



## The CSS Grid – Lab 10

**IMPORTANT!** Save all your work to a safe location such as OneDrive. Create a folder for GUI & Web Development into which you will save all your work for this module, arranged how you wish. Ideally you should create a folder each week for your lab exercises. Note that you should create a separate file for each exercise.

The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages and create layouts.

### *Exercise 1: The basics of the grid*

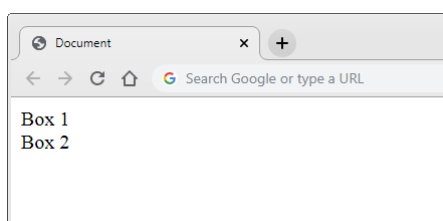
1. Create the following page that has the DIV structure as shown below:

```

1  <!doctype html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>Document</title>
6  </head>
7  <body>
8      <div class="container">
9          <div>Box 1</div>
10         <div>Box 2</div>
11     </div>
12 </body>
13 </html>

```

Note that the DIV with a class of “container” has 2 other DIVs inside it, as shown above. Save and run your page to ensure that there are no errors. Your page should look similar to as shown below:



- For the purposes of clarity – so you have a clear idea of exactly where and how all the DIVs are positioned - create a rule for DIVs with a black, solid, 1 pixel border around them:

```

3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <style>
7     div {border: 1px solid black;}
8
9   </style>
10 </head>
11 <body>

```

Refresh your page to ensure that the borders are visible.

- In order to use the CSS grid, you will need to apply a rule to the parent DIV class, which is “container”:

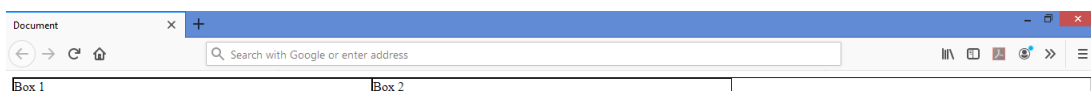
```
.container {display: grid;}
```

Although the grid has been applied, it will not look any different in the browser, as the default number of columns is 1.

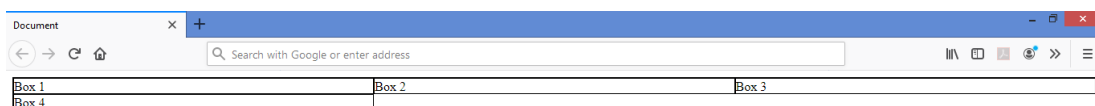
- Add a rule to specify the number and size of columns, as shown:

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

The example above is specifying three columns. Each is exactly 1 fraction of the page wide (1fr). As there are 3 fractions in this rule (1fr 1fr 1fr), then each column will be one third of a page wide. Save and run your page, it should look similar to as shown:



- Add an additional 2 DIVs inside the container DIV. Save and refresh your page, it should look similar to as shown:



Note that the 4<sup>th</sup> box has automatically appeared in a new row, underneath the first cell – “Box 1”.

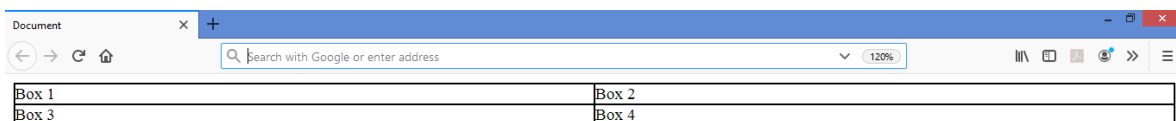
## Exercise 2: Specifying different column widths

1. Copy and save your code from exercise 1 into a new file. In this exercise you will experiment with different sizes and configurations for column widths in CSS grids.

