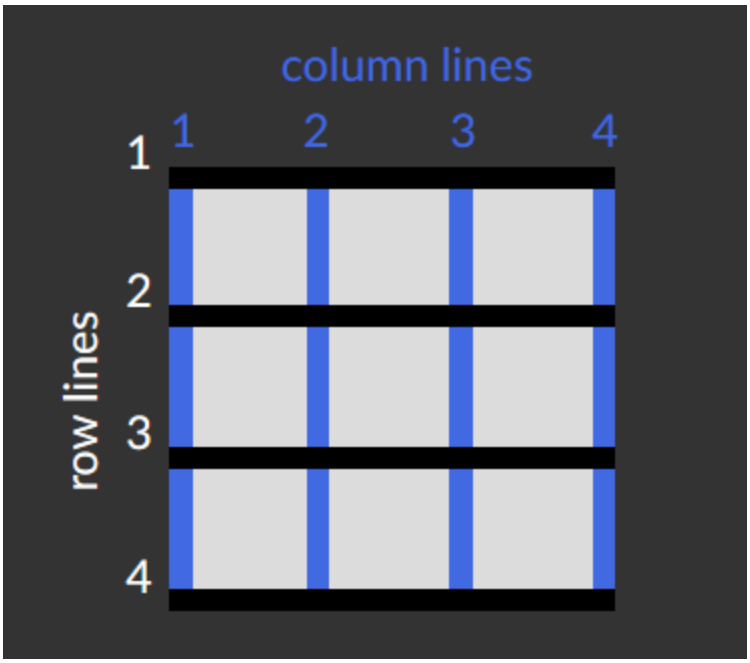
CSS Grid

Created By	Nate L
Last Edited	@Jan 4, 2019 5:37 PM

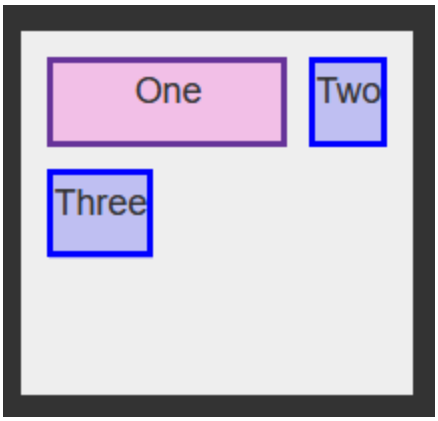
- Quick note, I would recommend using grid and flexbox if you can (grid for the overall structure of the document and flexbox for the finer details). A number of people seem to have thought one better than the other, but they actually compliment each other very well
  - Makes responsiveness much easier as well
  - To get an idea, would recommend this video: https://www.youtube.com/watch?v=dQHtT47eH0M
- In order to use CSS grid and turn an element into a grid container, set its display property to grid
  - The parent element is known as the container and its children are called items
- In order to use grid you need to use the `grid-template-columns` property on a grid container, this defines the width of the columns. You can also define the height of rows with `grid-template-rows`
  - You can use flex `<flex>` data type (which represents a fraction of the leftover space in the grid container), which is represented by the unit fr
  - You can use auto or percentages for column or row sizing

```
/* Example */

.container {
  display: grid;

  /* defines width of 3 columns, # of values determines # of columns */
  grid-template-columns: auto 2fr 50px;

  /* defines height for 3 rows */
  grid-template-rows: 2fr 1fr 20%;
}
```

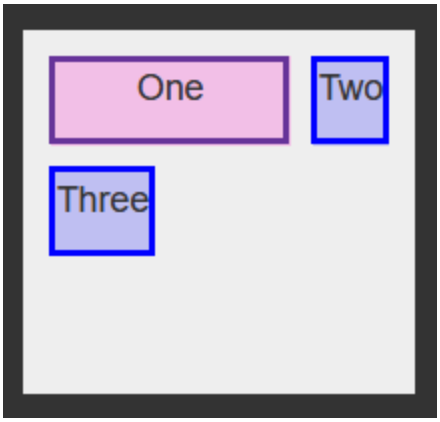
- You can create space between grid columns or rows with the `grid-column-gap` and `grid-row-gap` properties
- You can use `grid-gap` as a shorthand method of applying both gap properties within one declaration
  - If given one value it'll apply to both columns and rows
  - If give two values, the first will apply to rows, the second to the columns
- The `grid-column` property is a **shorthand** property for both `grid-column-start` and `grid-column-end` specifying a grid item's size and location with the grid column (specifying the inline-start and inline-end edge of its grid area)
  - In a 3×3 grid this is how the grid is divided



  - If we use `grid-column: 1 / 3;` on Item 1 it would look like this:



- The `grid-column-start` property specifies the grid item's start location within the grid column by contributing a line, span, or nothing (auto) to its grid placement
    - The start position defines the block-start edge of the grid-area, you can use span and an integer for the starting position to take up multiple spaces, ex: `grid-column-start: span 2;`



    - If using a negative integer for the start position, it counts in reverse starting from the end edge of the grid
        - This also works for `grid-column-end`
- The same syntax can be applied to `grid-row` in order to position a row and/or adjust the sizing of the grid item
- The content of a grid item is located in a box which is referred to as a cell; to align the content's position horizontally, use the `justify-self` property on a grid item (this is applied to a cell, unlike the other justify properties)

    - By default, this property has a value of `stretch`, which will make the content fill the whole width of the cell
    - Others can be found here: https://developer.mozilla.org/en-US/docs/Web/CSS/justify-self
        - Some values that are accepted are start, center, end
- To align an item vertically, use the `align-self` property

Key differences between `justify-content`, `justify-items` and `justify-self` in CSS Grid:

- The `justify-content` property controls the alignment of grid columns. It is set on the *grid container*. It does not apply to or control the alignment of grid items.
- The `justify-items` property controls the alignment of grid items. It is set on the *grid container*.
- The `justify-self` property overrides `justify-items` on individual items. It is set on *grid items* and inherits the value of `justify-items`, by default.

- Likewise for `align-items`, `align-self`, and `align-content`
- You can group cells of the grid together into an area (grid areas are one or more grid cells that make up a rectangular area on the grid)
    - Grid areas merge cells together to be more specific, you can specify an empty grid cell with a dot, ex:

```
grid-template-areas:
"header header header"
"  .  content content"
"footer footer footer"
```

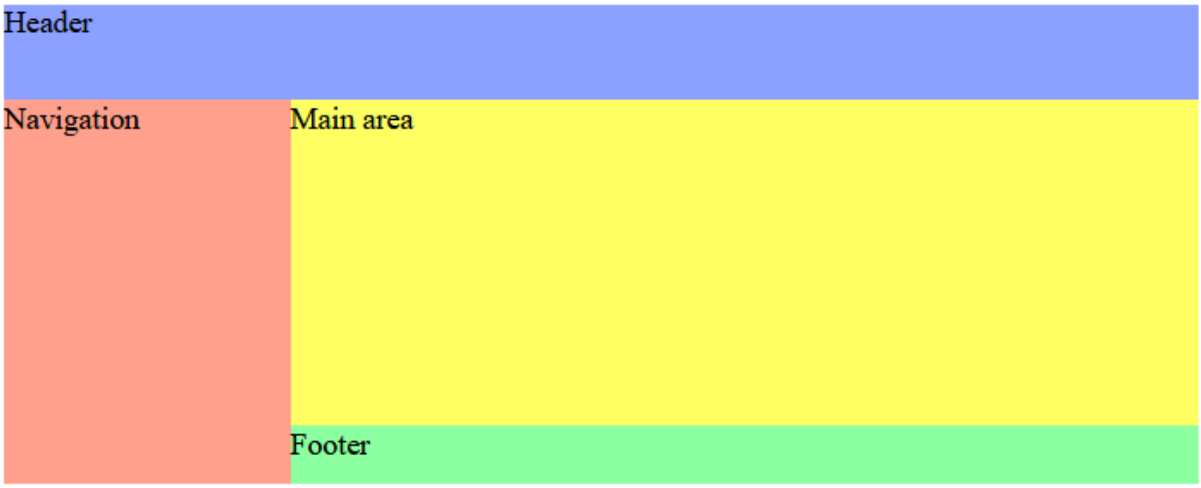
    - When using `grid-template-areas`, you have to specify which element goes where with the CSS Grid property, ex: `grid-area: header;`

