Grid Layout

CS106E, Young

Grid Basics

Using Grid Layout we divide part or all of our webpage into a grid and then place individual elements onto the grid. We begin by selecting an HTML element to divide – if we want to divide the entire webpage, select the <body> tag. We then specify the size of rows and columns in our grid. Here's what our CSS might look like:

```
body {
  display: grid;
  grid-template-columns: 100px 100px;
  grid-template-rows: 100px 100px;
}
```

I've specified that body will be broken into a 2x2 grid with each row and each column 100 pixels.

Next, we place individual elements on the webpage and place them into the grid. Suppose I have a <div> in my HTML file defined like this:

```
<div id="a">Alpha</a></div>
```

I can place it into my grid by writing a style rule selecting my div with an id selector:

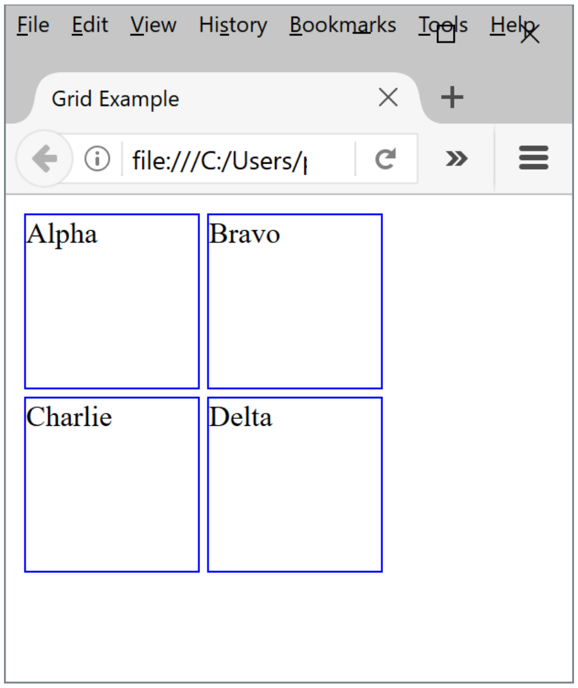
```
#a {
  grid-column-start: 1;
  grid-row-start: 1;
}
```

I specify the starting column and the starting row on my grid. If I don't provide an ending column or ending row, the assumption is that the element only takes a single column and row (we'll see column and row-spanning examples momentarily).

Note that CSS grids use a starting index of 1. This means my div placed at (1, 1) will be the top-left corner square in the grid.

Here's a complete HTML file using Grid Layout, the actual webpage is shown on the right. You can find this available in the accompanying zip file as "01-basic-grid.html". I won't be listing the other examples in their entirety, so you should get the examples from the zip file to follow along.

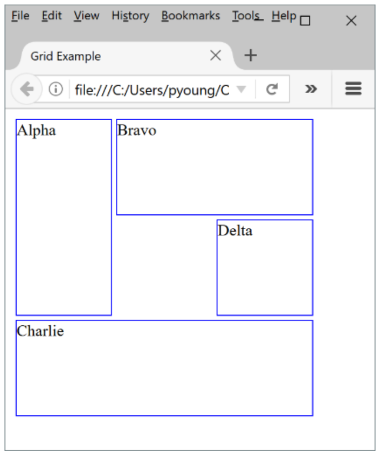
```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<title>Grid Example</title>
<style>
body {
   display: grid;
   grid-template-columns: 100px 100px;
   grid-template-rows: 100px 100px;
}
div {
   border: 1px solid blue;
   margin: 2px;
}
#a {
   grid-column-start: 1;
   grid-row-start: 1;
}
#b {
   grid-column-start: 2;
   grid-row-start: 1;
}
#c {
   grid-column-start: 1;
   grid-row-start: 2;
}
#d {
   grid-column-start: 2;
   grid-row-start: 2;
}
</style>
</head>
<body>
<div id="a">Alpha</a></div>
<div id="b">Bravo</a></div>
<div id="c">Charlie</a></div>
<div id="d">Delta</a></div>
</body>
</html>
```



Spanning Columns and Rows

We aren't limited to creating simple grids, as we can create much more complex layouts by allowing some elements to span multiple columns or rows. Here's an example of a website that includes spanning both columns and rows.