Change the rule for the *container* class so that is specifying 2 fractions, as shown below:

```
grid-template-columns: 1fr 1fr;
```

The page will now render the grid with 2 columns, each the same size, similar to as shown:

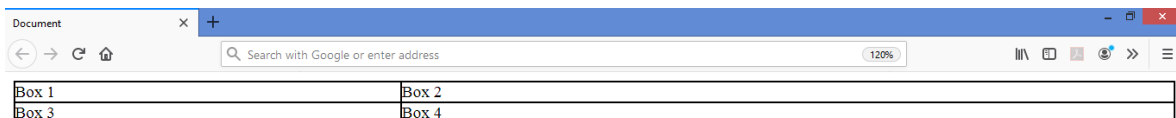


Box 1	Box 2
Box 3	Box 4

2. It is also possible to specify different fractions for each column. Change the grid rule so that it is as follows:

```
grid-template-columns: 1fr 2fr;
```

This rule will render 2 columns on the page. There is a total of 3 fractions specified in the rule (1fr + 2fr), so each fraction will be one-third the width of the page. The first column will be 1 fraction wide, and therefore one third of the page wide. The second column will be 2 fractions wide and therefore two thirds wide. Save and run your page, it should look similar to as shown:



Box 1	Box 2
Box 3	Box 4

3. Amend the rule for the grid to as follows:

```
grid-template-columns: 1fr 3fr 1fr;
```

This rule will render 3 columns on the page. There are a total of 5 fractions specified in the rule (1fr + 3fr + 1fr), so each fraction will be one-fifth the width of the page. The first column will be 1 fraction wide, and therefore one fifth of the page wide. The second column will be 3 fractions wide and therefore three fifths of the page wide. The final column will be 1 fraction and therefore one fifth of the page wide. Save and run your page, it should look similar to as shown:

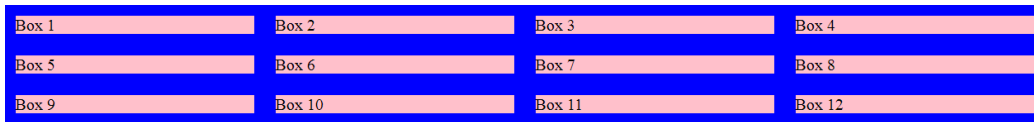
Box 1	Box 2	Box 3
Box 4		

4. Amend your code so that your page has 4 columns split over 7 fractions, and the end result is as shown:

Box 1	Box 2	Box 3	Box 4
-------	-------	-------	-------

### Exercise 3: Creating a grid

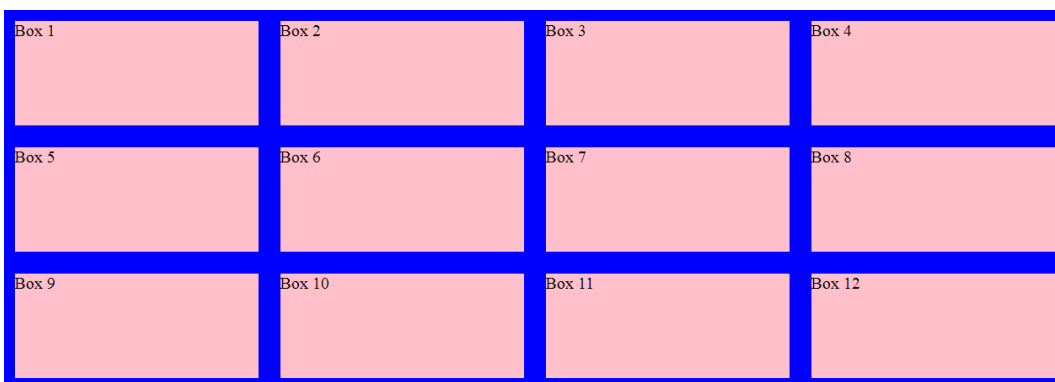
1. Create a grid that has 4 columns, each equal in size. Specify that each grid cell has a 10-pixel solid blue border with a pink background. Your page should look similar to as shown below:



