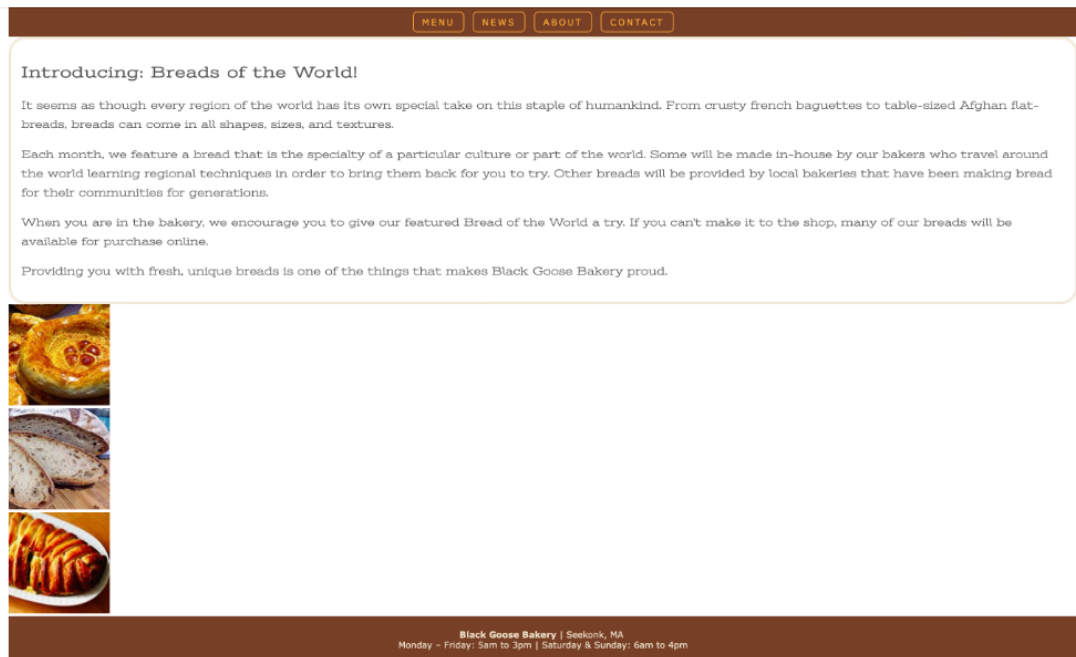


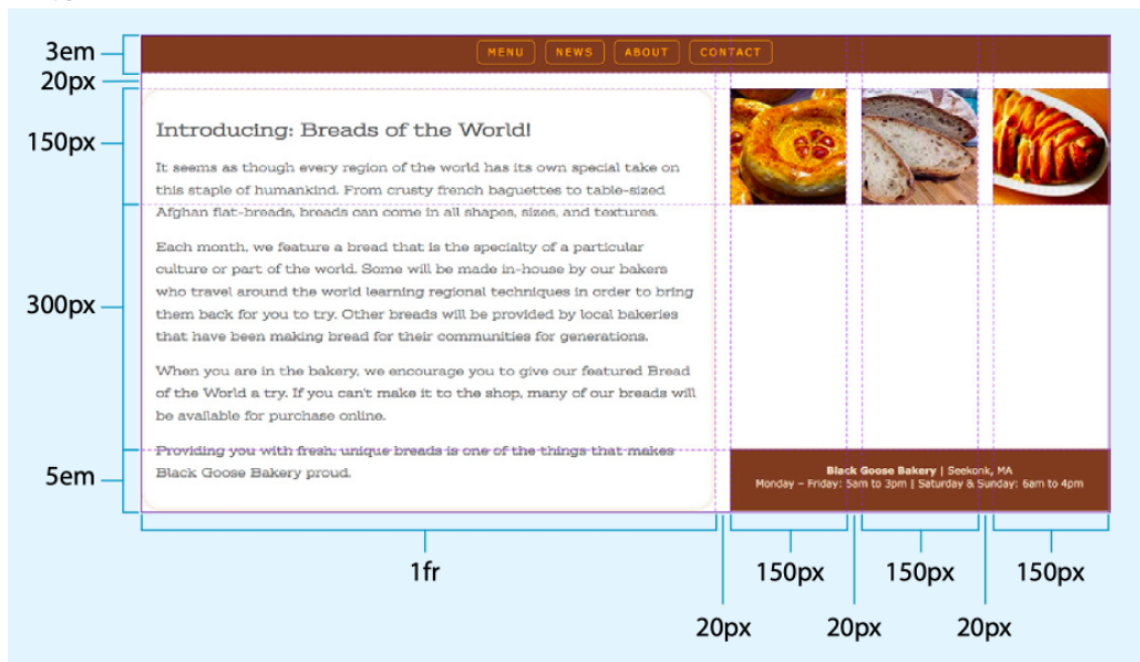
## INT102 Lab 12 Grid Layout

1. Unzip the lab12.zip and open the folder in VS code.

Before



After



2. Open the [grid.html](#). Start by turning the containing element, the #layout div, into a grid container by setting its display mode to “grid”:

```
#layout {  
  . . .  
  display: grid;  
}
```

3. The “After” picture show the row and column tracks required to accommodate the content in the designed layout. Start by defining the rows as specified in the sketch, using the [grid-template-rows](#) property. There should be six values, representing each the six rows.

```
#layout {  
  . . .  
  display: grid;  
  grid-template-rows: 3em 20px 150px 300px 5em ;  
}
```

4. Do the same for the seven columns. Because we want the text column to grow and shrink with the available space, we’ve specified its width in [fractional unit \(1fr\)](#). The remaining columns create 150px-wide cells for three images and 20px of space before them.

```
#layout {  
  . . .  
  display: grid;  
  grid-template-rows: 3em 20px 150px 300px 5em ;  
  grid-template-columns: 1fr 20px 150px 20px 150px 20px 150px ;  
}
```