We begin by dividing the <body> into a grid using the following CSS rule:



```
body {
  display: grid;
  grid-template-columns: 100px 100px 100px;
  grid-template-rows: 100px 100px 100px;
}
```

There are several different ways to define how an element spans rows or columns.

Using Start and End Properties

Just as we specified the starting column of an element, we can also specify the ending element. The version of the webpage using this technique is stored as the file "02-grid-span.html" in the example zip file.

Here are two examples rules I used to create this more complex webpage:

```
#a {
  grid-column-start: 1;
  grid-row-start: 1;
  grid-row-end: 3;
}
…
#c {
  grid-column-start: 1;
  grid-column-end: 4;
  grid-row-start: 3;
}
```

The end row or end column is not included in the element. So element #a starts in row 1 and actually ends *before* row 3 – it covers rows 1 and 2. Similarly element #c starts in column 1 and ends *before* column 4 – it covers columns 1, 2, and 3.

If you're not planning to develop websites after this class, you can skip the next two alternatives and move directly to the section on different ways of setting up the grid.

Using grid-row and grid-column Properties

We can do the same thing with a bit less typing by using the grid-row and grid-column properties. The version of the webpage using this technique is stored as the file "03-grid-span-alt.html" in the example zip file.

grid-column is a shorthand way of combining grid-column-start and grid-column-end. Use it by specifying the starting column followed by a slash '/' followed by the end column like this:

```
grid-column: 1 / 4;
```

This is exactly equivalent to:

```
grid-column-start: 1;
grid-column-end: 4;
```

grid-row works the same way for rows.

```
grid-row: 1 / 3;
```

is the same as:

```
grid-row-start: 1;
grid-row-end: 3;
```

Using span

We can specify the number of rows or columns spanned as an alternative to specifying the end row. The version of the webpage using this technique is stored as the file “04-grid-span-alt-2.html” in the example zip file.

This technique uses the same grid-row and grid-column CSS attributes as the last technique. The difference is that instead of specifying an end row, it specifies how many rows are spanned. I can write:

```
grid-column: 1 / span 3;
```

instead of:

```
grid-column: 1 / 4;
```

or

```
grid-row: 1 / span 2;
```

instead of:

```
grid-row: 1 / 3;
```

Specifying the Grid

We’ve previously seen that we can explicitly specify the measurements of each row and column using the grid-template-columns and grid-template-rows CSS properties:

```
body {
  display: grid;
  grid-template-columns: 100px 100px 100px;
  grid-template-rows: 100px 100px 100px;
}
```

We are not limited to specifying measurements in pixels — we can use any of the standard CSS measurement units such as inches or centimeters. Specification in percentages can be used for providing the column widths, but not for row heights – see the example: “05-grid-define-percent.html”:

```
body {
  display: grid;
  grid-template-columns: 33% 33% 33%;
  grid-template-rows: 1in 2cm 10mm;
}
```

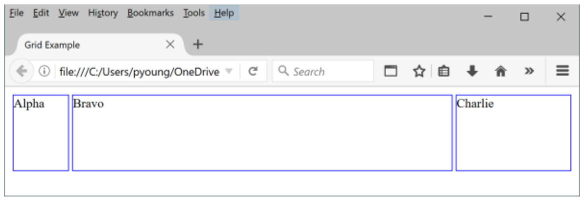
Fractional Measurements with fr

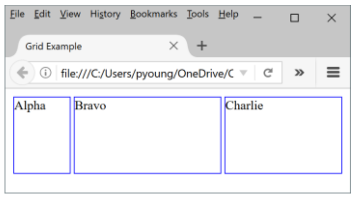
One design technique that websites sometimes use it to set fixed widths on some of the columns, and to let one or more of the other columns expand to fill the remaining space. We can get this by using the special fr measurement (which is an abbreviation for fraction).