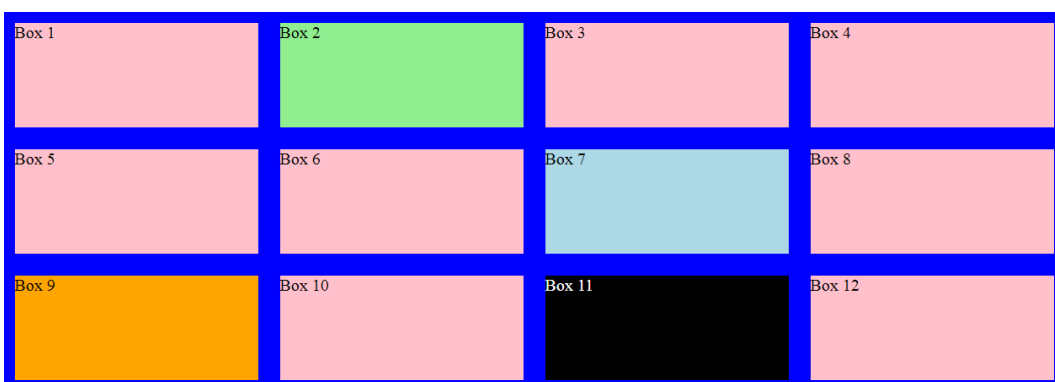
2. Change the height of the grid cells by specifying a height for each row using the `grid-template-rows` property:

```
grid-template-rows: 100px 100px 100px;
```

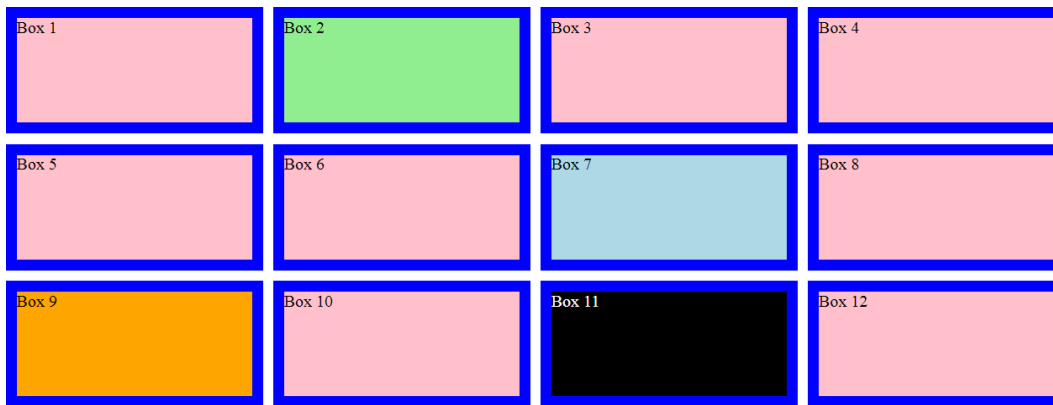
This is specifying that each row will have a height of 100 pixels. Your page should now look similar to as shown:



3. Amend your code so that boxes 2, 7, 9 and 11 are coloured as shown, using the colours lightgreen, lightblue, orange and black.



4. Add a grid gap of 10-pixels all around the grid using the `grid-gap` rule. Your page should now look similar to as shown:



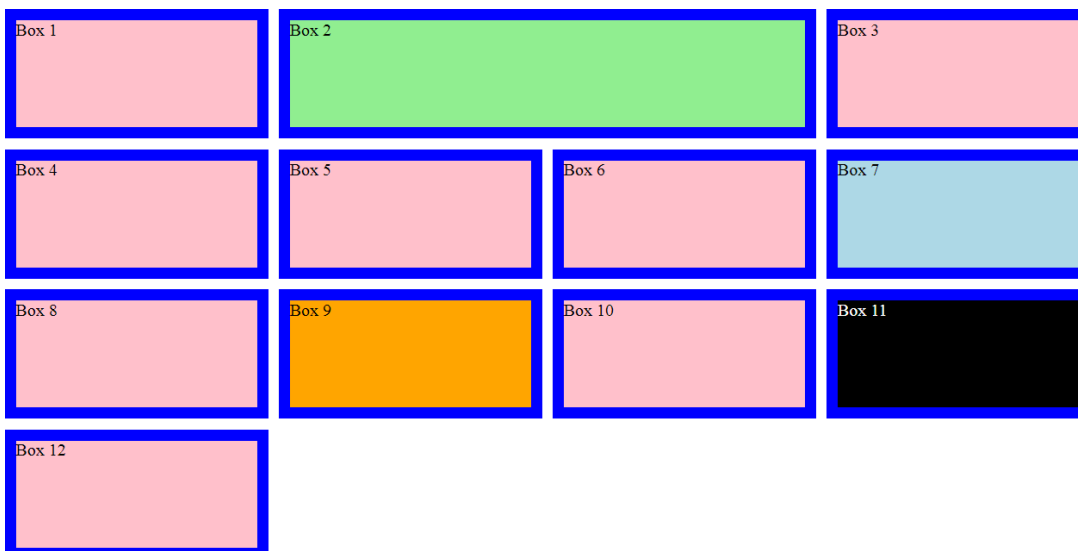
### Exercise 4: Spanning in a CSS grid

1. Copy and save your code from exercise 3 into a new file. In this exercise you will experiment with spanning columns and rows in CSS grids.

Span *Box 2* across 2 columns in the grid. To do this, you will need to apply the grid-column rule and specify how many columns the cell will span:

```
grid-column: span 2;
```

Save and refresh your page, it should look similar to as shown below:

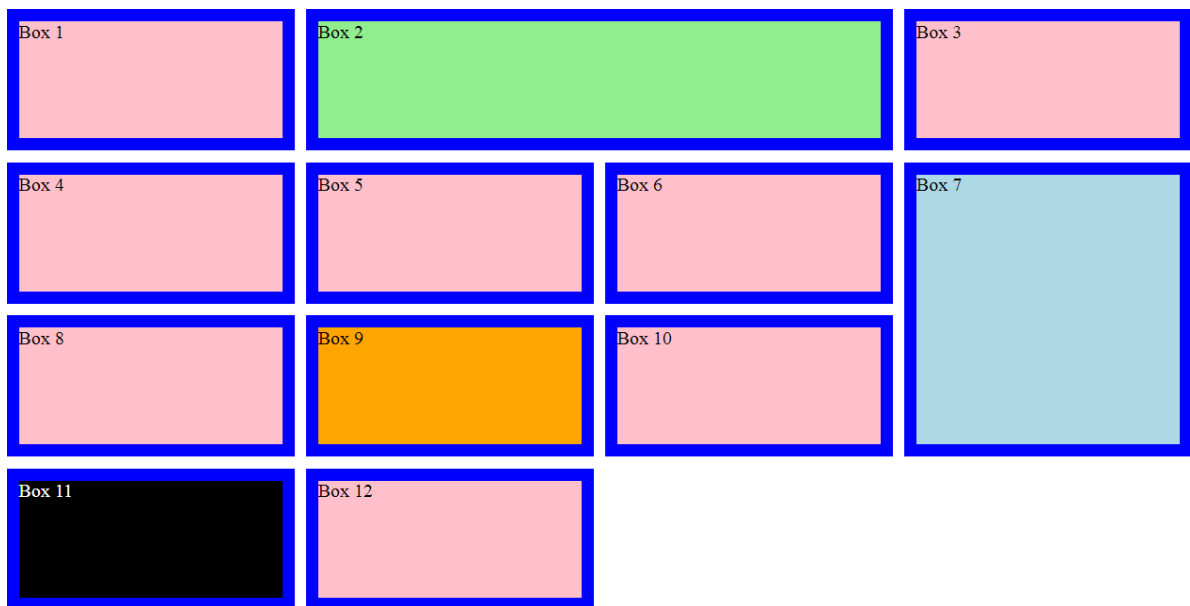


*The green box (Box 2) should now be spanning 2 columns. Note that you need to amend the rule for grid-template-rows, as a fourth row is now present as a result of using span.*

