CSS – Notes from Week 8 Recap

Contents

Grid Garden 1
- Columns: 2
  - grid-column-start 2
  - grid-column-end 3
  - Negative Values 3
  - Span 4
  - grid-column 5
- Rows: 6
  - grid-row-start 6
  - grid-row-end 6
  - grid-row 6
- Combining column and row 6
  - Example: spanning multiple columns and rows 7
- grid-area 7
- Multiple items can overlap 8
- Order 9
  - Using a negative number 9
- Grid-template-columns 10
  - Repeat function 10
  - Using different length units 10
- grid-template 12
  - Final challenge 12

Grid Garden

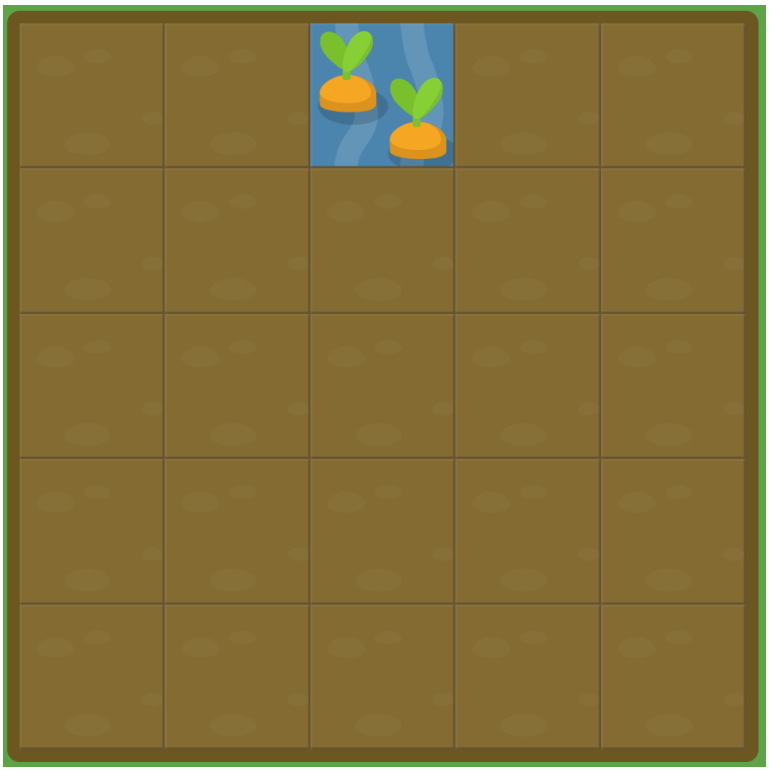
https://cssgridgarden.com/ - watering crops and poisoning weeds in the grid garden.

Starts with pre-defined CSS of:

```
#garden {
  display: grid;
  grid-template-columns: 20% 20% 20% 20% 20%;
```

```
  grid-template-rows: 20% 20% 20% 20% 20%;
}
```

The above code is selecting an element with an ID of "garden", turning it in to a grid and creating 5 rows and columns that are equally spaced (20% each). This creates this grid:



Remember the lines start counting from the very edge of the box – so the left hand border is column 1!!

Columns:

grid-column-start

Solved the 1st level using:

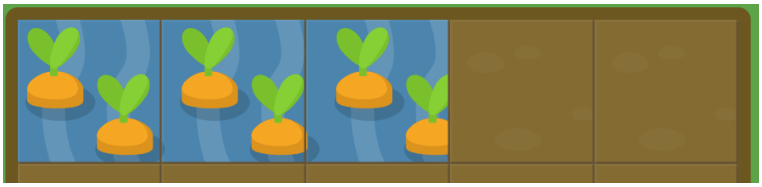
```
#water {
        grid-column-start: 3;
}
```

The above code moves the element with an ID of "water" to the 3rd vertical grid line, which is another way of saying the 3rd vertical border from the left in the grid.

grid-column-end

When **grid-column-start** is used alone, the grid item by default will span exactly one column. However, you can extend the item across multiple columns by adding the **grid-column-end** property.

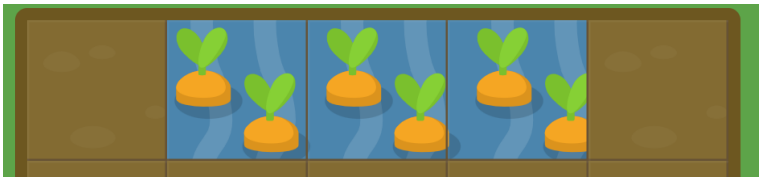
You need to include the grid column you are going up to!



