CSYM019
Internet Programming

Muawya Eldaw
muawya.eldaw@northampton.ac.uk

Week 3 - CSS

- Quick CSS recap
- CSS Box model
- CSS Units
- Browser developer tools
- Layouts with display: grid

HTML

- A basic valid HTML file looks like this:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>My Web Page!</title>
    <meta charset="UTF-8" />
  </head>

  <body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>My Web Page!</title>
    <meta charset="UTF-8" />
    <link rel="stylesheet" href="styles.css" />
  </head>

  <body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

Name of the CSS file
(should have a .css extension*)

The “rel” (relationship) tells
The browser what type of
File is being referenced

Link tag

CSS file

- It is recommended that a CSS file is placed in the same folder as the HTML file.
- A CSS file should be given a .css extension.
- Use your editor (Atom, VS Code, Sublime – see last week's notes) to create the CSS file

CSS real example

- A selector can be a tag name
- One property is “color” (Note the American spelling!)
- To set H1 elements to red and paragraphs to blue:

```
h1 {
    color: red;
}

p {
    color: blue;
}
```

CSS properties

- When you target a tag it will affect any tag of that type:

```
h1 {
    color: red;
}

p {
    color: blue;
}
```

Page heading

Paragraph 1

Paragraph 2

```
<body>
  <h1>Page heading</h1>
  <p>Paragraph 1</p>
  <p>Paragraph 2</p>
</body>
```

CSS Box model

- CSS has several properties which perform a very similar task: Add gaps around an element
- These properties are:
  - Border
  - Padding
  - Margin
- All three of these can be applied to the same element

Box model example

```
div {
    background-color: darkblue;
    color: white;
}
```

```
    <div>Element 1</div>
    <div>Element 2</div>
```

Element 1
Element 2

Margin

- Margin is used to describe the gap *between different elements:*

```
div {
    background-color: darkblue;
    color: white;
}
```


```
div {
    background-color: darkblue;
    color: white;
    margin: 20px;
}
```

Padding

- Padding is used to describe the gap *inside the element around the text*

```
div {
    background-color: darkblue;
    color: white;
}
```

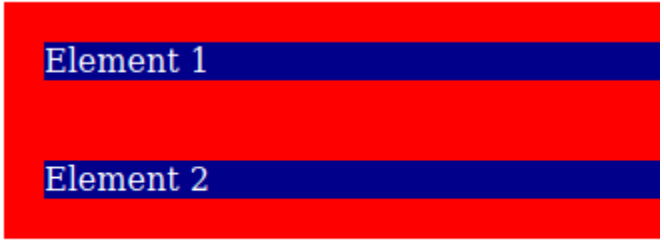

```
div {
    background-color: darkblue;
    color: white;
    padding: 20px;
}
```

Border

- Border is used to draw a border around the element with a specific colour:

```
div {
    background-color: darkblue;
    color: white;
}
```


```
div {
    background-color: darkblue;
    color: white;
    border: 20px solid red;
}
```