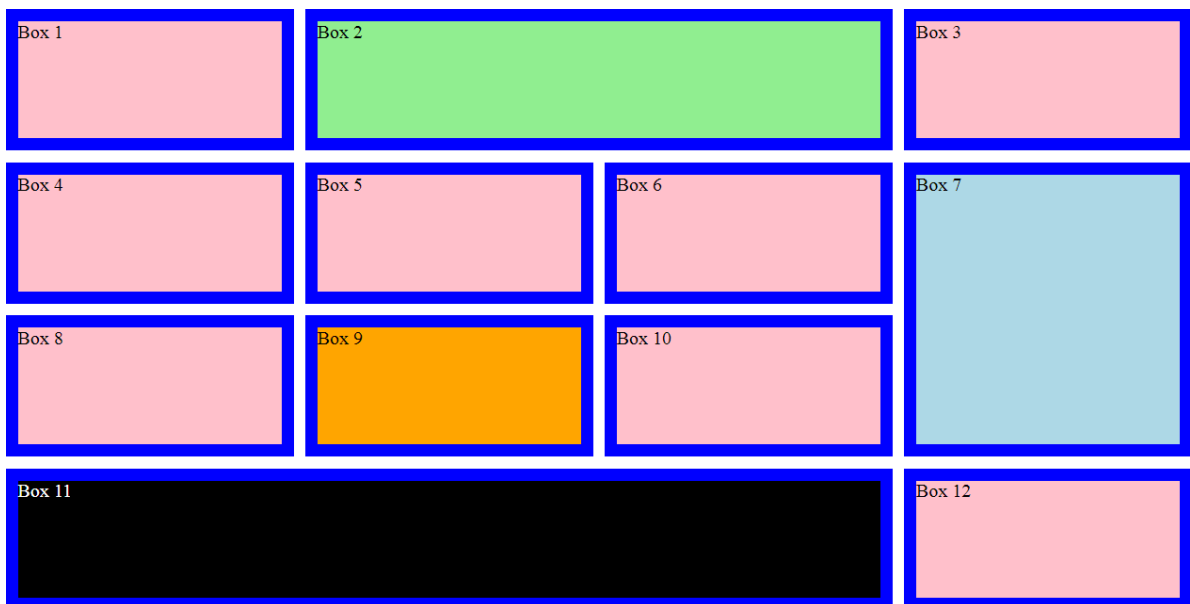
2. Span *Box 7* across 2 rows in the grid. To do this, you will need to apply the grid-row rule and specify how many columns the cell will span:

```
grid-row: span 2;
```

3. Save and refresh your page, it should look similar to as shown below:



4. Amend your code so that your page looks similar to as shown here:



## Exercise 5: Specifying various row heights in a CSS grid

1. Create a new grid similar to as shown below.  
All cells should have a 5px solid black border, with an orange background. There should be a grid gap of 5-pixels. You will use a grid rule to specify a different row height for each row of the grid.

Similar to the previous exercise, there should be 12 DIVs inside the container DIV. The grid should have 3 columns, each of an equal size, as shown here:

Box 1	Box 2	Box 3
Box 4	Box 5	Box 6
Box 7	Box 8	Box 9
Box 10	Box 11	Box 12

Note that there are 4 rows in this grid. Row 1 starts with *Box 1*, row 2 starts with *Box 4*, row 3 starts with *Box 7* and the fourth row starts with *Box 10*.

Using the `grid-template-rows` rule, you will specify a specific height for each row. Row 1 will have an automatic height, row 2 will have a height of 100 pixels, row 3 will have a height of 250 pixels and row 4 will have a automatic height.

```
grid-template-rows: auto 100px 250px auto;
```

Save and refresh your page, it should look similar to as shown below:

Box 1	Box 2	Box 3
Box 4	Box 5	Box 6
Box 7	Box 8	Box 9
Box 10	Box 11	Box 12

*Note that rows with an automatic height will increase as required. In other words, if a row with an automatic height has a DIV with a large amount of text, it will increase in height to allow all the text to fit.*

2. Change the background colour of Box 6 to light blue, and span it by 4 rows. You will not need to specify a height as it will automatically extend to the height of each row. Your finished page should look similar to as shown:

Box 1	Box 2	Box 3
Box 4	Box 5	Box 6
Box 7	Box 8	
Box 9	Box 10	
Box 11	Box 12	

## Exercise 6: Creating a page layout using CSS Grid

In this exercise, we will create a web page with multiple panels containing text and use positioning to create our finished page.

