GRID GARDEN

Level 1 of 28

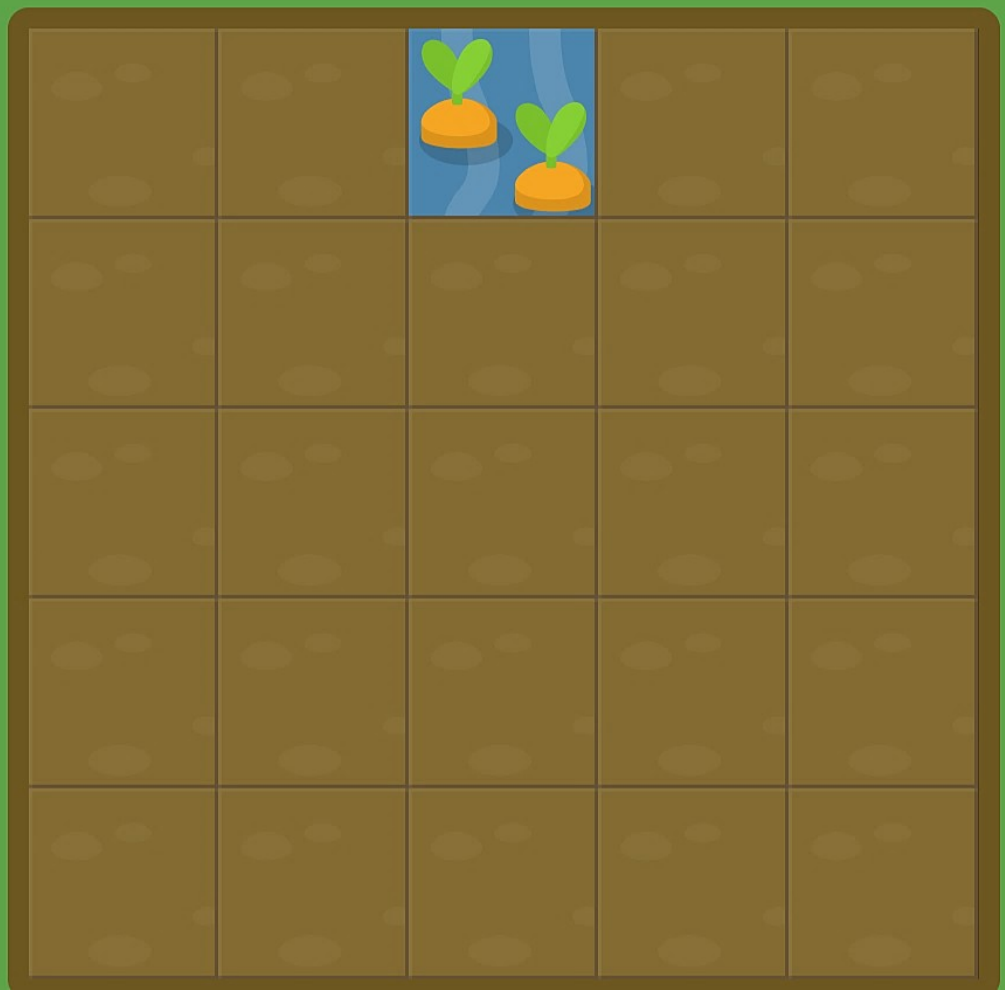
Welcome to Grid Garden, where you write CSS code to grow your carrot garden! Water only the areas that have carrots by using the `grid-column-start` property.

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.

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

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

Next



GRID GARDEN

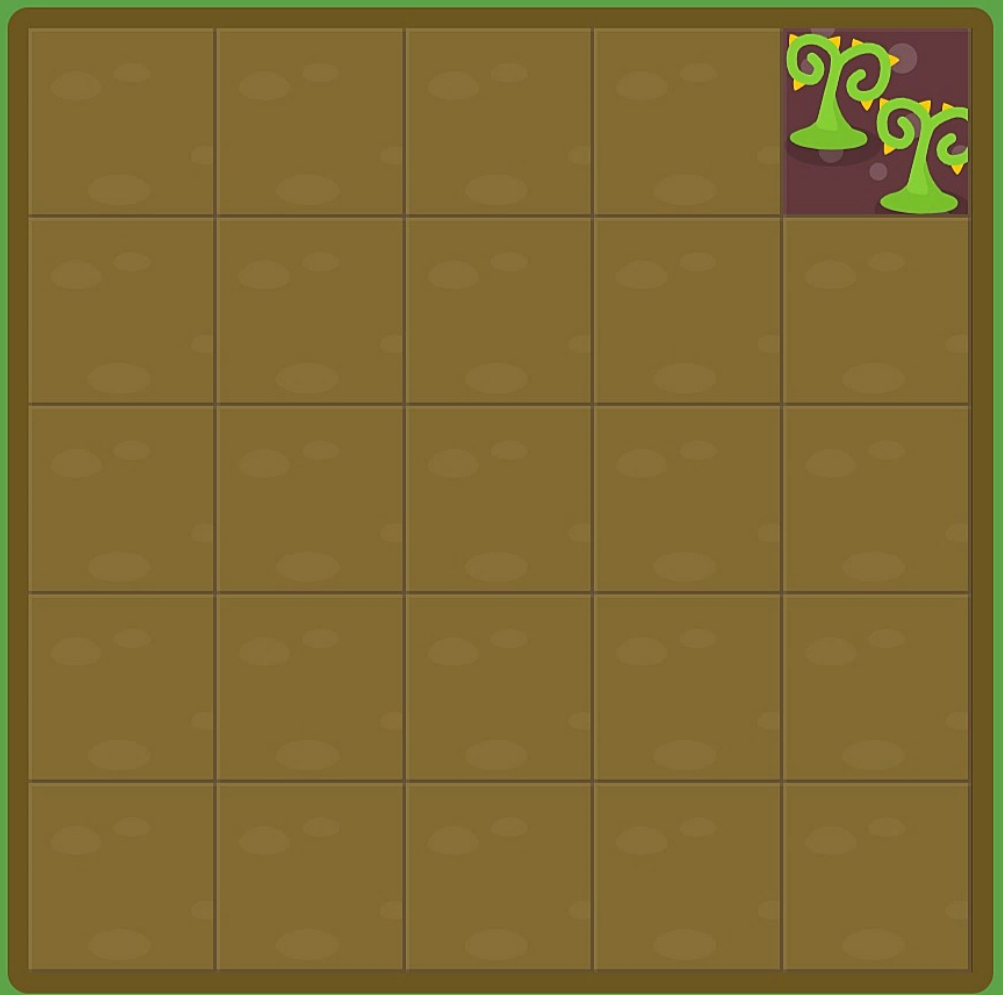
Level 2 of 28

Uh oh, looks like weeds are growing in the corner of your garden. Use `grid-column-start` to poison them. Note that the weeds start at the 5th vertical grid line.

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

#poison {
  grid-column-start:5;
}
```

Next



GRID GARDEN

Level 3 of 28

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.

Using `grid-column-end`, water all of your carrots while avoiding the dirt. We don't wa[illegible]to waste any water! Note that the carrots start at the 1st vertical grid line and er[illegible]

Specifies a grid item's end position within the grid columns.

`<integer>` `span <integer>`

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

GRID GARDEN

Level 4 of 28

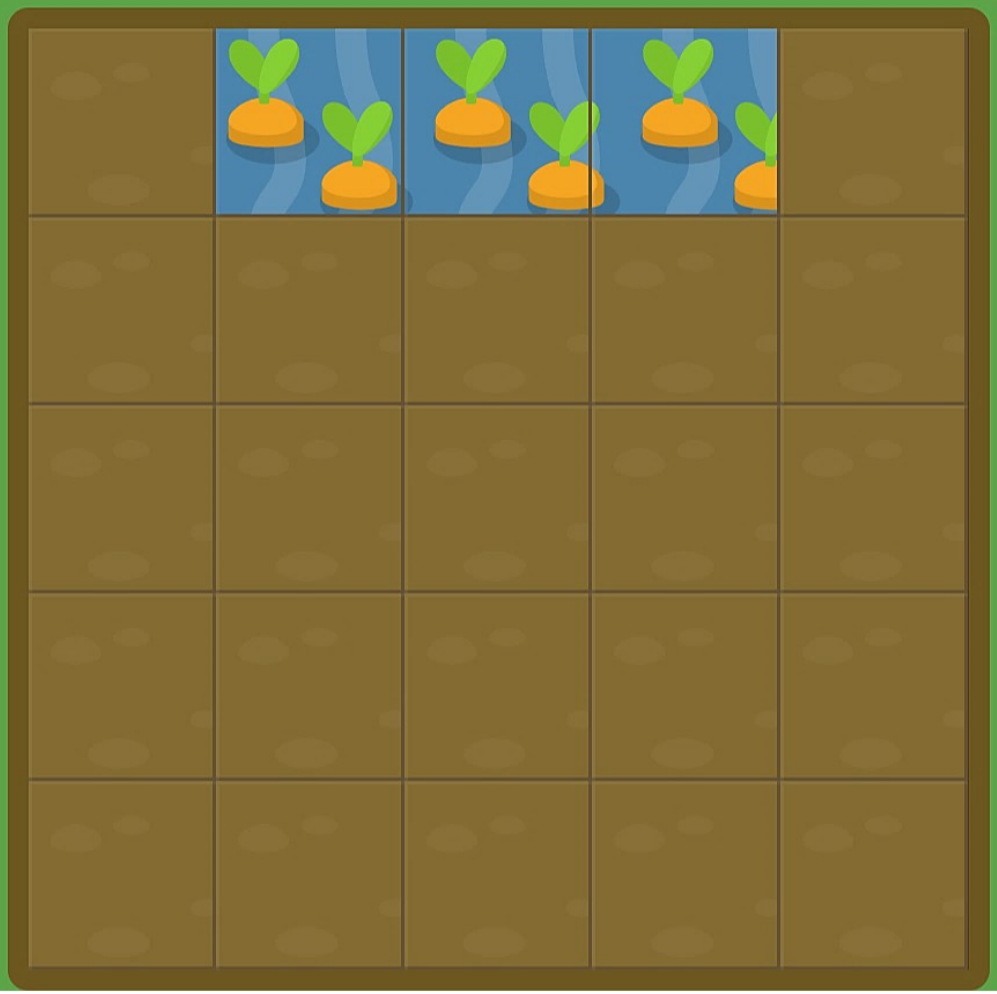
When pairing `grid-column-start` and `grid-column-end`, you might assume that the end value has to be greater than the start value. But this turns out not the case!

Try setting `grid-column-end` to a value less than 5 to water your carrots.

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

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

Next



GRID GARDEN

Level 5 of 28

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.

Try setting `grid-column-end` to a negative value.

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

Next

GRID GARDEN

Level 6 of 28

Now try setting `grid-column-start` to a negative value.

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

#poison {
  grid-column-start:4;
}
```