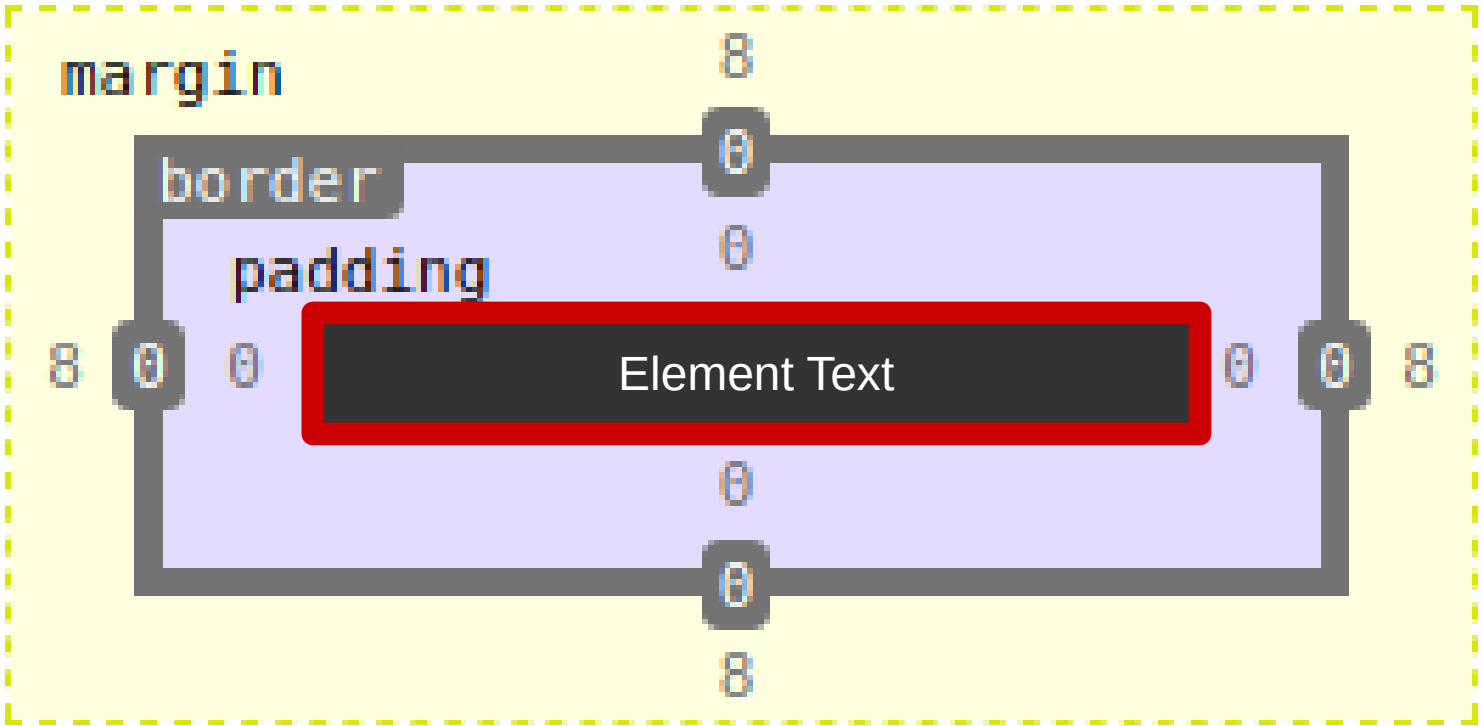


Box model

- These can all be applied to the same element:

```
div {
    background-color: darkblue;
    color: white;
}
```


```
div {
    background-color: darkblue;
    color: white;
    border: 20px solid red;
    padding: 20px;
    margin: 20px;
}
```


margin
8
border
0
padding
0
8
0
0
Element Text
0
0
8
0
0
8

CSS units

- All my examples used px – pixels
- This is generally a bad idea because all screens are different sizes and people use different font sizes
- Instead, you can use *em*
- Which is the width of an *m* character
- This is useful when you want to apply consistent spacing regardless of font sizes

px

```
p {
    background-color: darkblue;
    color: white;
    padding: 5px;
}

h1 {
    background-color: darkred;
    color: white;
    padding: 5px;
}
```

```
    <h1>Heading</h1>
    <p>Paragraph</p>
```

em

```
p {
    background-color: darkblue;
    color: white;
    padding: 1em;
}

h1 {
    background-color: darkred;
    color: white;
    padding: 1em;
}
```


```
<h1>Heading</h1>
<p>Paragraph</p>
```