In this exercise, we will create a web page with multiple panels containing text and use positioning to create our finished page.

1. Create a basic grid layouts as shown below.

Box 1	Box 2	Box 3
Box 4		

The above is using a total of 4 fractions and 3 columns for the layout.

2. Span the first box so it is across the entire page:

Box 1		
Box 2	Box 3	Box 4

3. Amend the width of the container so that it takes up 40% of the page.
4. Amend the contents of the Box 1 DIV so that the page is as following:

<h2 style="text-align: center;">The Story of CSS Grid, from it's Creators</h2>		
Box 2	Box 3	Box 4

5. Add the text below to Box 2:

While the modern concept of a grid layout has been with us since the Industrial Revolution, grids have been a design tool for centuries. As such, it shouldn't come as a shock that grid-based layouts have been a goal of CSS since the beginning. According to Dr. Bert Bos, who co-created CSS with Håkon Wium Lie, grid-based layouts have actually been on his mind for quite some time. "CSS started as something very simple," Bos recalled. "It was just a way to create a view of a document on a very simple small screen or desktop. (Source: <https://www.w3.org/2019/04/24-css-grid-101/>)"

So that the page is as following:

<h2 style="text-align: center;">The Story of CSS Grid, from it's Creators</h2>		
<p>While the modern concept of a "grid layout" has been with us since the Industrial Revolution, grids have been a design tool for centuries. As such, it shouldn't come as a shock that grid-based layouts have been a goal of CSS since the beginning. According to Dr. Bert Bos, who co-created CSS with Håkon Wium Lie, grid-based layouts have actually been on his mind for quite some time. "CSS started as something very simple," Bos recalled. "It was just a way to create a view of a document on a very simple small screen or desktop. (Source: <a href="https://www.w3.org/2019/04/24-css-grid-101/">https://www.w3.org/2019/04/24-css-grid-101/</a>)"</p>	Box 3	Box 4



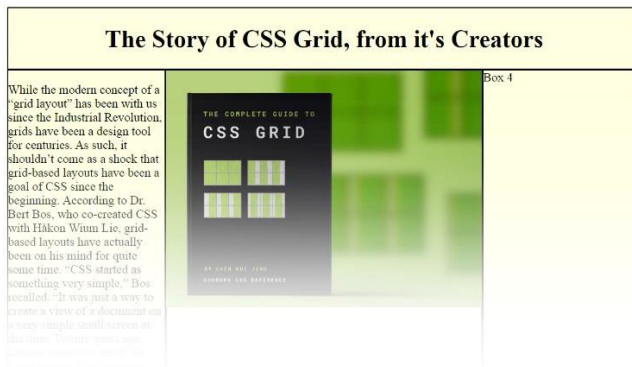
6. Using the grid image, available for download – see lab10\_images folder included with this lab on Moodle, amend the contents of Box 3 so that it contains the image tag for the image:

```
<div>
  
</div>
```

7. Save and run your page. Note that the image is too big for the grid and needs to be resized. To resize it, you can create a CSS rule to resize all images or if you prefer, give the image tag an ID and create a rule to target that ID.

```
img {width: 100%;}
```