In “06-fr-basic.html”, for example, I’ve specified that my left column should be 100 pixels, my right column should be 150 pixels and the center column should expand out to take as much space as possible.

```
body {
  display: grid;
  grid-template-columns: 75px 1fr 150px;
  grid-template-rows: 100px;
}
```

Here’s how this will appear in two different browser widths, notice the far-left and far-right columns remain the same size, while the center column expands or shrinks as needed.

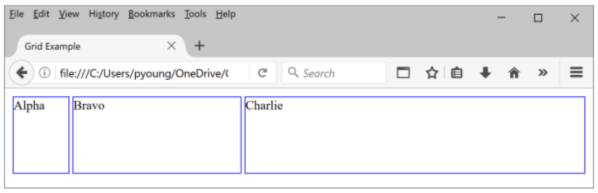

File Edit View History Bookmarks Tools Help
Grid Example
file:///C:/Users/pyoung/OneDrive
Search
Alpha
Bravo
Charlie

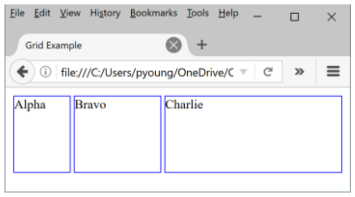

File Edit View History Bookmarks Tools Help
Grid Example
file:///C:/Users/pyoung/OneDrive/C
Alpha
Bravo
Charlie

You can divide the remaining space into as many fr units as you want. In “07-fr-adv.html” I’ve specified that the far-left column should remain at 75 pixels, and that the remaining space should be divided between the center and right columns, with the right column allocated for twice as much space (by setting it to “2fr”) as the center column (which is “1fr”):

```
body {
  display: grid;
  grid-template-columns: 75px 1fr 2fr;
  grid-template-rows: 100px;
}
```

Here’s the webpage displayed at two different web browser widths:


File Edit View History Bookmarks Tools Help
Grid Example
file:///C:/Users/pyoung/OneDrive/(
Search
Alpha
Bravo
Charlie


File Edit View History Bookmarks Tools Help
Grid Example
file:///C:/Users/pyoung/OneDrive/C
Alpha
Bravo
Charlie

Using Auto for Row Heights

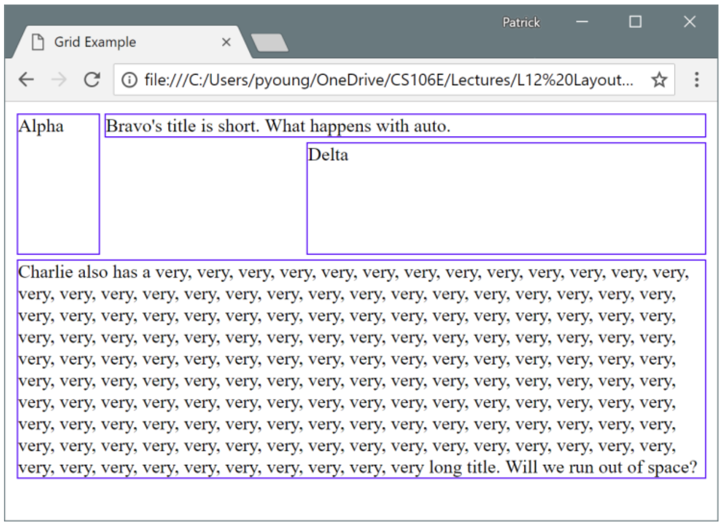
Determining how tall a row should be can be difficult. This can be particularly problematic if the website is designed to accommodate different news stories of different lengths that will be swapped in and out daily. We can skip heights on rows altogether by simply specifying:

```
grid-template-rows: auto;
```

With this setting, the web browser will lengthen each row as long as necessary to fit the longest of the elements in the row. You can also mix auto and fixed layout. In “08-auto.html” I’ve specified that the first and third rows should be as tall as they need to be, but the second row should be 100 pixels:

```
body {
  display: grid;
  grid-template-columns: 75px 1fr 2fr;
  grid-template-rows: auto 100px auto;
}
```

Here's an example of what the output might look like. The placement of the elements on the grid is the same as the one we used in the "Spanning Columns and Rows" section, however, I've changed the element contents to make the example a bit more interesting. Notice that the rows set by auto can either be much shorter or much taller than the preset 100-pixel tall row.