Next

Grid Garden is created by Codepip • YouTube • Twitter • GitHub • English

Want to learn CSS flexbox? Play Flexbox Froggy.

GRID GARDEN

Level 7 of 28

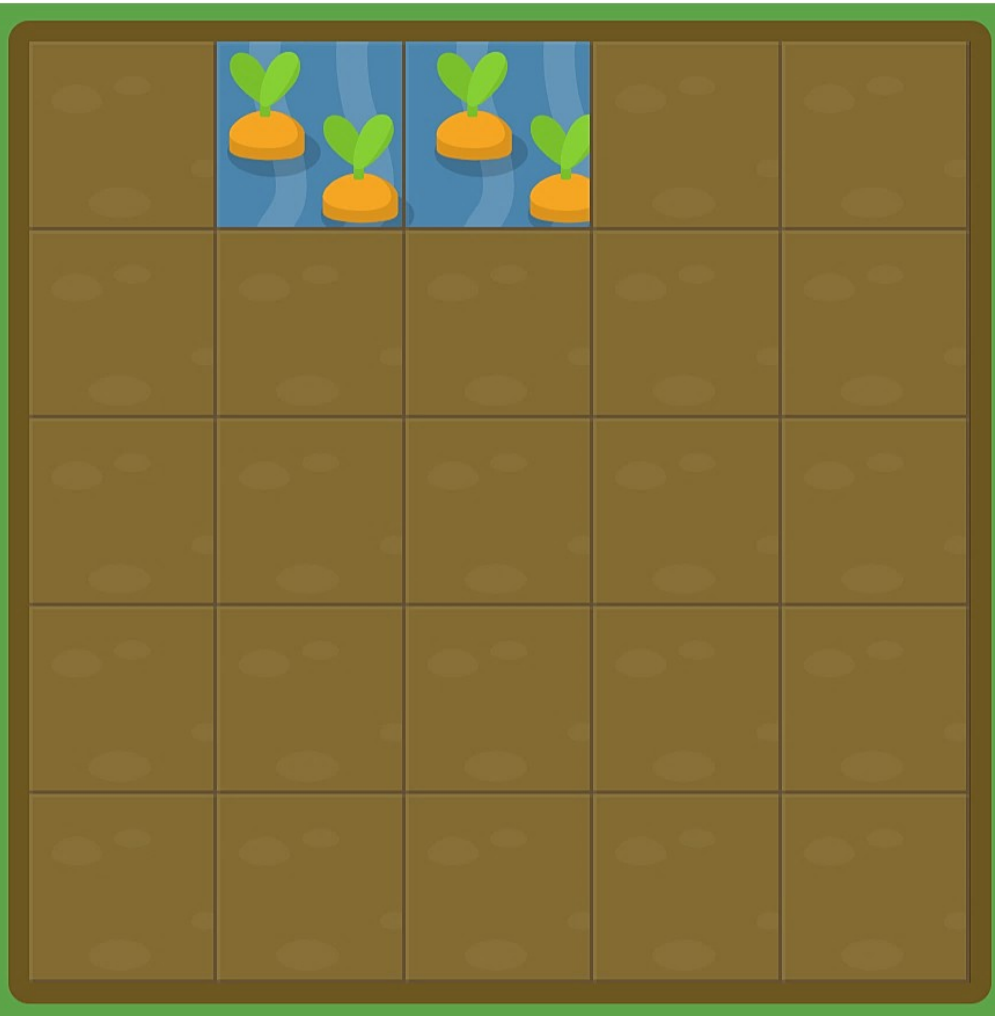
Instead of defining a grid item based on the start and end positions of the grid lines, you can define it based on your desired column width using the `span` keyword. Keep in mind that `span` only works with positive values.

For example, water these carrots with the rule `grid-column-end: span 2;`.

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

#water {
  grid-column-start: 2;
  grid-column-end:4;
}
```

Next



GRID GARDEN

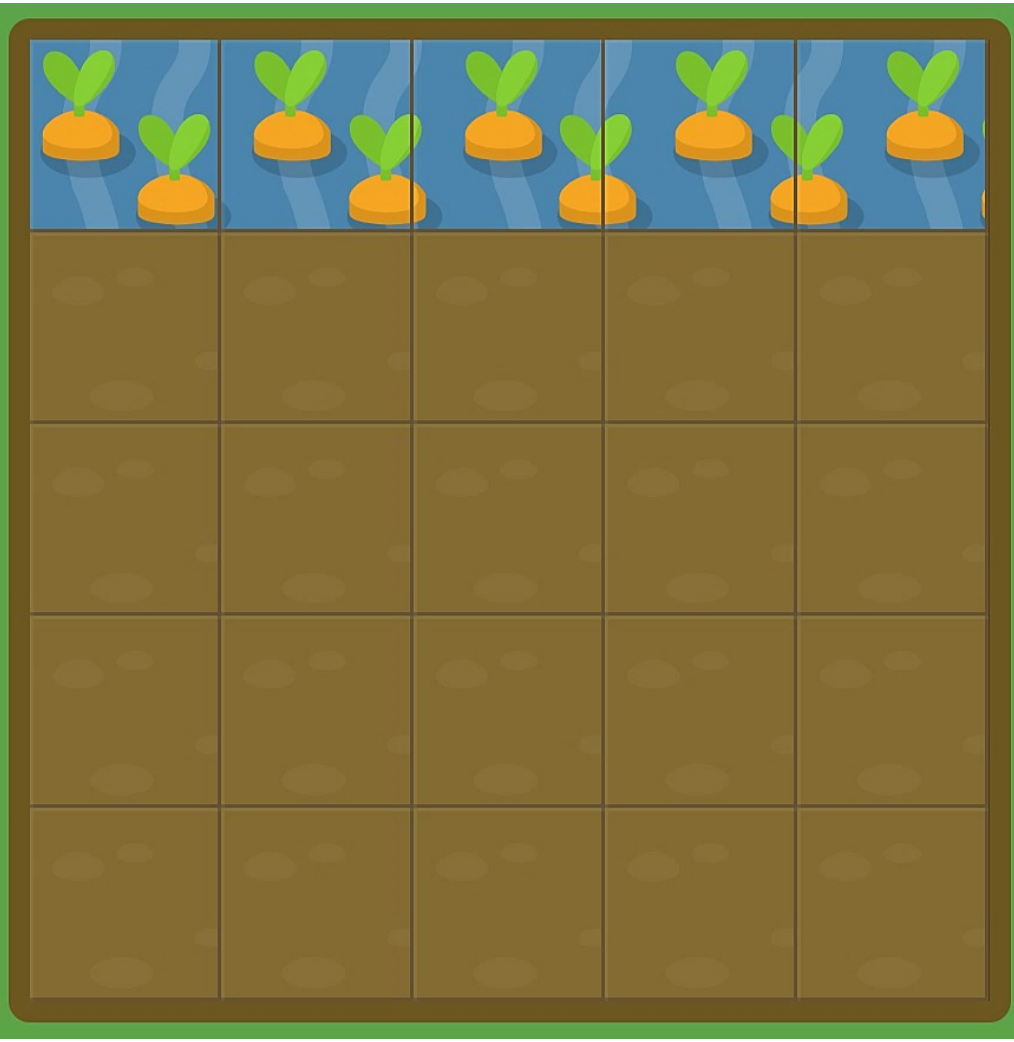
Level 8 of 28

Try using `grid-column-end` with the `span` keyword again to water your carrots.

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

#water {
  grid-column-start: 1;
  grid-column-end:6;
}
```

Next



Grid Garden is created by Codepip • YouTube • Twitter • GitHub • English

Want to learn CSS flexbox? Play Flexbox Froggy.