Solution code for the above:

```
#water {
        grid-column-start: 1;
        grid-column-end: 4;
}
```

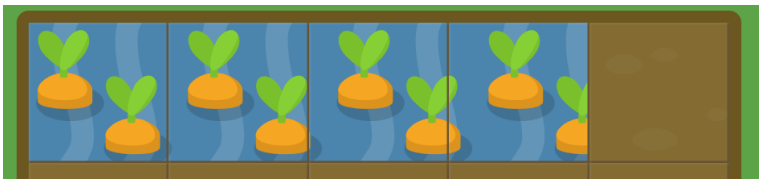
You can also start from the end and work backwards!



```
#water {
        grid-column-start: 5;
        grid-column-end: 2;
}
```

Negative Values

If you want to count grid lines from the right instead of the left, you can give **grid-column-start** and **grid-column-end** negative values. For example, you can set it to -1 to specify the first grid line from the right:

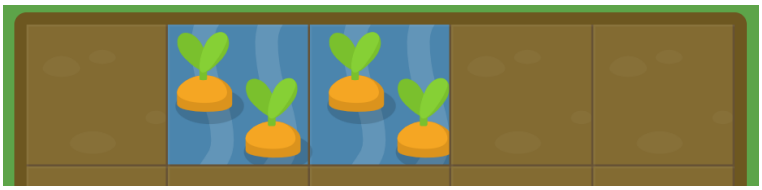


```
#water {
	grid-column-start: 1;
	grid-column-end: -2;
}
```

```
#water {
	grid-column-start: -3;
}
```

Span

Instead of defining a grid item based on the start and end positions of the grid lines, you can define it based on your desired column width using the **span** keyword. Keep in mind that **span** only works with positive values.



```
#water {
	grid-column-start: 2;
	grid-column-end: span 2;
}
```

You can also use the **span** keyword with **grid-column-start** to set your item's width relative to the end position.

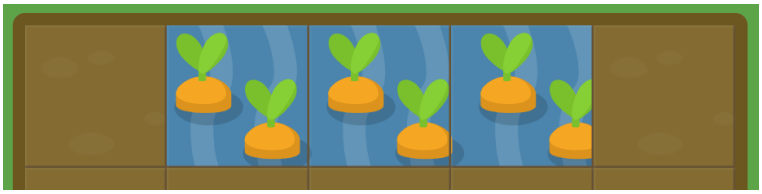
```
#water {
        grid-column-start: span 3;
        grid-column-end: 6;
}
```

grid-column

Typing both grid-column-start and grid-column-end every time can get tiring. Fortunately, **grid-column** is a shorthand property that can accept both values at once, separated by a slash. For example, **grid-column: 2 / 4;** will set the grid item to start on the 2nd vertical grid line and end on the 4th grid line.

```
#water {
        grid-column: 4 / 6;
}
```

The **span** keyword also works with this shorthand property:



```
#water {
        grid-column: 2 / span 3;
}
```

Rows:

grid-row-start

One of the things that sets CSS grids apart from flexbox is that you can easily position items in two dimensions: columns and rows. **grid-row-start** works much like **grid-column-start** except along the vertical axis.

```
#water {
        grid-row-start: 3;
}
```

grid-row-end

Works like grid-column-end as above.