However, because the last six columns are a repeating pattern, it would be easier to use the [repeat\(\)](#) function to repeat the spaces and figure columns three times:

```
#layout {  
  . . .  
  display: grid;  
  grid-template-rows: 3em 20px 150px 300px 5em ;  
  grid-template-columns: 1fr repeat(3, 20px 150px) ;  
}
```

- Finally, let assign names to the grid lines that border the grid area where the main content element should appear. The names give us some intuitive options for placing that item later. The main area starts at the third row track, so assign the name “**main-start**” to the grid line between the second and third row track measurements:

```
grid-template-rows: 3em 20px [main-start] 150px 300px 5em
```

The main area extends into the last row track, so assign the name “**main-end**” to the last grid line in the grid (after the last row track):

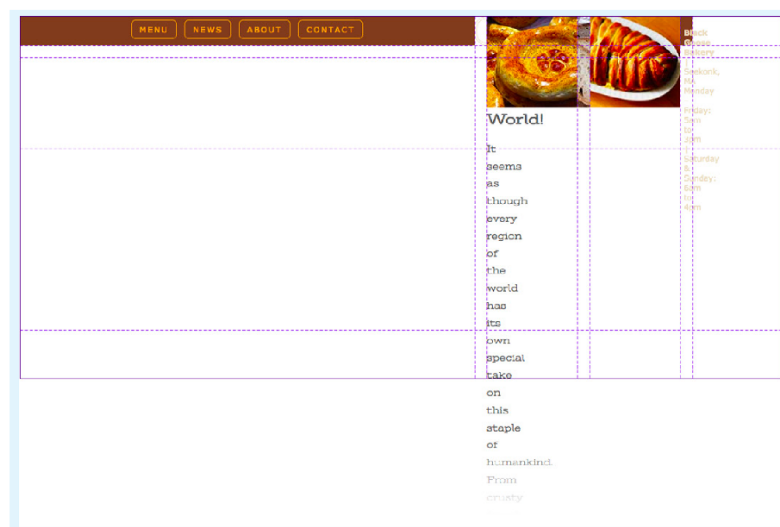
```
grid-template-rows: 3em 20px [main-start] 150px 300px 5em [main-end] ;
```

```
#layout {
  margin-left: 5%;
  margin-right: 5%;
  display: grid;
  grid-template-rows: 3em 20px [main-start] 150px 300px 5em [main-end];
  grid-template-columns: [main-start] 1fr [main-end] repeat(3, 20px 150px);
}
```