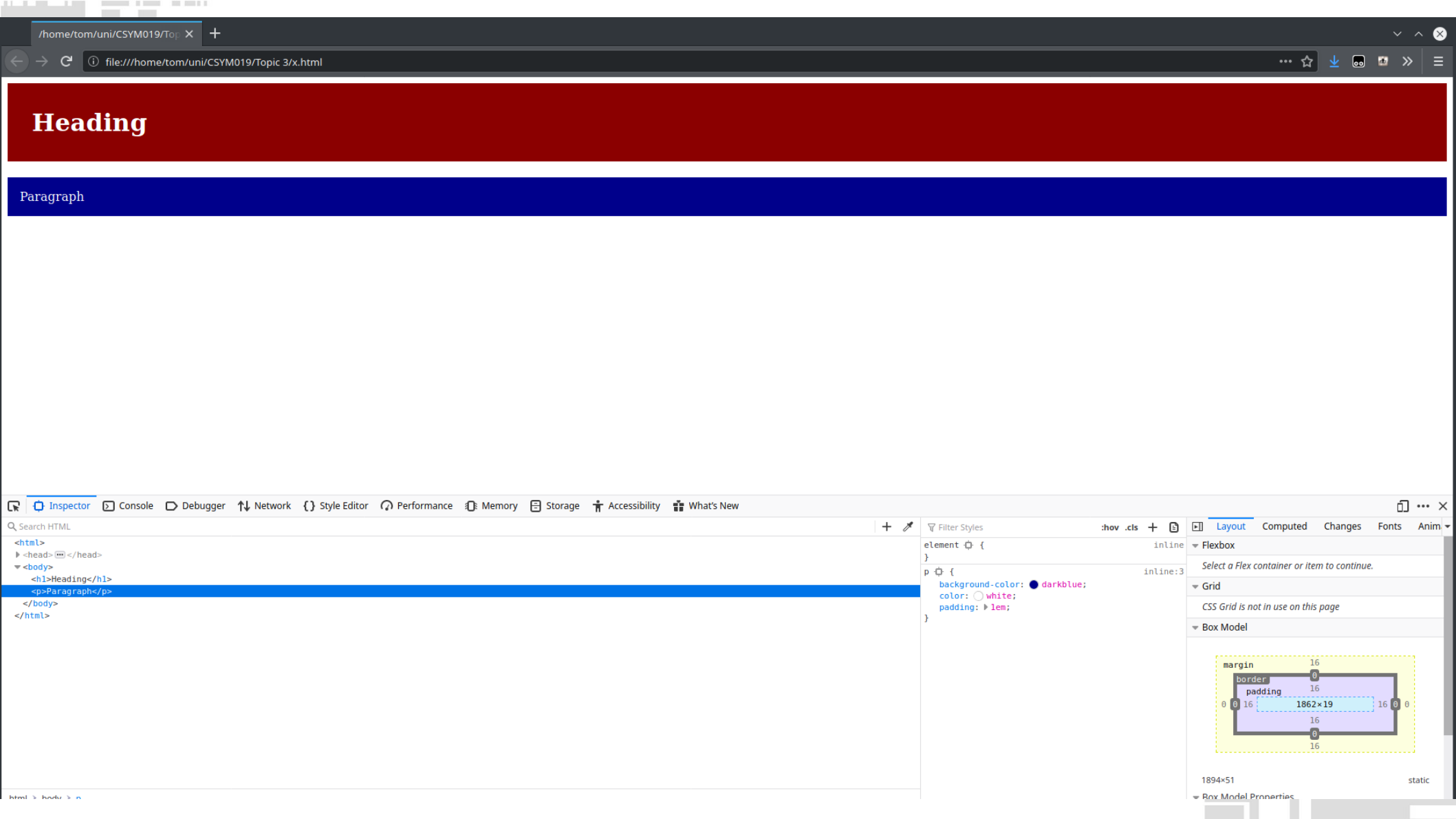
Using EM means the padding
Is proportional to the size
of the text

There are many different units in CSS

- px – measurement in pixels (e.g. 80px)
- pt font point (only really useful for fonts) (e.g. 12pt)
- % percentage of containing element size e.g. (50%)
- vh/vw – Viewport width and viewport height (100vw is 100% the width of the browser)
- This not a complete list!
- Further reading: https://css-tricks.com/the-lengths-of-css/

Browser developer tools

- Modern web browsers contain a lot of tools for web developers to help them build their websites
- You can open the developer tools in most browsers by pressing F12 on the keyboard or right clicking and selecting “inspect element”


/home/tom/uni/CSYM019/Top
file:///home/tom/uni/CSYM019/Topic 3/x.html
Heading
Paragraph
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
What's New
Search HTML
<html>
<head> </head>
<body>
<h1>Heading</h1>
Paragraph
</body>
</html>
Filter Styles
:hov .cls
element { inline
}
p { inline:3
background-color: darkblue;
color: white;
padding: 1em;
}
Layout
Computed
Changes
Fonts
Anim
Flexbox
Select a Flex container or item to continue.
Grid
CSS Grid is not in use on this page
Box Model
margin
border
padding
1862×19
1894×51
static
Box Model Properties

Layouts

- Last week I tasked you with trying to create a page with the following layout:

First: Back to HTML!

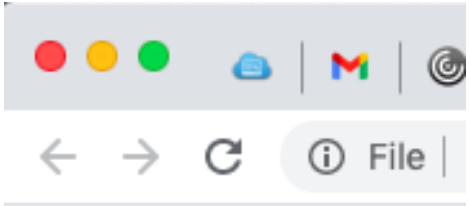
- There are specific HTML elements that should be used to signify different parts of the page:
- <header> for header sections
- <nav> for navigation bars
- <main> for the main content area
- <aside> for sidebars*
- <footer> for the page footer

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <header>
      <h1>Heading</h1>
    </header>

    <nav>
      <ul>
        <li>
          <a href="#">Link 1</a>
        </li>
        <li>
          <a href="#">Link 2</a>
        </li>
        <li>
          <a href="#">Link 3</a>
        </li>
      </ul>
    </nav>
    <main>
      <p>Lorem ipsum....</p>
    </main>

    <aside>
      Right hand side
    </aside>

    <footer>
      &copy; Your Name 2020
    </footer>
  </body>
</html>
```



Heading

- Link 1
- Link 2
- Link 3

Lorem ipsum....

Right hand side
© Your Name 2020

Add colours to see what is what

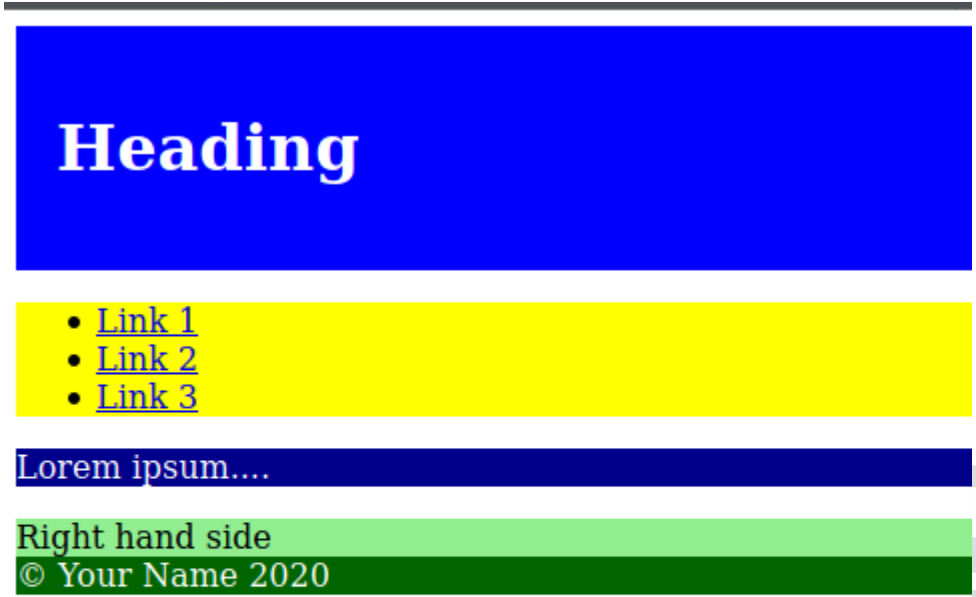
```css
header {
 background-color: blue;
 color: white;
 padding: 20px;
}


nav {
 background-color: yellow;
}

main {
 background-color: darkblue;
 color: white;
}

aside {
 background-color: lightgreen;
}

footer {
 background-color: darkgreen;
 color: white;
}
```



A quick caveat

- There are many, many ways to achieve this including:
  - **position: absolute** (never do this, it will never display correctly in all browsers)
  - **display: table**
  - **display: flex**
  - **display: grid**
- I am only showing you the arguably best option.

display: grid

- Display: grid is a very flexible but quite difficult solution. I will show the simplest way of using it
- Firstly, you need to set the containing element, of the elements you want to position, to display: grid
- In our case it’s the body element

```
body {
 display: grid;
}
```

grid-area

- Next, you label each element you want to position using grid-area
- These names are chosen by you and referenced later on
- It’s good practice to just use the element names

```css
header {
 background-color: blue;
 color: white;
 padding: 20px;
 grid-area: header;
}


nav {
 background-color: yellow;
 grid-area: nav;
}

main {
 background-color: darkblue;
 color: white;
 grid-area: main;
}

aside {
 background-color: lightgreen;
 grid-area: aside;
}

footer {
 background-color: darkgreen;
 color: white;
 grid-area: footer;
}
```

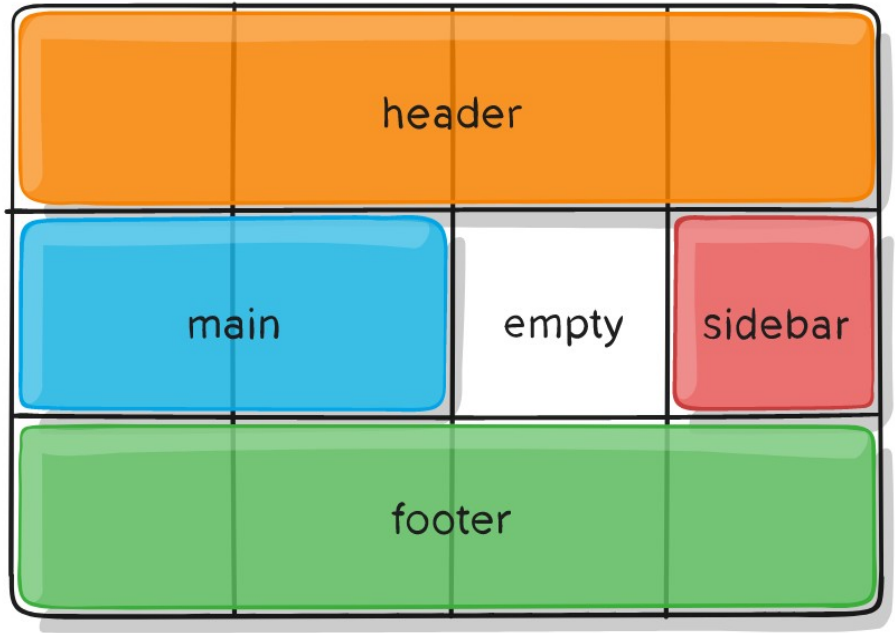
columns

- You then specify how many columns there are on the grid and their sizes on the container (body element)
- And tell it to align the elements into rows:
- (Use the row with the most number of columns!)
- 3 columns: 20% 60% 20%:

```
body {
 display: grid;
 grid-template-columns: 20% 60% 20%;
 grid-template-rows: auto;
}
```

grid

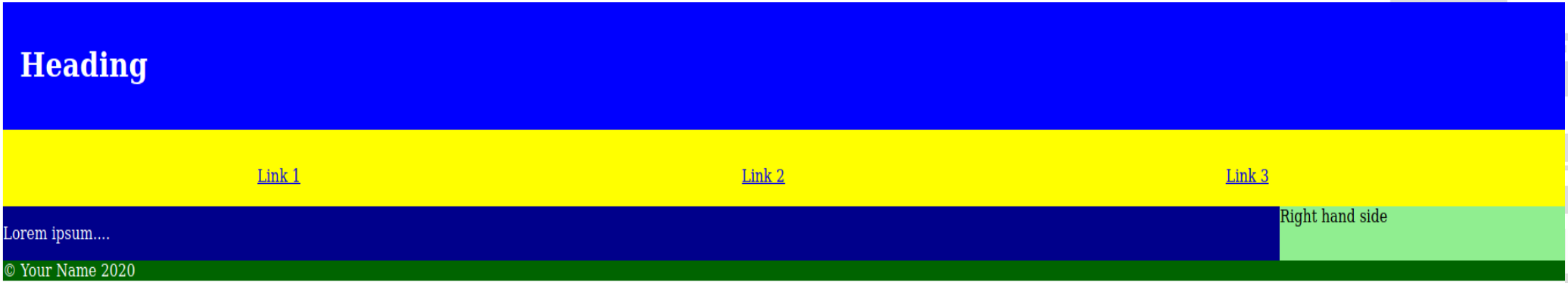
- This will create a grid with as many rows as necessary and 3 columns.
- You then specify which elements occupy which parts of the grid



```
body {
 display: grid;
 grid-template-columns: 20% 60% 20%;
 grid-template-rows: auto;
 grid-template-areas: "header header header"
                      "nav main aside"
                      "footer footer footer";
}
```

Exercises

- 1. Follow the slides to create the 3 column layout
  - Make sure you use the correct HTML elements (<header>, <main>, <nav>, <footer> and <aside>)
- 2. Once you've got the basic layout, tweak it to meet the following designs:
- a):



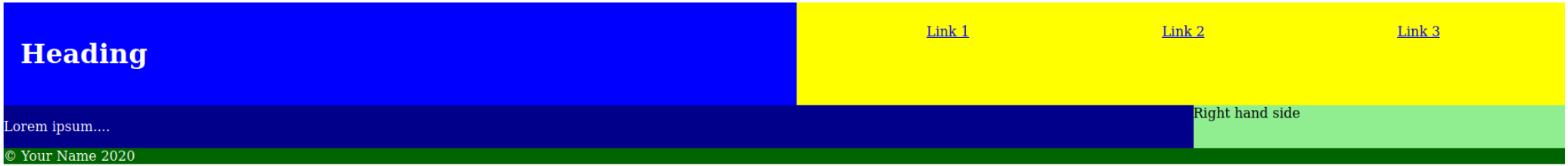
- b):


Heading
Link 1
Link 2
Link 3
Lorem ipsum....
© Your Name 2020

- c):


Heading
Link 1
Link 2
Link 3
Lorem ipsum....
Right hand side
© Your Name 2020

- d):


Heading
Link 1
Link 2
Link 3
Lorem ipsum....
Right hand side
© Your Name 2020

- Experiment with different designs and see if you can make your pages look nicer than mine (it is not a high bar!)
  - Try different paddings, margins, borders, fonts and adding real content to the pages