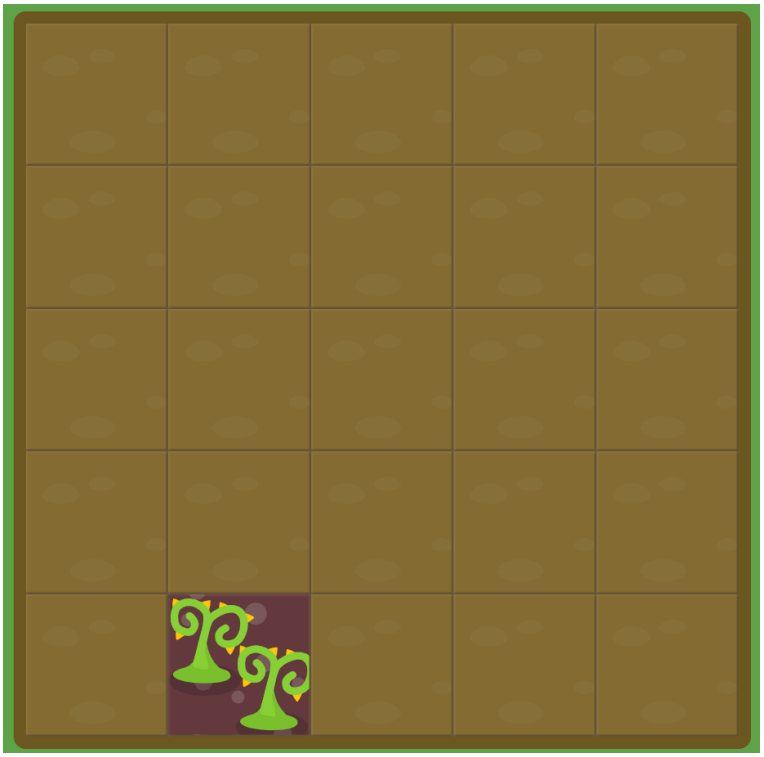
grid-row

Shorthand property to take both start and end at the same time, separated by a slash, the same way as grid-column.

N.B. Don't forget you can use span with rows in the same way as columns.

Combining column and row

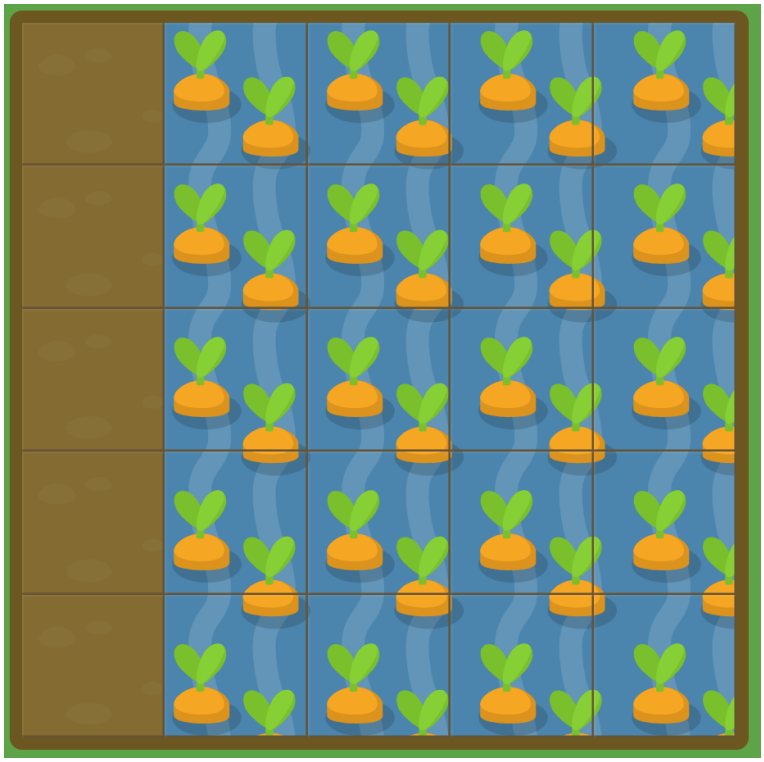
We then began to practise combining both column and row, using only the short hand property. If you just put 1 number after grid-column or grid-row, that's the same as writing grid-column-start or grid-row-start.



```
#water {
        grid-column: 2;
        grid-row: 5;
}
```

Example: spanning multiple columns and rows

```
#water {
        Grid-column: 2 / span 4;
        Grid-row: 1 / span 6;
}
```

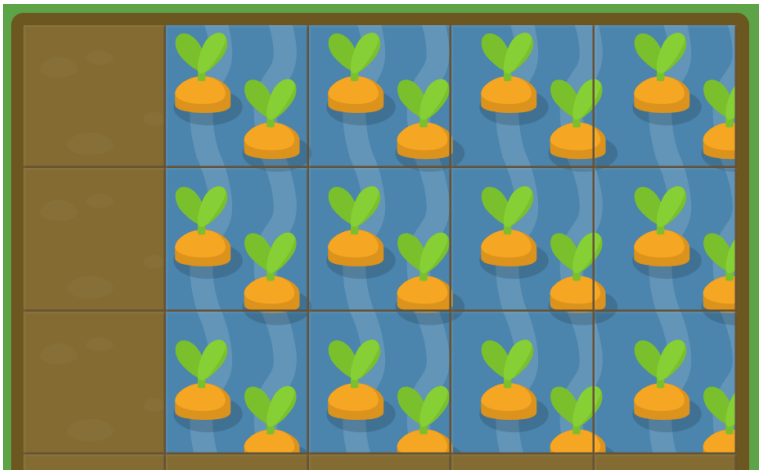


grid-area

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**, followed by **grid-column-end**.

e.g.

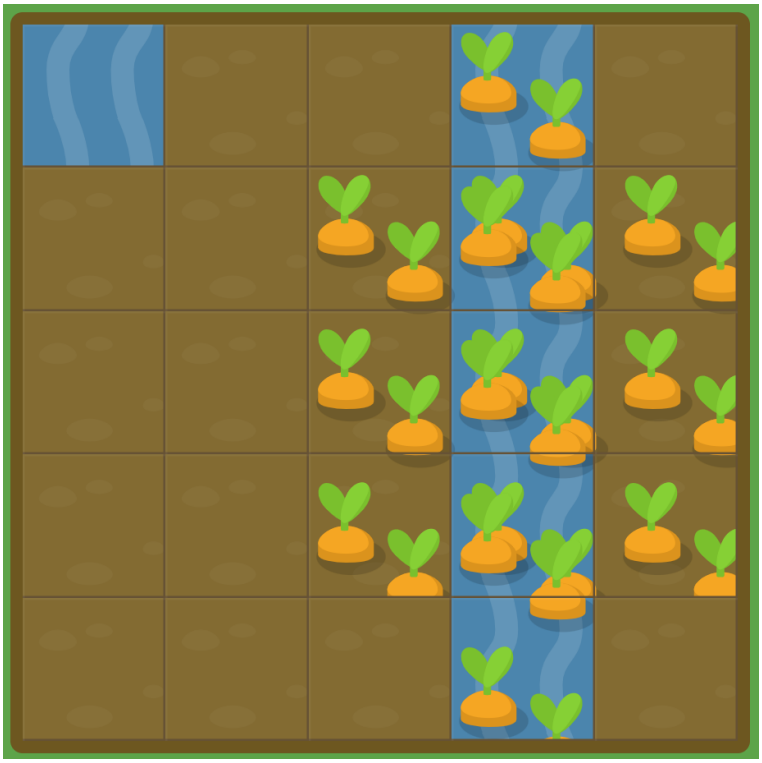
```
#water {
        Grid-area: 1 / 2 / span 3 / span 4;
}
```



Multiple items can overlap

Start with this code:

```
#water-1 {
  grid-area: 1 / 4 / 6 / 5;
}
```



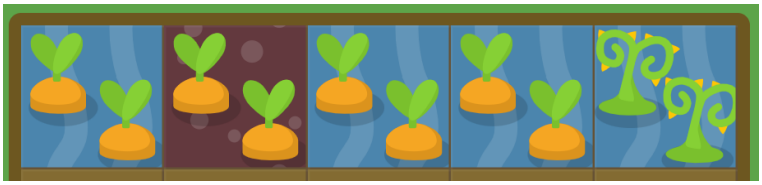
Then add this code to overlap and complete:

```
#water-2 {
  Grid-area: 2 / 3 / span 3 / span 3;
}
```

Order

If grid items aren't explicitly placed with grid-area, grid-column, grid-row, etc., they are automatically placed according to their order in the source code. We can override this using the **order** property, which is one of the advantages of grid over table-based layout.
By default, all grid items have an **order** of 0, but this can be set to any positive or negative value, similar to **z-index**.

In this example, the poison is in the wrong place:



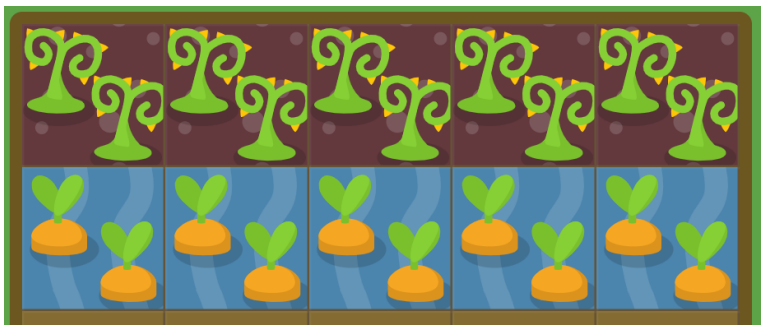
```
.water {
  Order: 0;
}

#poison {
  Order: 1;
}
```