- Now do the same for the grid lines that mark the boundaries of the column track where the main content goes:

```
#layout {
  . . .
  display: grid;
  grid-template-rows: 3em 20px [main-start] 150px 300px 5em [main-end];
  grid-template-columns: [main-start] 1fr [main-end] repeat(3, 20px 150px);
}
```

- Save the document and look at it in a browser. Because we haven’t specified where the grid items go, they flowed into the cells sequentially, making the mess as shown in the picture. However, the grid overlay reveals that the structure of the grid looks solid.



### Placing Items on a grid

- Let's start by placing the nav element into the first row of the grid, using the four grid line properties:

```
nav {  
  grid-row-start: 1;  
  grid-row-end: 2;  
  grid-column-start: 1;  
  grid-column-end: 8; /* you could also use -1 */  
}
```

- Now place the figures in their positions on the grid. Start by putting the third figure (**#figC**) in its place in the far-right column by using the shorthand **grid-row** and **grid-column** properties. It goes between the 3<sup>rd</sup> and 4<sup>th</sup> row grid lines and extends from the 7<sup>th</sup> to 8<sup>th</sup> column lines.

For the columns, instead of 7 and 8, use the negative value for the last line and span it one space to the left to get the starting point:

```
#figC {  
  grid-row: 3 / 4;  
  grid-column: span 1 / -1;  
}
```

Now position the **#figA** and **#figB** elements by using the grid area property with line values. Remember that the values go in the order top, left, bottom, right (counterclockwise around the area).

```
#figA {  
  grid-area: 3 / 3 / 4 / 4;  
}  
#figB {  
  grid-area: 3 / 5 / 4 / 6;  
}
```

10. We gave the grid lines around the main area names, so let's use them to place the main grid item:

```
main {  
  grid-row: main-start / main-end;  
  grid-column: main-start / main-end;  
}
```

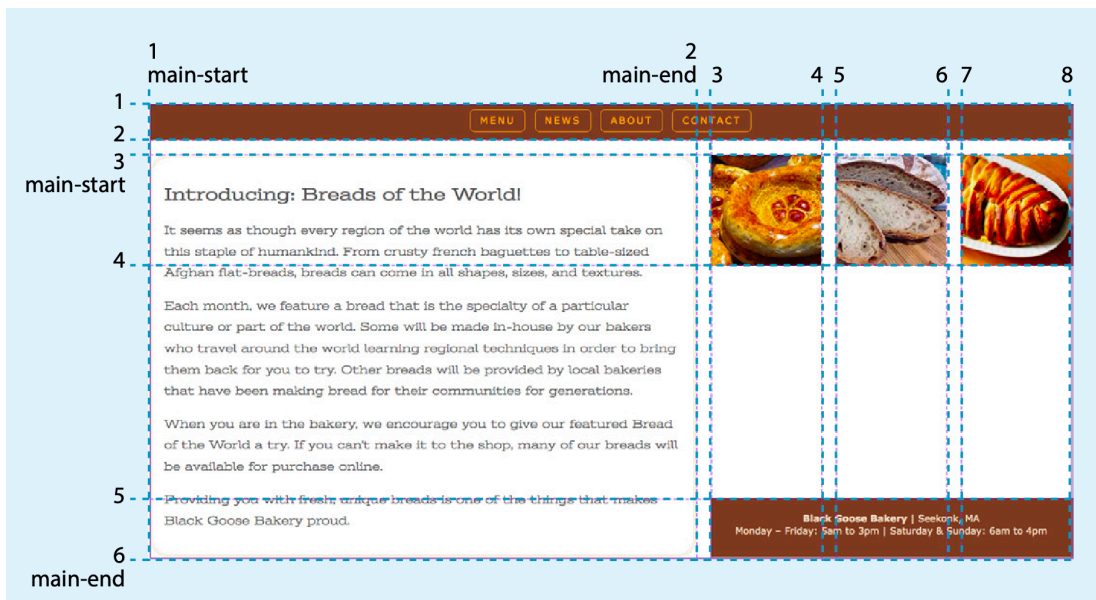
Do you remember that when you name lines around an area `*-start` and `*-end`, it creates an implicitly name area `*`? Because we named the lines according to this syntax, we could also place the main element with `grid-area` like this:

```
main {  
  /* grid-row: main-start / main-end;  
  grid-column: main-start / main-end; */  
  grid-area: main;  
}
```