- Example and usage:

```
/* Example page */
<section id="page">
  <header>Header</header>
  <nav>Navigation</nav>
  <main>Main area</main>
  <footer>Footer</footer>
</section>
```

```
#page {
  display: grid;
  width: 100%;
  height: 250px;
  /* create grid areas */
  grid-template-areas: "head head"
                       "nav  main"
                       "nav  foot";
  grid-template-rows: 50px 1fr 30px;
  grid-template-columns: 150px 1fr;
}

#page > header {
  /* Specify what elements belong in a grid area,
     do this for each element  */
  grid-area: head;
  background-color: #8ca0ff;
}

#page > nav {
  grid-area: nav;
  background-color: #ffa08c;
}

#page > main {
  grid-area: main;
  background-color: #ffff64;
}

#page > footer {
  grid-area: foot;
  background-color: #8cffa0;
}
```

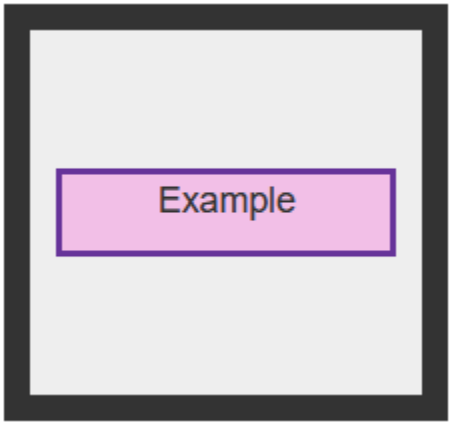
- Results in this:



- You can still use `grid-area` without specifying a template by using integers similar to `grid-column` and `grid-row`
  - Syntax:

```
grid-area: horizontal line to start at
/ vertical line to start at
/ horizontal line to end at
/ vertical line to end at;

grid-area: 2 / 1 / 2 / 4;

/*
The above would start on the 2nd horizontal line and 1st vertical
It would end at the 2nd horizontal line and 4th vertical
*/
```

  - Result:



- You can reduce repetition by using the `repeat` function in conjunction with the `grid-template-columns` and `grid-template-rows` to specify how many columns/rows and their width/height accordingly
  - Repeat's syntax: `repeat(# of columns/rows, value to be repeated);`
  - Example: `grid-template-columns: repeat(200, 30px);`, this would create 200 columns at 30px width each
  - Another example: `grid-template-rows: repeat(10, 1fr, 40px) 70px;`, this would create 10 repeating rows of 1fr and 40px with a single 70px row at the very end (21 rows total)
- You can use another function for `grid-template-columns/rows` called minmax, this is used to limit the size of items when the grid container changes size, this helps create more responsive columns
  - Ex: `grid-template-column: minmax(20px, 100px);`, at minimum the column will be 20px and at max will be 100px
    - Can also be used with the `repeat` function
  - Using `auto-fill` with `minmax` allows for the creation of flexible layout, this property allows you to automatically insert as many rows or columns as possible depending on the container
  - Ex: `grid-template-columns: repeat(auto-fill, minmax(20px, 1fr));`, will keep inserting columns and stretching them until it can insert another
    - If the container can't fit all the items on one row it will move them down to a new one

        - If there's empty space in the container after the maximum size of the columns are reached, it'll place empty columns in the empty space
    - Another property, `auto-fit` works similarly to auto-fill with the difference of instead of placing empty columns/rows, it'll stretch your items to fit the size of the container
        - If all items can't fit on one line, they'll move down to a new one
- For a more responsive site, you can use media queries to rearrange grid areas
- When you turn an element into a grid, it only affects the behavior of its direct descendants
    - You can nest grids by turning direct descendants into grids as well