Using a negative number

```
.water {
  order: 0;
}

.poison {
  Order: -1;
}
```



Grid-template-columns

We were given a prefilled grid with 5 columns of 20% width each. You can have as many columns as you want and they don't have to be equally spaced (could be 50% 25% 25% to make 3 columns with the 1st being twice as wide as the 2nd and 3rd)

Repeat function

Use the repeat function as a shorthand to create multiple columns with the same width. So instead of:

grid-template-columns: 20% 20% 20% 20% 20%

you can write:

grid-template-columns: repeat(5, 20%);

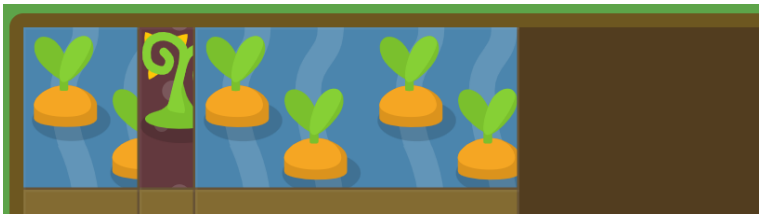
First argument is how many columns, second argument is the width percentage.

Using different length units

grid-template-columns doesn't just accept values in percentages, but also length units like pixel and em. You can even mix different units together. Started with this layout:

```
#garden {
  display: grid;
  grid-template-columns: 100px 3em 40%
}
```

Above code turns it into:



Grid also introduces a new unit, the fractional **fr**. Each **fr** unit allocates one share of the available space. For example, if two elements are set to **1fr** and **3fr** respectively, the space is divided into 4 equal shares; the first element occupies 1/4 and the second element 3/4 of any leftover space.

When columns are set with pixels, percentages, or ems, any other columns set with **fr** will divvy up the space that's left over.

Here the carrots form a 50 pixel column on the left, and the weeds a 50 pixel column on the right. With **grid-template-columns**, create these two columns, and use **fr** to make three more columns that take up the remaining space in between.

```
#garden {
  display: grid;

  grid-template-rows: 20% 20% 20% 20% 20%;
}

#water {
  grid-area: 1 / 1 / 6 / 2;
}

#poison {
  grid-area: 1 / 5 / 6 / 6;
}

```

Using this code to divide it as per instructions above:

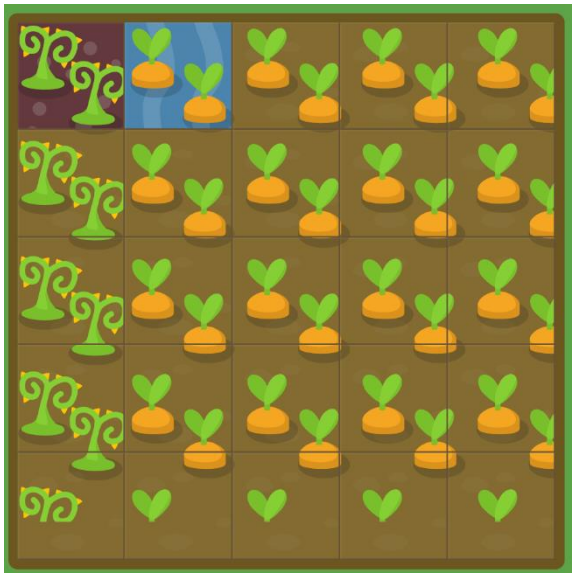
grid-template-columns: 50px 1fr 1fr 1fr 50px;

grid-template

grid-template is a shorthand property that combines grid-template-rows and grid-template-columns.
For example, **grid-template: 50% 50% / 200px;** will create a grid with two rows that are 50% each, and one column that is 200 pixels wide.

Final challenge

Your garden is looking great. Here you've left a 50-pixel path at the bottom of your garden and filled the rest with carrots.
Unfortunately, the left 20% of your carrots have been overrun with weeds. Use CSS grid one last time to treat your garden.



```
#garden {
  display: grid;
  grid-template: 1fr 50px / 20% 1fr
}
```