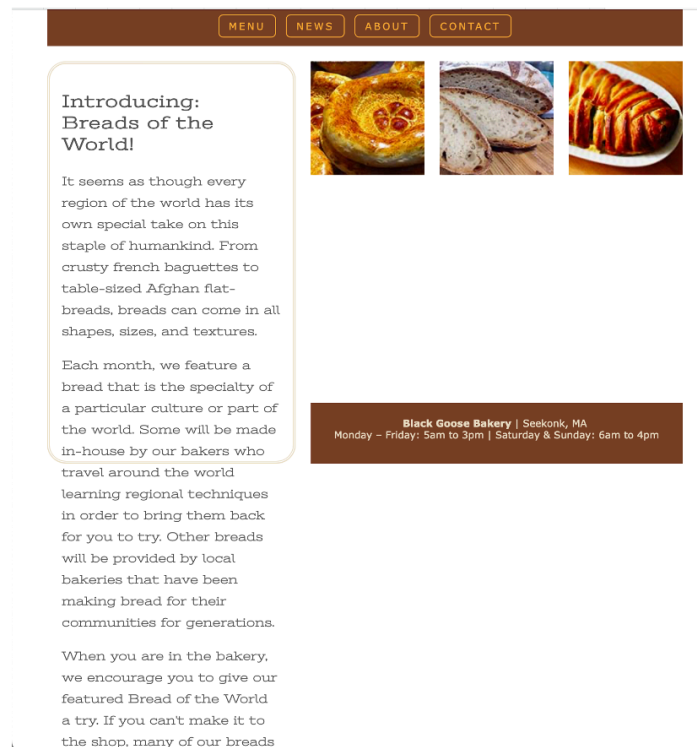
11. Finally, we can put the footer into its place. It starts at the last row grid line and spans back one track. For columns, it starts at the third line and goes to the last.

```
footer {  
  grid-row: 5 / 6;  
  grid-column: 3 / -1;  
}
```

12. Save the document and look at it in a browser.



13. When browser is wide, the layout works fine, but when it is made narrower, the text in the main element overflows its cell. That's because the 300-pixel height we gave that row is not sufficient to hold the text when it breaks onto additional lines or is resized larger.



We can fix that by changing the measurement of the fifth row track to **auto**. In that way, the height of that row will always be at least big enough to hold the content. The **min-content** value would work as well, but **auto** is always the first value to try:

```
#layout {  
  margin-left: 5%;  
  margin-right: 5%;  
  display: grid;  
  grid-template-rows: 3em 20px [main-start] 150px auto 5em [main-end];  
  grid-template-columns: [main-start] 1fr [main-end] repeat(3, 20px 150px);  
}
```

If you reload the page in the browser, the text is always contained in its grid area, regardless of the width of the window.

If you reload the page in the browser, the text is always contained in its grid area, regardless of the width of the window.



14. When you finish, rename the `bakery-styles.css` to `xxxxxxxxxxx.css` (`xxxxxxxxxxx` is your student id) and upload the `xxxxxxxxxxx.css` to LEB2.