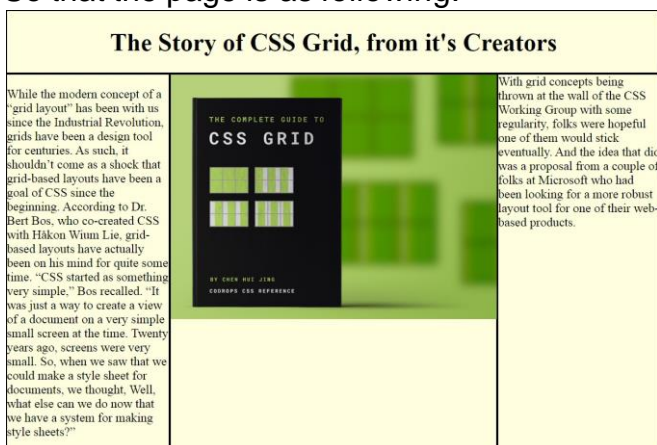
After applying this rule, your page should look similar to as shown here:



8. Add the text below to Box 4:

With grid concepts being thrown at the wall of the CSS Working Group with some regularity, folks were hopeful one of them would stick eventually. And the idea that did was a proposal from a couple of folks at Microsoft who had been looking for a more robust layout tool for one of their web-based products.

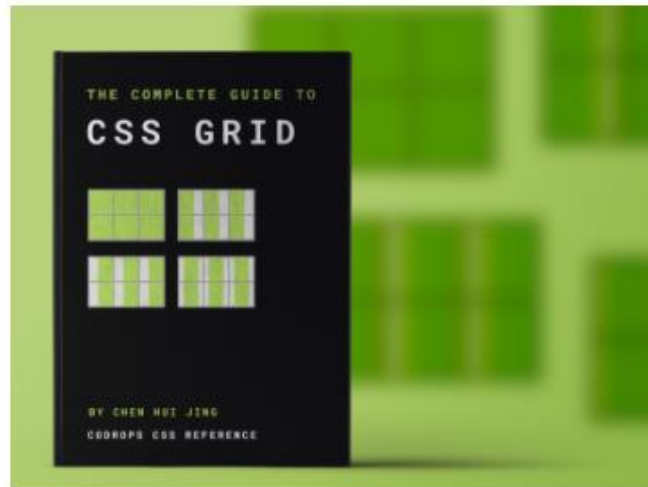
So that the page is as following:



9. Remove the background colour and the border. Add a grid gap so that it appears similar to as shown:

# The Story of CSS Grid, from it's Creators

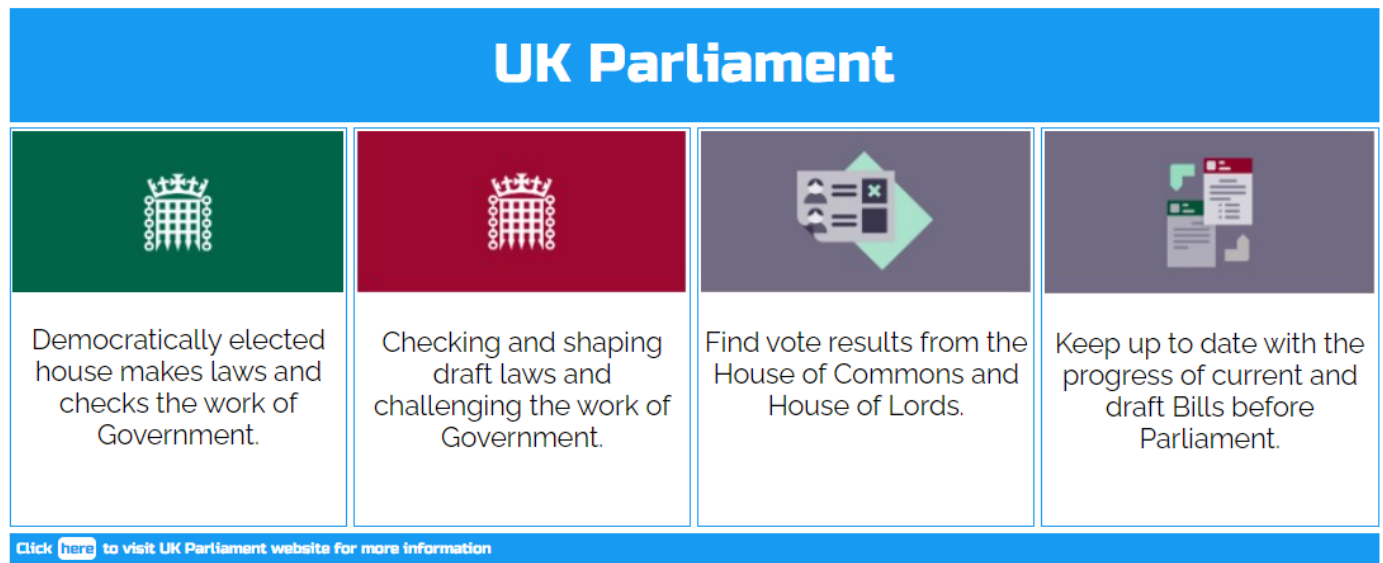
While the modern concept of a grid layout has been with us since the Industrial Revolution, grids have been a design tool for centuries. As such, it shouldn't come as a shock that grid-based layouts have been a goal of CSS since the beginning. According to Dr. Bert Bos, who co-created CSS with Håkon Wium Lie, grid-based layouts have actually been on his mind for quite some time. "CSS started as something very simple," Bos recalled. "It was just a way to create a view of a document on a very simple small screen at the time. Twenty years ago, screens were very small. So, when we saw that we could make a style sheet for documents, we thought, Well, what else can we do now that we have a system for making style sheets?"