GRID GARDEN

Level 9 of 28

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

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

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

Next

GRID GARDEN

Level 10 of 28

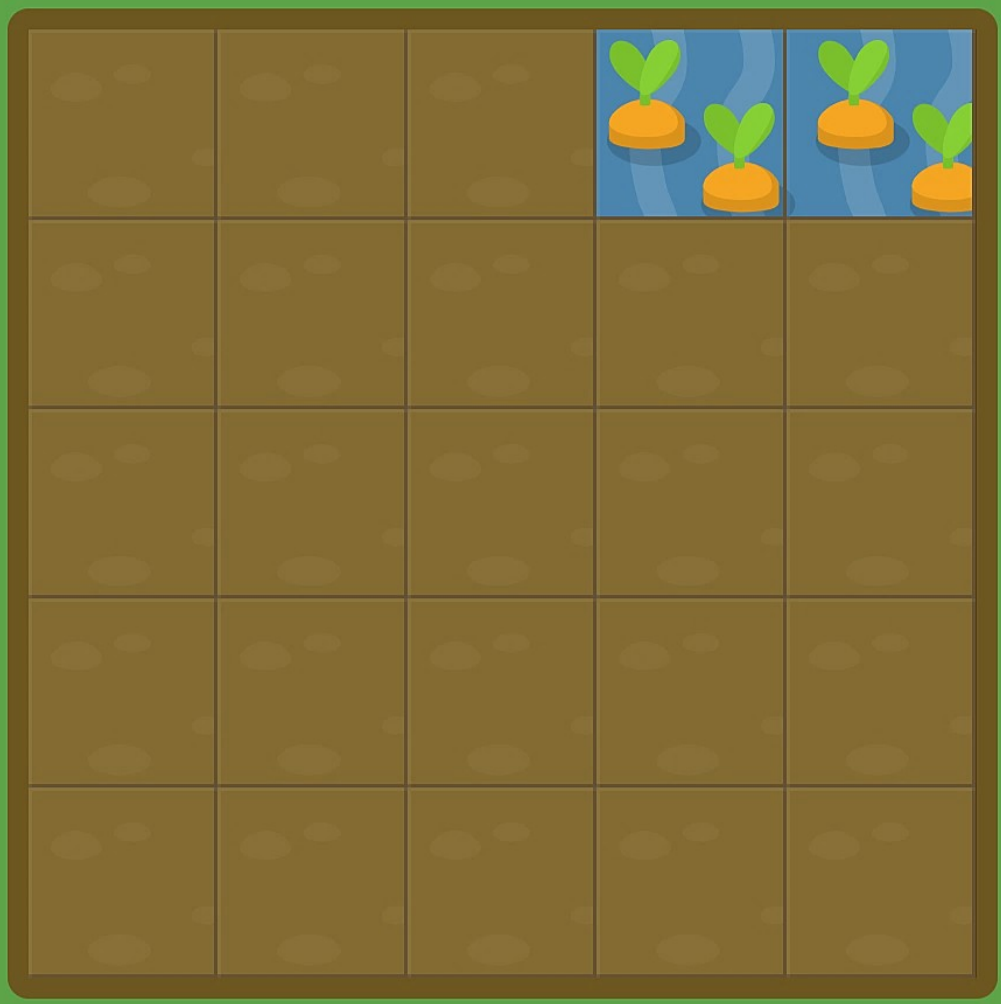
Typing both `grid-column-start` and `grid-column-end` every time can get tiring. Fortunately, `grid-column` is a shorthand property that can accept both values at once, separated by a slash.

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.

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

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

Next



GRID GARDEN

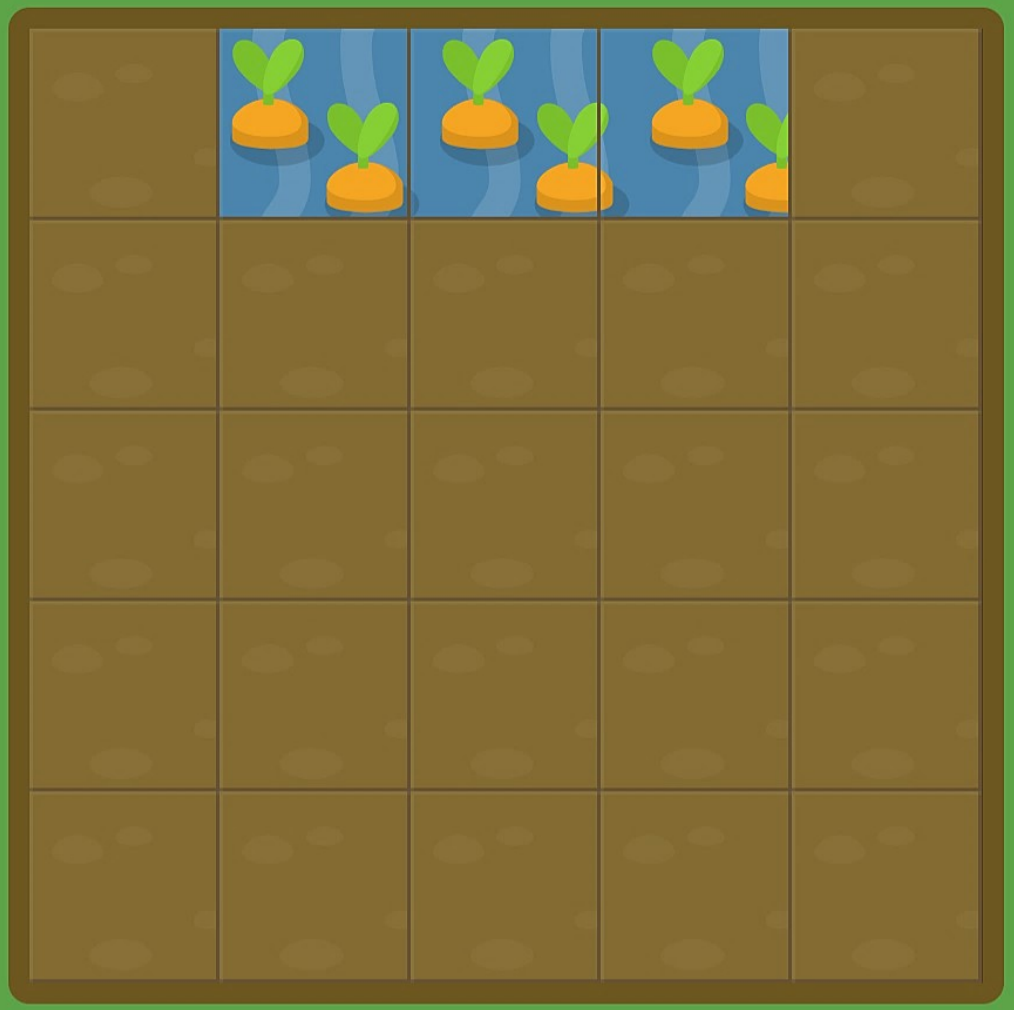
Level 11 of 28

Try using `grid-column` to water these carrots. The `span` keyword also works with this shorthand property so give it a try!

