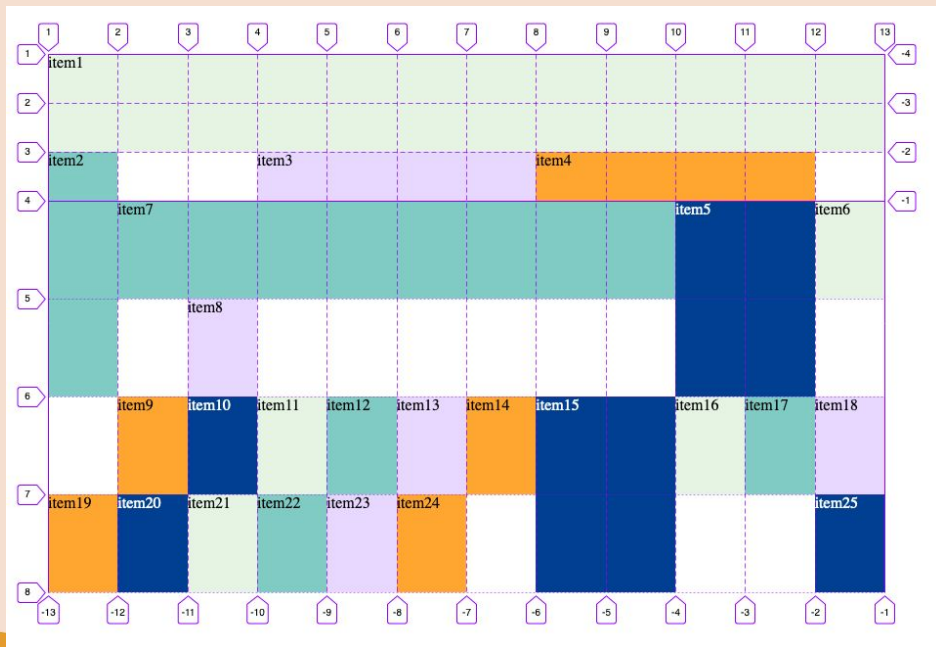
/* make the second item be three columns tall */
.item2 {
  grid-row: span 3;
}

/* make the third item span four columns, but end
at line 8 */
.item3 {
  grid-column: span 4 / 8;
}

/* make the fourth item span four columns, but
start at line 8 */
.item4 {
  grid-column: 8 / span 4;
}
```

CSS Grids :: Exercise

Try this exercise: [grid-placement-exercise-start.html](https://www.junoblog.com/css-grid-placement-exercise-start.html). The answer key is available [here](https://www.junoblog.com/css-grid-placement-exercise-answer-key.html).



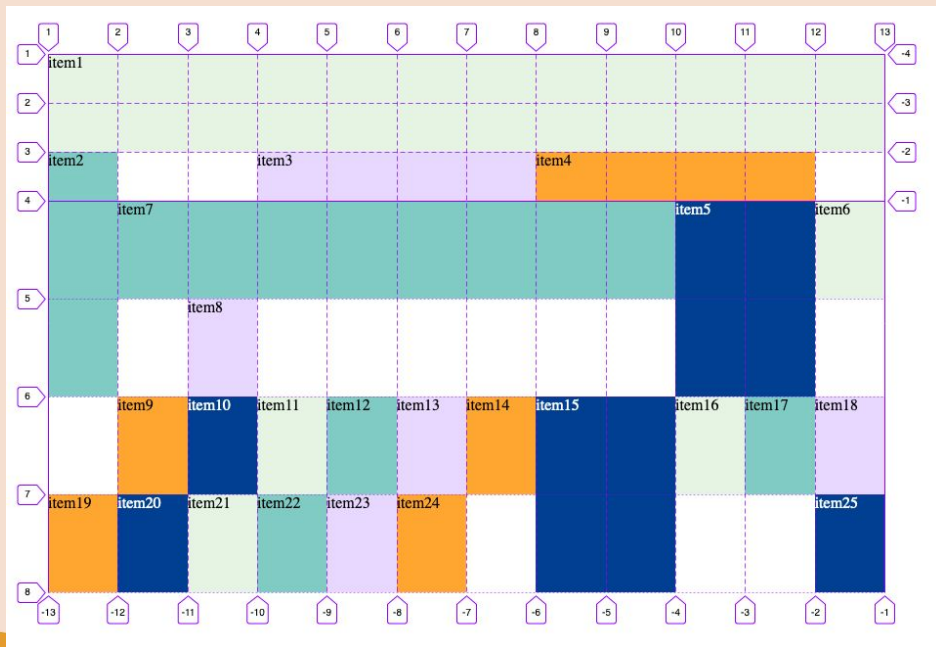
```
/* make the fifth item span 2 columns and 2 rows,
but end at the second to last column */
.item5 {
  grid-column: span 2 / -2;
  grid-row: span 2;
}

/* make the seventh item fill in the gap left by
item5
  columns: start at line 2 and end at line 10
  (-4 if you want!)
  rows: start at line 4 and span 1 row
*/
.item7 {
  grid-column: 2 / 10;
  grid-row: 4;
}

/* make item8 start at line 3 */
.item8 {
  grid-column-start: 3;
}
```

CSS Grids :: Exercise

Try this exercise: [grid-placement-exercise-start.html](https://junoconge.com/grid-placement-exercise-start.html). The answer key is available [here](https://junoconge.com/grid-placement-exercise-answer.html).



```
/* make item9 start at line 2 */
.item9 {
  grid-column: 2;
}

/* make item15 start at column 8 and span 2
columns, and start at row 6 and go until the end
this one is tricky! */
.item15 {
  grid-column: 8 / span 2;
  grid-row: 6 / 8;
}

/* make the last item (25) end at the last column
*/
.item25 {
  grid-column-end: -1;
}
```