Patrick
Grid Example
file:///C:/Users/pyoung/OneDrive/CS106E/Lectures/L12%20Layout...

Alpha

Bravo's title is short. What happens with auto.

Delta

Charlie also has a very, very, very, very, very, very, very, very, very, very, very, very, very,
very, very, very, very, very, very, very, very, very, very, very, very, very, very, very, very,
very, very, very, very, very, very, very, very, very, very, very, very, very, very, very, very,
very, very, very, very, very, very, very, very, very, very, very, very, very, very, very, very,
very, very, very, very, very, very, very, very, very, very, very, very, very, very, very, very,
very, very, very, very, very, very, very, very, very, very, very, very, very, very, very, very,
very, very, very, very, very, very, very, very, very, very, very, very, very, very, very, very,
very, very, very, very, very, very, very, very, very, very, very, very, very, very, very, very,
very, very, very, very, very, very, very, very, very, very, very, very, very, very, very, very,
very, very, very, very, very, very, very, very, very, very long title. Will we run out of space?

Other Layout Options

CSS Grid Layout provides some other options which you may want to explore further. We can specify minimum and maximum sizes for columns, allowing them to expand or contract within the limits we provide. If we have a lot of rows and columns to specify, we can have them auto-generated as long as we provide preset row and column widths. We can even repeat patterns of widths or heights over and over again, for example repeating the pattern of 50px, 150px, 25px over and over to create many columns or many rows.

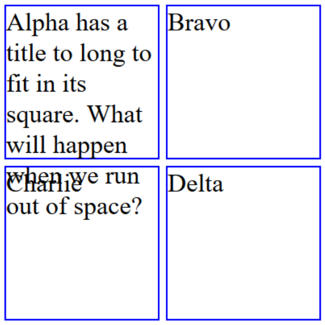
Additional Issues

Nesting

Keep in mind that while we set the display: grid on the <body> for all our examples in this handout, this layout method can actually be used on other elements as well. In fact, if an element we layout with our grid has its own internal elements, it can lay them out with its own internal grid. We can also mix Grid Layout with the other techniques I discussed in class such as Flexbox layout.

Overflow

As we've seen, we can specify both the height and the width of a grid element. What happens when the element we place in the grid doesn't actually fit? Under normal circumstances, the element overflows past the grid element's boundaries, spilling on top of other elements. In the file "09-grid-overflow.html" I've put too much text in the top-left corner of my grid and it's spilling into the next row:


Alpha has a title to long to fit in its square. What will happen when we run out of space?

Bravo

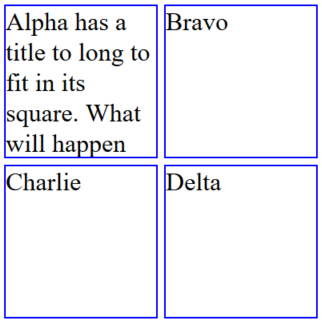
Charlie

Delta

If you don't want this to happen, the first thing you can do is make sure there's plenty of space – for example specifying widths for your columns, but leaving your row heights to auto. As an alternative, you can set an element's overflow CSS property to hidden. "10-grid-overflow-hidden.html" is the same as before, except now I use the rule:

```
#a {
  grid-column-start: 1;
  grid-row-start: 1;
  overflow: hidden;
}
```

This will hide any part of the element that does not fit in its assigned grid space. However, note that this will cut-off any extra text, which may or may not be what you want. Here's the same page as before with overflow: hidden.


Alpha has a title to long to fit in its square. What will happen

Bravo

Charlie

Delta

To Learn More

The Mozilla developer website has a good detailed description of Grid Layout:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout