So far, we have :

1. Added elements to a webpage.

2. Placed these elements into groups called 'div's.

3. Assigned class labels to elements on a webpage

4. Applied style rules that change how these elements look- mostly changing color, font, borders, sizing.

Let's dig a little deeper into css to cover using css for defining behaviors and layouts.

CSS and Changes in Appearance on Interaction

Css can set rules for the appearance of your page, but you can also use it to specify some interactive changes. The way this is often done is through **pseudo classes.**

Here's a w3 schools reference : https://www.w3schools.com/css/css_pseudo_classes.asp

Suppose you want to change the appearance of a class when elements of that class are hovered over- like changing their color. In addition to writing style rules that apply to a class, you can write specific rules that apply to that class when you are hovering over it.

Here is the html, which creates a div with a class.

```
<div class="myDivClassName">
       <h1> I am some text </h1>
   </div>
```

Then we want to define style rules for the default look of the class, and for when hovering over the div. We can do this using an additional label, which is called a pseudo class.

```
.myDivClassName{
    background-color: orange;
}

.myDivClassName:hover{
    background-color: blue;
}
```

Here is the result, without hovering:

While hovering:

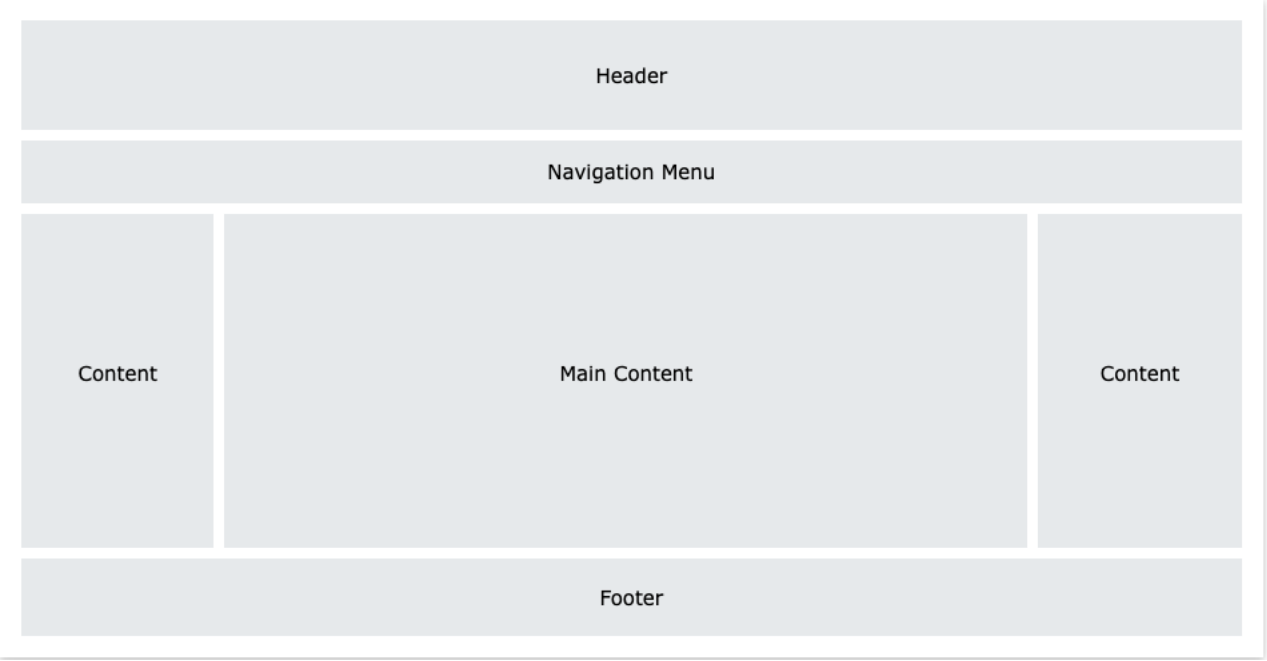
CSS and Layout

The next natural question is- how do I layout these elements on the page? How do I manage what happens when the page resizes?

There are a number of different options here.

Layout option 1: Use CSS

With CSS, you can create a page layout, that includes a header, a footer, and multiple columns in the body.



W3 schools has a nice description here : https://www.w3schools.com/css/tryit.asp?filename=trycss_website_layout_blog

Let's look at the layout above.

First, it has a header and a footer. These take up the whole page, and the main layout setting here is the placement of the text.

```
.header {
  background-color: #F1F1F1;
  text-align: center;
  padding: 20px;
}
```

But what about those columns in the middle? How do we make that work? Let's make an example. Suppose this is my html:

```
  <div>
    <h2>Column 1</h2>
    <p>I am a column which has lots of excellent content.  Read me!  I have so many interesting things to say. </p>
  </div>

  <div>
    <h2>Column 2</h2>
    <p>I am just a boring middle column.  Ignore me.  I am sure the other columns have more to say than I do.  Bla Bla Bla. </p>
  </div>

  <div>
    <h2>Column 3</h2>
    <p>Ok, Ok, Do what you want.  Read me.  Don't read me.  I do not care.  I don't think about other people and just focus on being the best column of te
  </div>
```

By default, if each of those columns is a div, they will be placed on top of each other, like this:

Column 1

I am a column which has lots of excellent content. Read me! I have so many interesting things to say.