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

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

Next



GRID GARDEN

Level 12 of 28

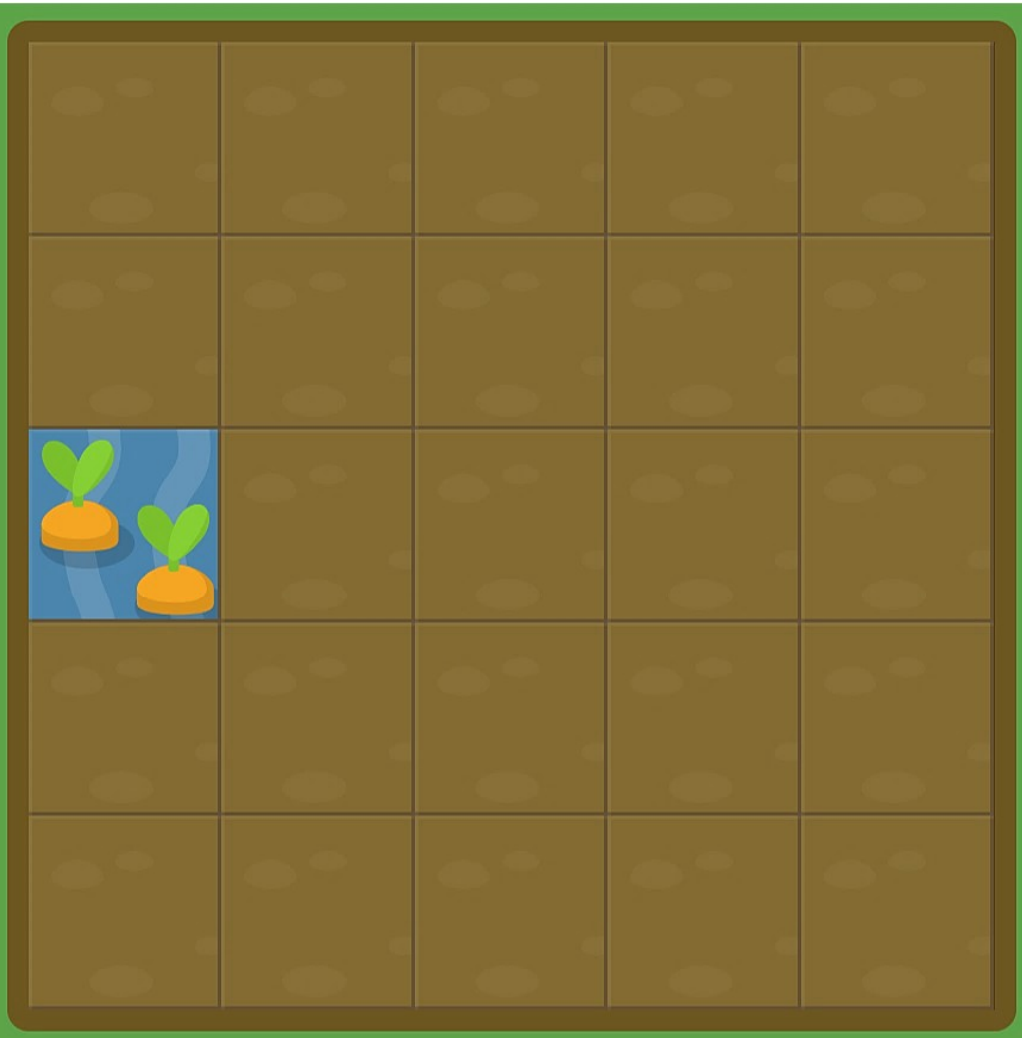
One of the things that sets CSS grids apart from flexbox is that you can easily position items in two dimensions: columns and rows. `grid-row-start` works much like `grid-column-start` except along the vertical axis.

Use `grid-row-start` to water these carrots.

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

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

Next



GRID GARDEN

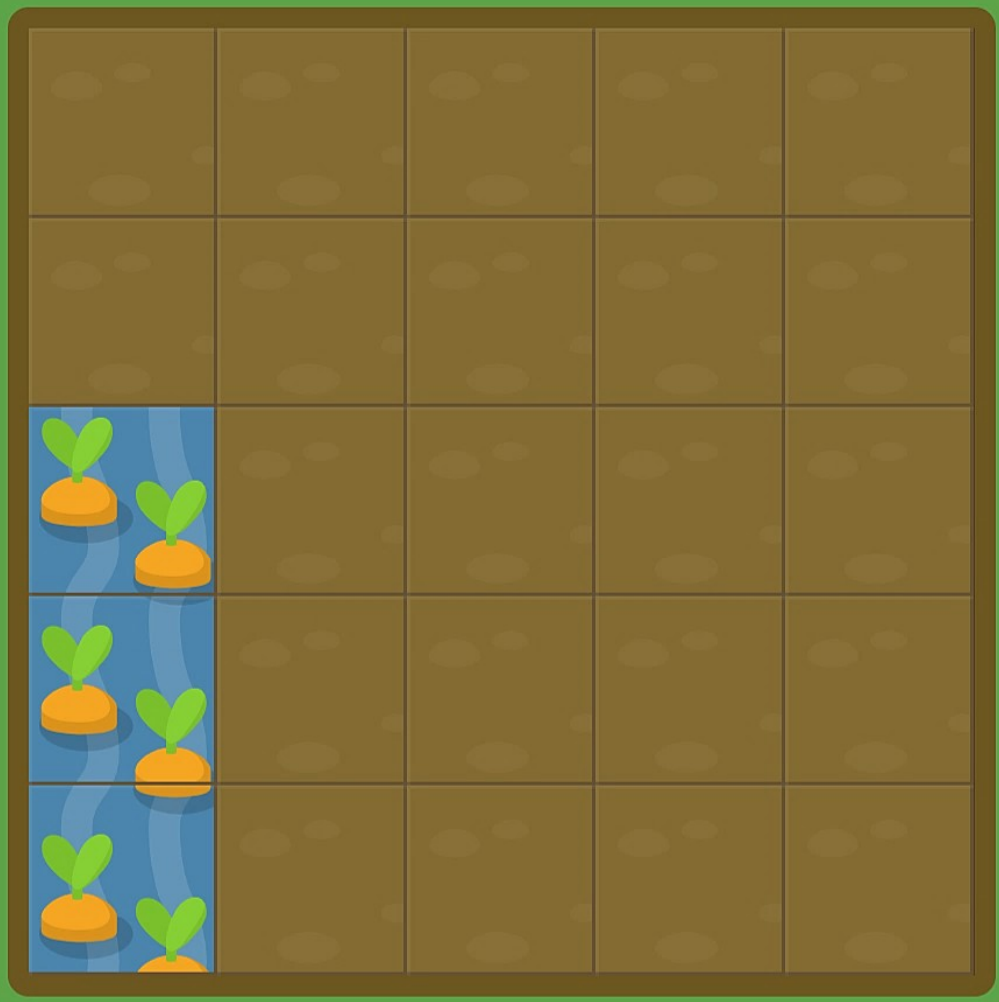
Level 13 of 28

Now give the shorthand property `grid-row` a try.

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

#water {
  grid-row:3/6;
}
```

Next



Grid Garden is created by Codepip • YouTube • Twitter • GitHub • English

Want to learn CSS flexbox? Play Flexbox Froggy.

GRID GARDEN

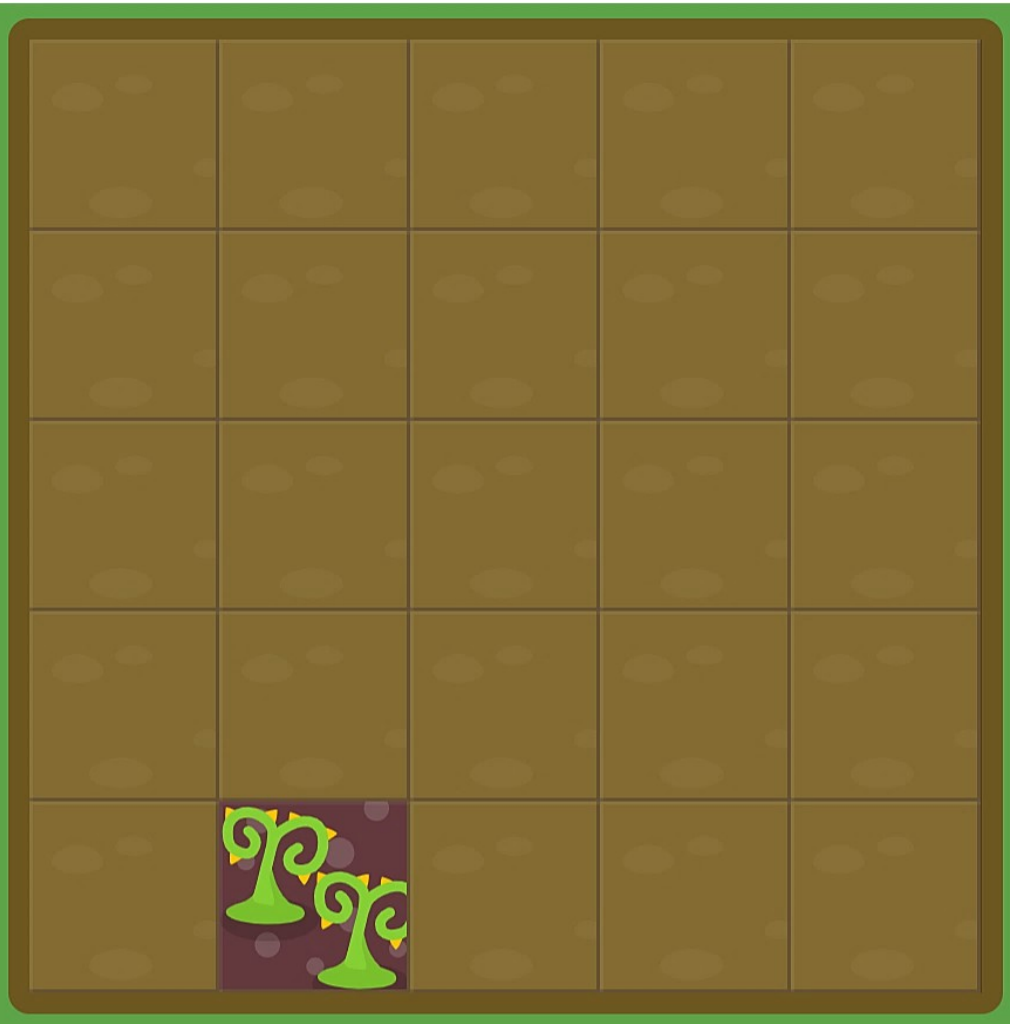
Level 14 of 28

Use `grid-column` and `grid-row` at the same time to set position in both dimensions.

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

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

Next



GRID GARDEN

Level 15 of 28

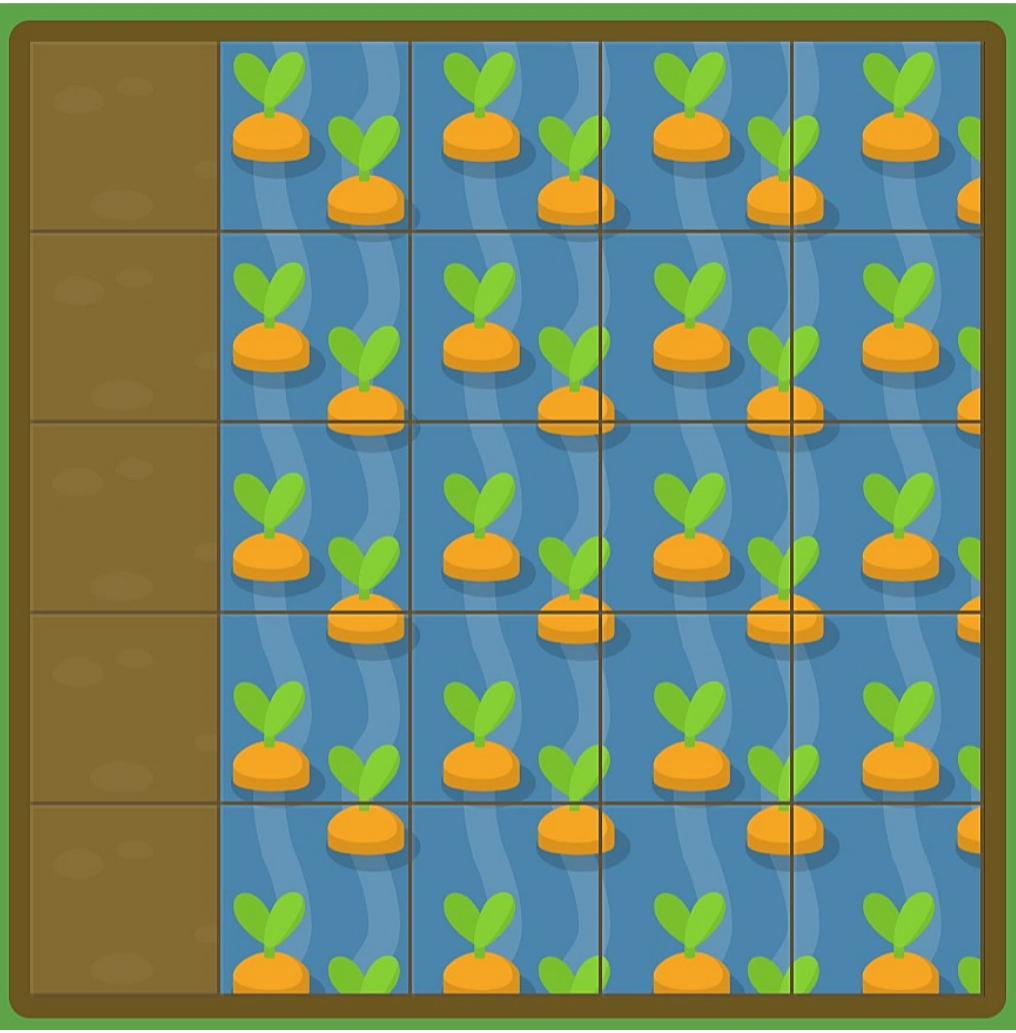
You can also use `grid-column` and `grid-row` together to span larger areas within the grid. Give it a try!

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

#water {
  grid-column:2/6;
  grid-row:1/6;
}
```

Next

Grid Garden is created by Codepip • YouTube • Twitter • GitHub • English



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

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

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

#water {
  grid-area:1/2/4/6;
}
```

Next

GRID GARDEN

Level 17 of 28

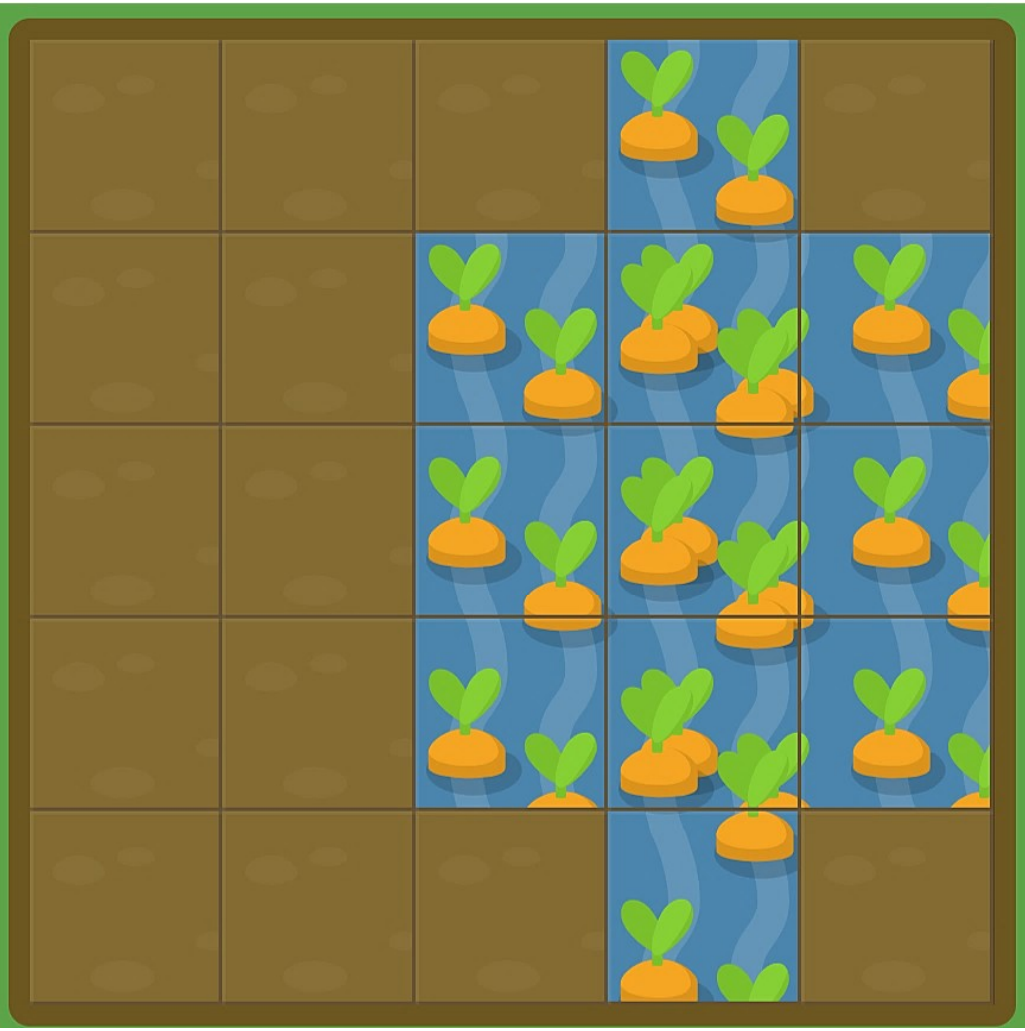
How about multiple items? You can overlap them without any trouble. Use `grid-area` to define a second area that covers all of the unwatered carrots.

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

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

#water-2 {
  grid-area: 2/3/5/6;
}
```

Next



GRID GARDEN

Level 18 of 28

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.

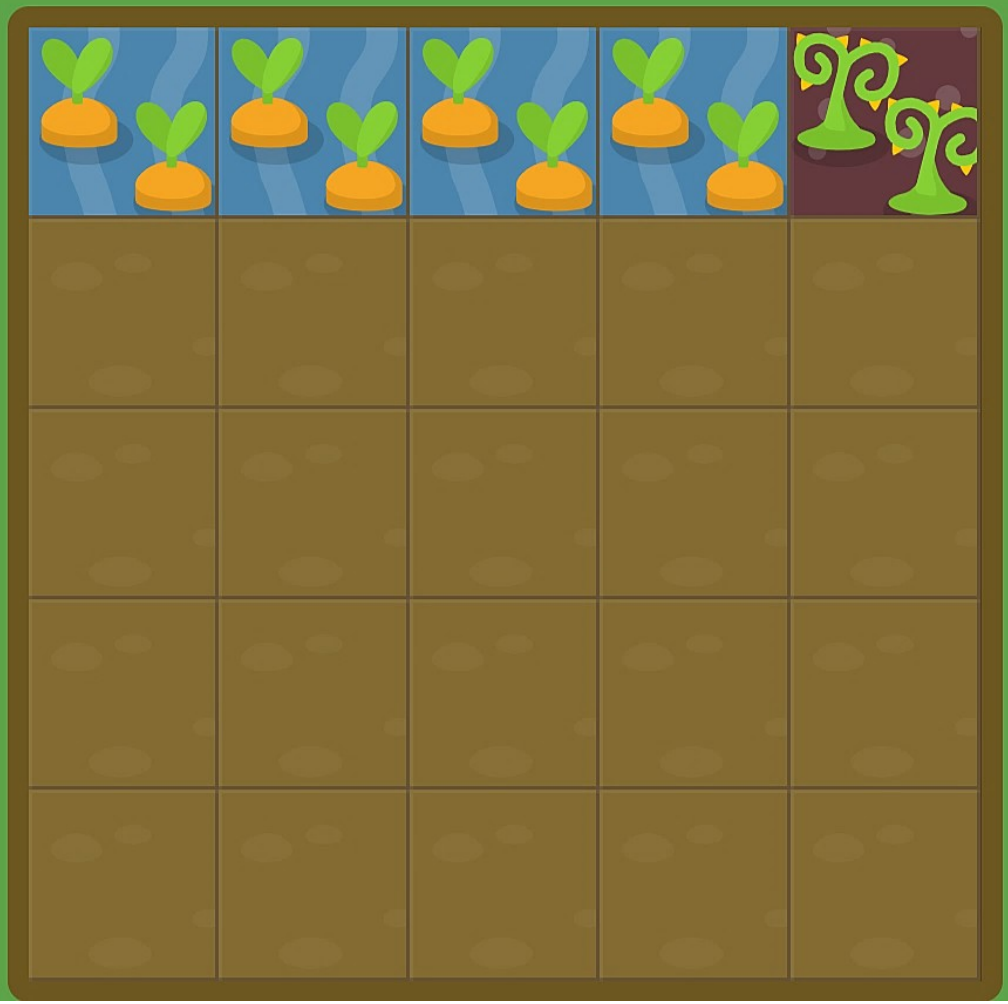
By default, all grid items have an `order` of 0, but this can be set to any positive or negative value, similar to `z-index`.

Right now, the carrots in the second column are being poisoned and the weeds in the last column are being watered. Change the `order` value of the poison to fix this right away!

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

.water {
  order: 0;
}

#poison {
```



GRID GARDEN

Level 19 of 28

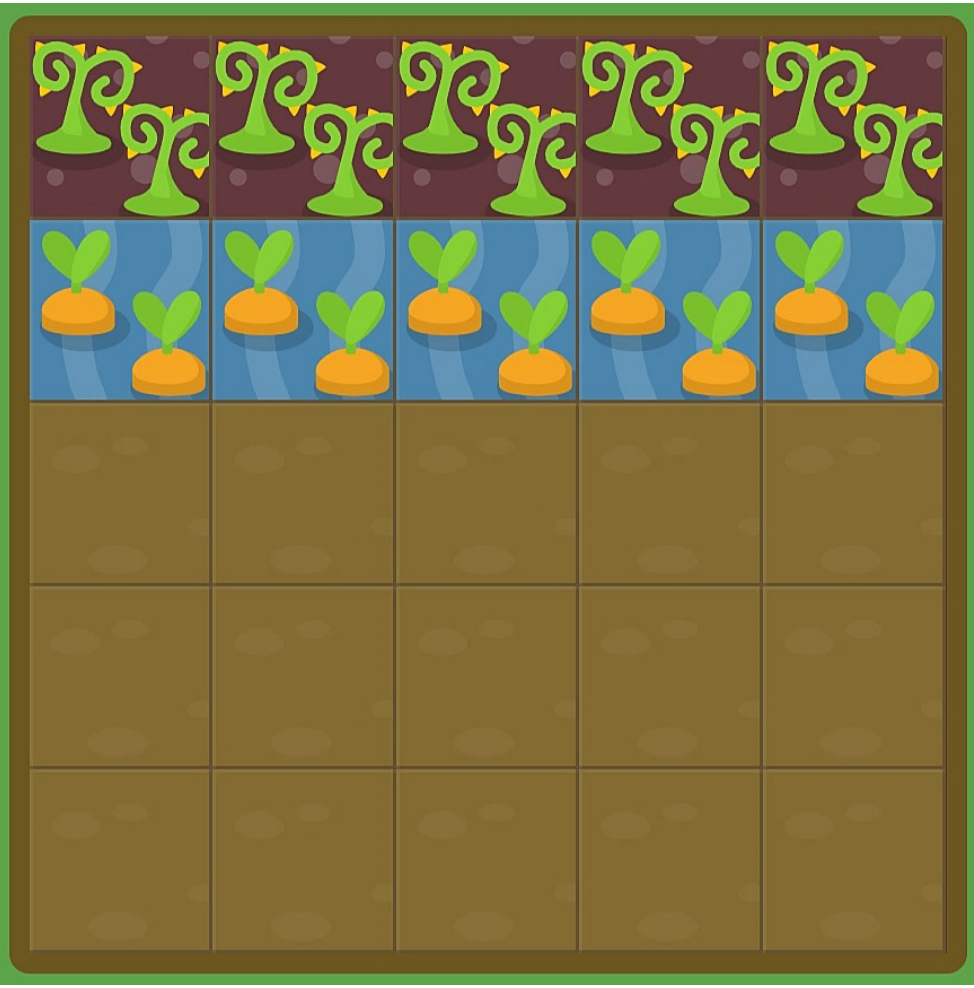
Now the water and poison are alternating, even though all of the weeds are at the start of your garden. Set the `order` of the poisons to remedy this.

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

.water {
  order: 0;
}

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

Next



GRID GARDEN

Level 20 of 28

Up to this point, you've had your garden set up as a grid with five columns, each 20% of the full width, and five rows, each 20% of the full height.

This was done with the rules `grid-template-columns: 20% 20% 20% 20% 20%;` and `grid-template-rows: 20% 20% 20% 20% 20%;` Each rule has five values which create five columns, each set to 20% of the overall width of the garden.

But you can set the grid up however you like. Give `grid-template-columns` a new value to water your carrots. You'll want to set the width of the 1st column to be 50%.

```
#garden {
  display: grid;
  grid-template-columns:50%;
  grid-template-rows: 20% 20% 20% 20% 20%;
}

#water {
  grid-column: 1;
  grid-row: 1;
}
```

GRID GARDEN

Level 21 of 28

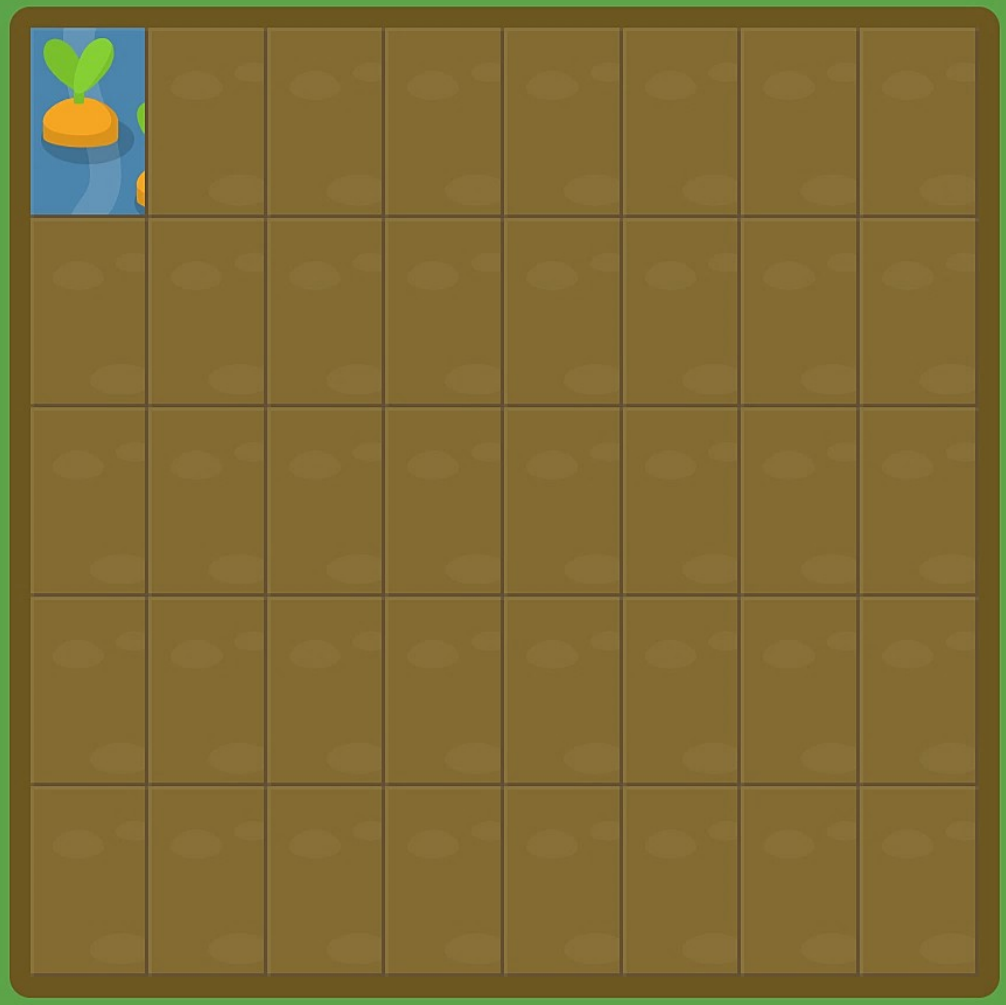
Specifying a bunch of columns with identical widths can get tedious. Luckily there's a `repeat` function to help with that.

For example, we previously defined five 20% columns with the rule `grid-template-columns: 20% 20% 20% 20% 20%;`. This can be simplified as `grid-template-columns: repeat(5, 20%);`

Using `grid-template-columns` with the `repeat` function, create eight columns each with 12.5% width. This way you won't overwater your garden.

```
#garden {
  display: grid;
  grid-template-columns: repeat(8,12.5%)
  grid-template-rows: 20% 20% 20% 20% 20%;
}

#water {
  grid-column: 1;
  grid-row: 1;
}
```



GRID GARDEN

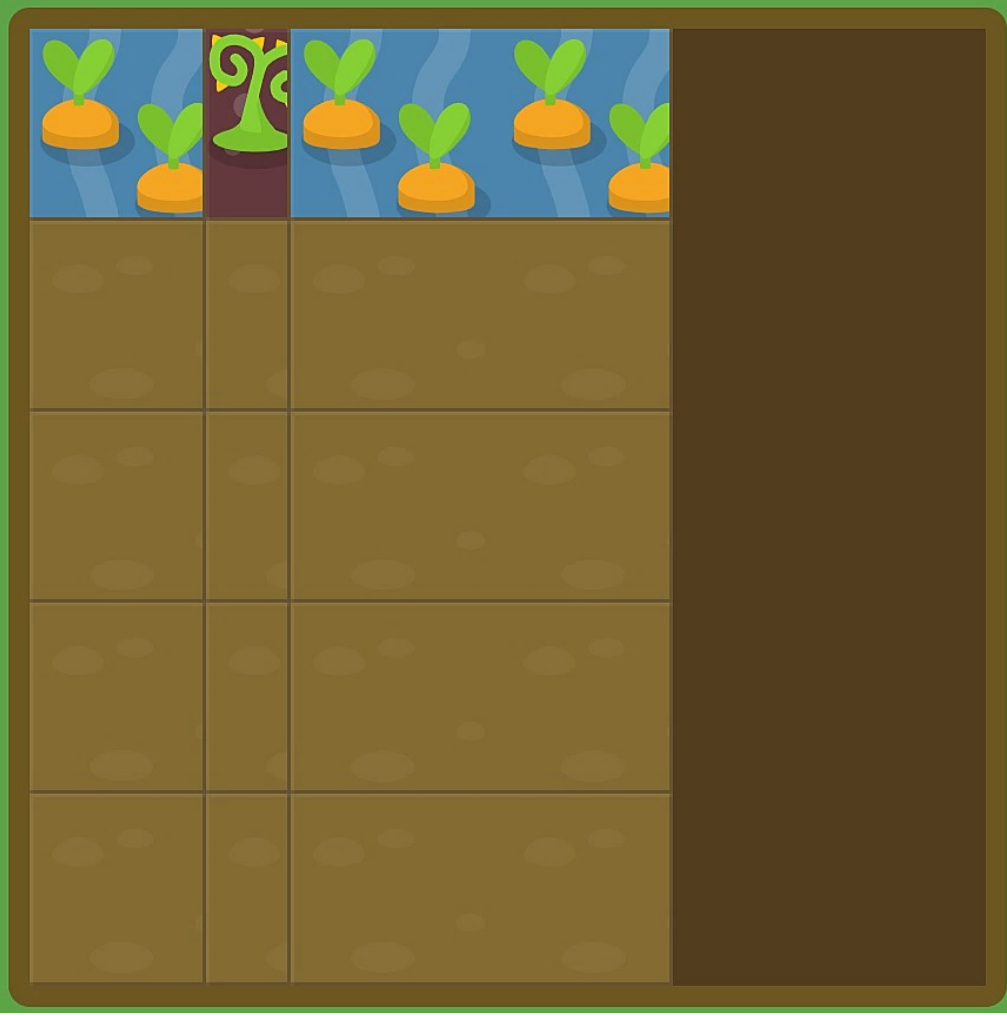
Level 22 of 28

`grid-template-columns` doesn't just accept values in percentages, but also length units like pixels and ems. You can even mix different units together.

Here, set three columns to `100px`, `3em`, and `40%` respectively.

```
#garden {
  display: grid;
  grid-template-columns:100px 3em 40%;
  grid-template-rows: 20% 20% 20% 20% 20%;
}
```

Next



GRID GARDEN

Level 23 of 28

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.

Here, weeds make up the left 1/6 of your first row and carrots the remaining 5/6. Create two columns with these widths using `fr` units.

```
#garden {
  display: grid;
  grid-template-columns: 1fr 5fr;
  grid-template-rows: 20% 20% 20% 20% 20%;
}
```

Next

GRID GARDEN

Level 24 of 28

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.

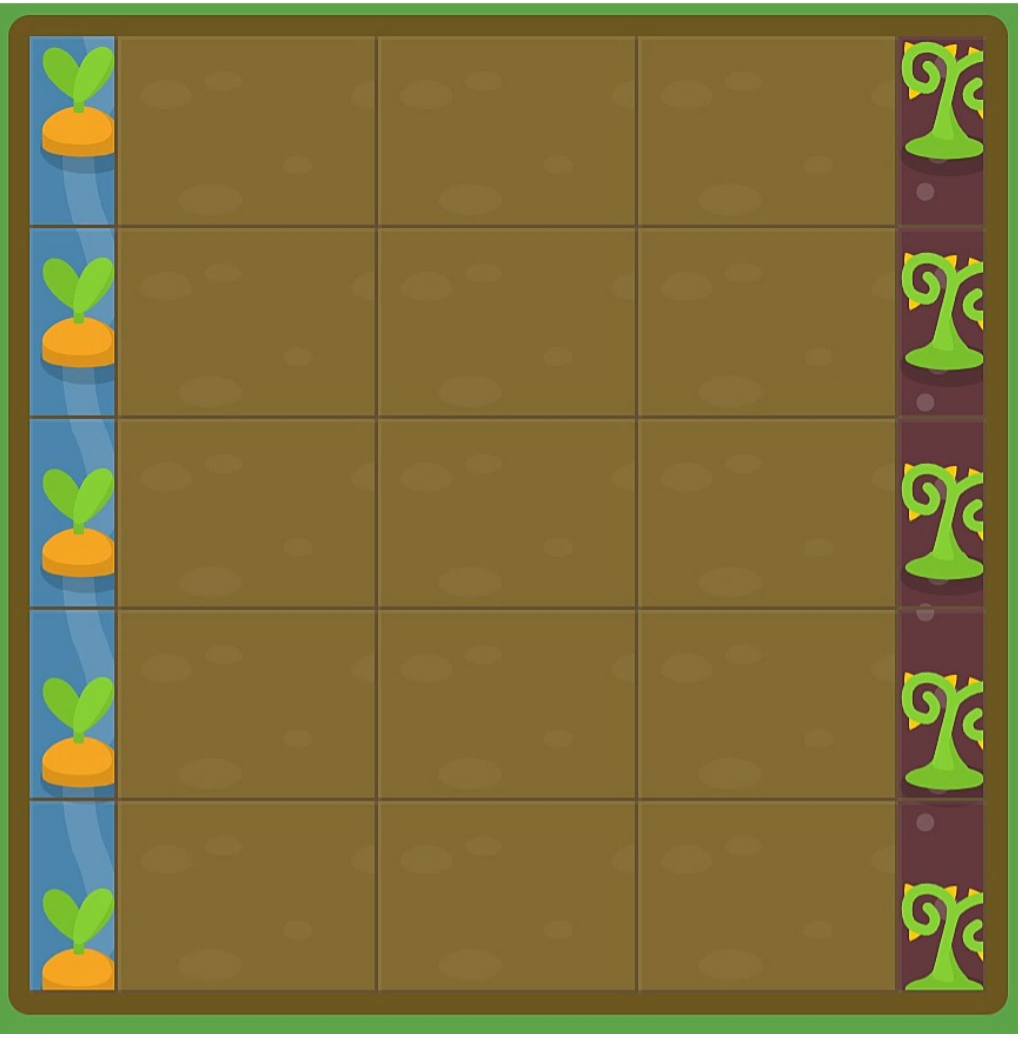
```
#garden {
  display: grid;
  grid-template-columns:50px 1fr 1fr 1fr 50px;
  grid-template-rows: 20% 20% 20% 20% 20%;
}

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

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

```

Next



GRID GARDEN

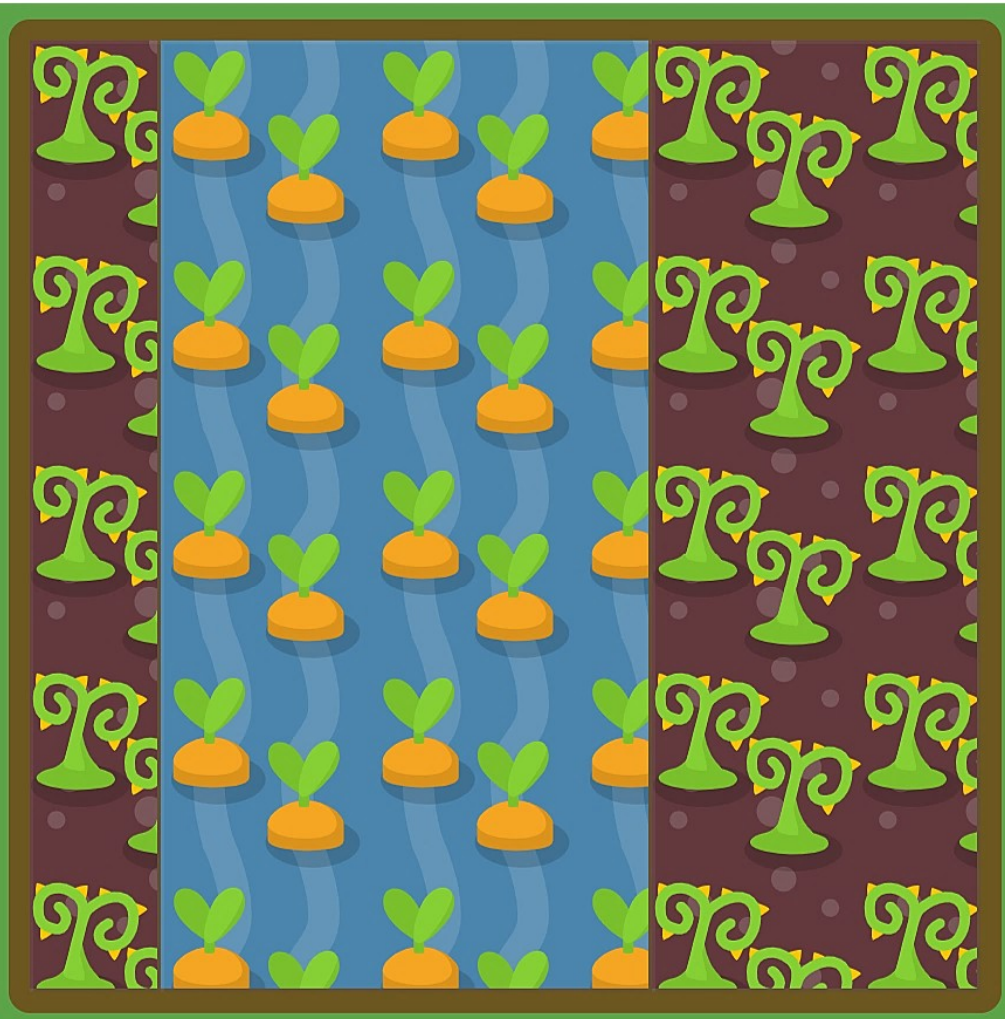
Level 25 of 28

Now there is a 75 pixel column of weeds on the left side of your garden. 3/5 of the remaining space is growing carrots, while 2/5 has been overrun with weeds.

Use `grid-template-columns` with a combination of `px` and `fr` units to make the necessary columns.

```
#garden {
  display: grid;
  grid-template-columns: 75px 3fr 2fr;
  grid-template-rows: 100%;
}
```

Next



GRID GARDEN

Level 26 of 28

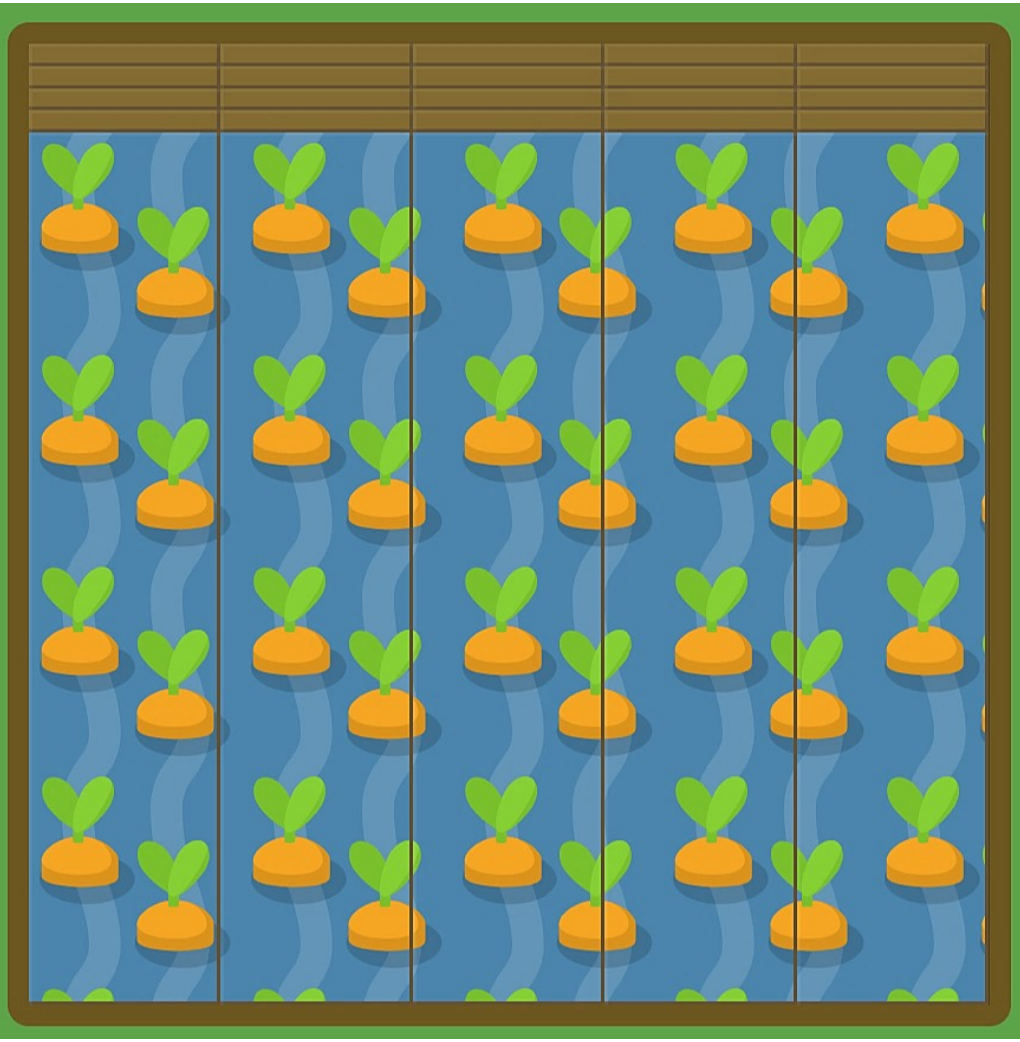
`grid-template-rows` works much the same as `grid-template-columns`.

Use `grid-template-rows` to water all but the top 50 pixels of your garden. Note that the water is set to fill only your 5th row, so you'll need to create 5 rows in total.

```
#garden {
  display: grid;
  grid-template-columns: 20% 20% 20% 20% 20%;
  grid-template-rows:12.5px 12.5px 12.5px 12.5px
}

#water {
  grid-column: 1 / 6;
  grid-row: 5 / 6;
}
```

Next



GRID GARDEN

Level 27 of 28

`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.

Try using `grid-template` to water an area that includes the top 60% and left 200 pixels of your garden.

```
#garden {
  display: grid;
  grid-template: 60% 40% /200px
}

#water {
  grid-column: 1;
  grid-row: 1;
}
```

GRID GARDEN

Level 28 of 28

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/ 1fr 4fr;
}
```

Next