# Exercise 58

Rev. 1Aug2019

## CSS GRID

Oh, the good old days. But not so good if you happened to be working with CSS positioning in particular. It was enough to drive someone mad with the seemingly-endless fidgeting of `postion` and `float` values needed to get the layout just right. Some people gave up and used HTML tables, which were never meant for the job.

And then came Flexbox. Flexbox provides a more efficent way to lay out, align, and distribute space among items in a container. Flexbox has been available to web designers for years and it's done yeoman work in taming the madness of CSS positioning. It's one drawback has been that it works best with *either* columns *or* rows, but doesn't work well with both.

Both Flexbox and Grid are built into the browser's rendering engine. There's nothing to install to make them work.

Enter CSS Grid. CSS Grid is enormously powerful, yet remarkably easy. In this exercise, we'll see experiment with it a bit in order to see both in action. Let's start with this layout:

| | |
|---|---|
| A1 | B1 |
| A2 | B2 |

The HTML for this is:

```
<main>
  <div>A1</div>
  <div>B1</div>
  <div>A2</div>
  <div>B2</div>
</main>
```

And the CSS:

```
main {
  display: grid;
  grid-template-columns: 1fr 1fr;
}

main > div {
  border: 1px solid maroon;
  font-size: 1.3em;
}
```

With CSS Grid, you start with the parent container — `<main>` in our case. The `display` property is set to `grid`. From there, you have a number of options. I opted to define two columns, and set their width to `1fr`. `fr` is a new size unit, introduced with CSS Grid. It stands for *fraction* — as in a fraction of the free space left. Using `1fr 1fr` tells the browser to take whatever space is available and divide it in two equal parts.

Let's try making it a little trickier:

| A1 | B1 | C1 |
|---|---|---|

The CSS:

```
main {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr;
}
```

Now the browser takes whatever free space there is, divides it into *four* equal parts and gives the width of two of those parts to the second column and a single part to the first and last columns.

We can match fr units with traditional ones: `grid-template-columns: 250px 2fr 1fr` is interpreted by the browser to mean: set the first column at a fixed width of 250px. Take the remaining free space and divide it into three. Give two of those parts to the second column and a single part to the last column.

| A1 | B1 | C1 |
|---|---|---|

To see a more complex use of CSS check out `index.html` and `index.css`. Complex layouts were never so easy.