Column 2

I am just a boring middle column. Ignore me. I am sure the other columns have more to say than I do. Bla Bla Bla.

Column 3

Ok, Ok, Do what you want. Read me. Don't read me. I do not care. I don't think about other people and just focus on being the best column of text I can be no matter who r me.

Let's override that default behavior by specifying layout behavior in css.

First- we need to group these into a div called 'row' (since we want all 3 in the same row). And then we will call each one a column. Here's the new html:

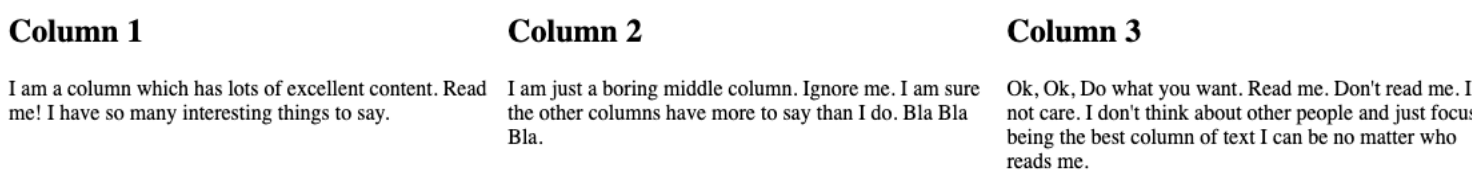
```
<div class="row">
  <div class="column">
    <h2>Column 1</h2>
    <p>I am a column which has lots of excellent content.  Read me!  I have so many interesting things to say. </p>
  </div>
```

```
  <div class="column">
    <h2>Column 2</h2>
    <p>I am just a boring middle column.  Ignore me.  I am sure the other columns have more to say than I do.  Bla Bla Bla. </p>
  </div>

  <div class="column">
    <h2>Column 3</h2>
    <p>Ok, Ok, Do what you want.  Read me.  Don't read me.  I do not care.  I don't think about other people and just focus on being the best column of tex
  </div>
</div>
```

Now the css, which specifying the layout of the columns in the row.

```
/* Create three equal columns that floats next to each other */

.column {
  float: left;
  width: 33%;
}
```

And the result:



What if you want columns to have different sizes? Let's try!

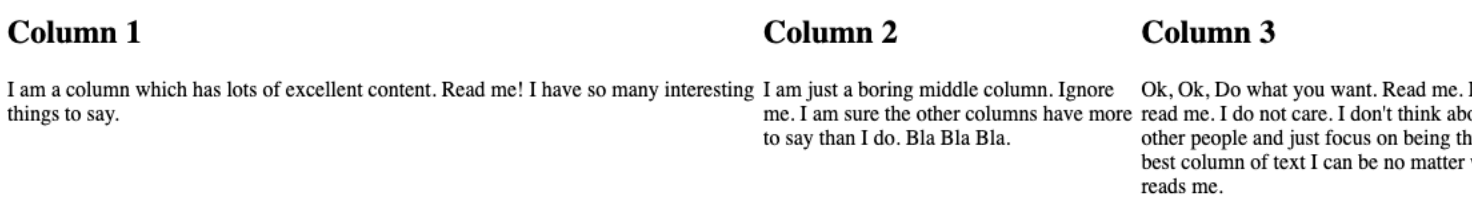
Html:

```
div class="row">
  <div class="bigColumn">
    <h2>Column 1</h2>
    <p>I am a column which has lots of excellent content.  Read me!  I have so many interesting things to say. </p>
  </div>

  <div class="smallColumn">
    <h2>Column 2</h2>
    <p>I am just a boring middle column.  Ignore me.  I am sure the other columns have more to say than I do.  Bla Bla Bla. </p>
  </div>

  <div class="smallColumn">
    <h2>Column 3</h2>
    <p>Ok, Ok, Do what you want.  Read me.  Don't read me.  I do not care.  I don't think about other people and just focus on being the best column of tex
  </div>
</div>
```

CSS:

```
.bigColumn{
  float: left;
  width: 50%;
}

.smallColumn{
  float: left;
  width: 25%;
}
```

The result:



To recap- we put the elements we wanted to layout with respect to each other into a div together, and then described the rules for them using css.