With grid concepts being thrown at the wall of the CSS Working Group with some regularity, folks were hopeful one of them would stick eventually. And the idea that did was a proposal from a couple of folks at Microsoft who had been looking for a more robust layout tool for one of their web-based products.

## Exercise 7: Creating a page layout using CSS Grid

1. Recreate the web page shown below:

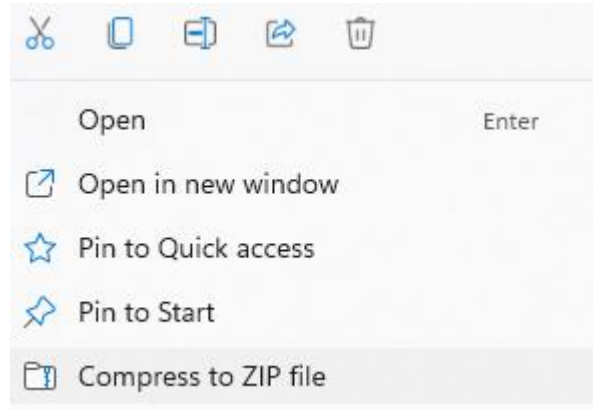


Note the following:

- Page uses a grid layout which splits up the page into 4 fractional parts – columns, and 3 rows.
- Background color used for top row and bottom row is `rgb(23, 154, 242)`.
- The `grid-gap` property is used.
- Font used for top and bottom rows is Russo One – available at [google fonts](#).
- Images for this page are available for download on Moodle – see lab10\_images folder included with this lab on Moodle.
- Text below images should be placed in paragraphs and use the Raleway font - available at [google fonts](#).
- Consider using padding, font-size, text-align and other css properties to style the page similar to what is shown in the screenshot.
- In the bottom row, there is a link to the UK parliament website (<https://www.parliament.uk/>) – see the here button - Clicking on this should open the UK parliament website in a separate tab. The link should be styled as a button, contain no underline, and it should implement the hover pseudo class to change to a background color and text color of your choosing.

## Upload your work to Moodle

1. Navigate to the location of the folder where you saved all your work for today's lab.
2. Right-click on the folder and select "Compress to ZIP file". This will create a compressed version of any files you have worked on for the lab.



3. There should be a new compressed file created. This is the file that you will need to upload to Moodle.
4. In Moodle, navigate to the "Submissions" section, and click the **upload** link for current lab.
5. Click "add submission" and add the ZIP file you created here. Make sure you complete the submission process. Your lab work has been submitted.