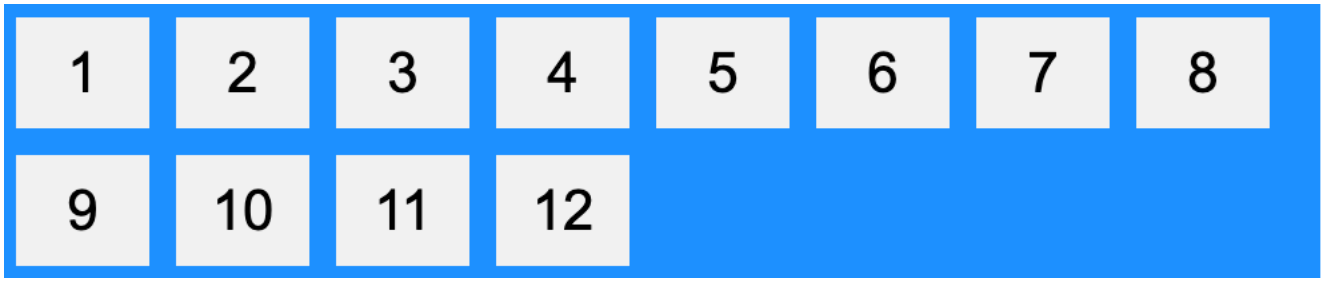
There are LOTS of positioning rules in CSS.

You also CAN set responsive layout rules using CSS. **What is responsive layout?** A responsive layout is one where as the page resizes, elements shift and resize in useful ways, so you can look at the page comfortably on a phone, on a small laptop screen or ok a 4k computer monitor.

Look at this example, where the three columns become one row, when the page is made smaller.

https://www.w3schools.com/css/tryit.asp?filename=trycss_website_layout_grid

Layout option 2: Use CSS Flexbox



CSS flexbox is a layout module that allows you to have more control over the layout of elements.

W3 schools has a tutorial: https://www.w3schools.com/css/css3_flexbox.asp

The principle is: you define a container which will contain other elements, and call it a flex container. This container can have settings for how to wrap elements if they exceed the space provided. The container can have settings for how to *justify* the content- centered, at the start (flex start), at the end (flex end). You can have the content stretch and fill the container horizontally or vertically. And more.

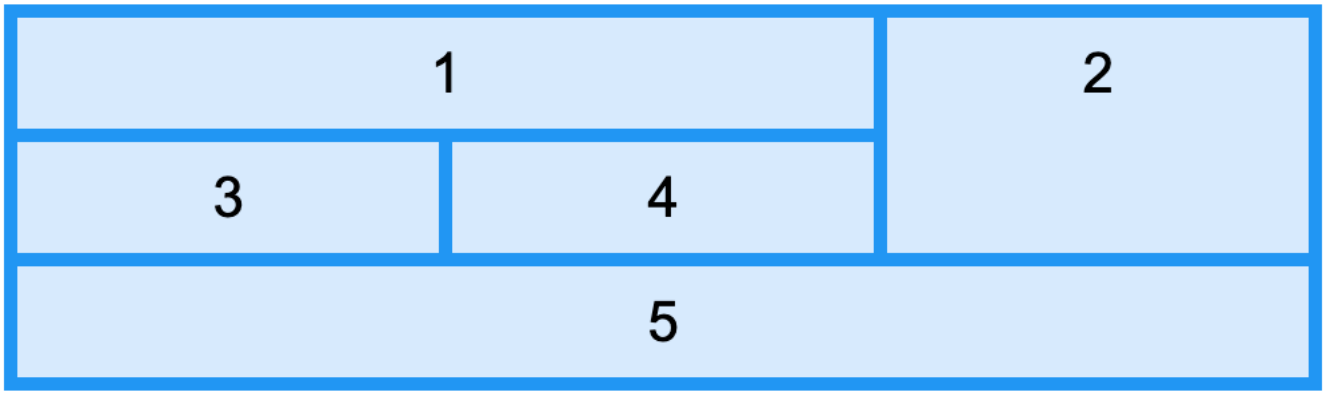
https://www.w3schools.com/css/css3_flexbox_container.asp

Items in the flex container can also have properties.

https://www.w3schools.com/css/css3_flexbox_items.asp

The result is- you can customize the sizing, positioning and behavior of elements within a parent element.

Layout option 3: Use CSS Grid



With CSS grid, you create a virtual grid, of columns and rows. Then you specify that the items within can take up varied numbers of these columns and rows.

Details here:

https://www.w3schools.com/css/css_grid.asp

Layout option 4: Use a CSS framework like Bootstrap

Bootstrap is a CSS framework, which means it provides responsive layout BUT ALSO a lot more. Bootstrap is one of the most common web dev tools in use today.
https://www.w3schools.com/bootstrap/bootstrap_ver.asp

Bootstrap may or may not be a good choice for what you are going to do. It is ideal for cases where you want to get a site out quickly that doesn't involve much custom styling or layouts or behavior. Sometimes when you want something very specific, you may find yourself working against Bootstrap, rather than really benefiting from it.

Layout option 5: MORE ????

The internet is big. There are lots of tools out there.