# CSY1018
# Web Development
# Topic 3

Tom Butler
thomas.butler@northampton.ac.uk

# Topic 3 - CSS

- CSS Box model
- CSS Units
- CSS colours
- Browser developer tools
- Semantic markup
- Layouts with display: grid

# 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;
}
```

Element 1
Element 2

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

# Margin

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

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

Element 1
Element 2

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

Element 1

Element 2

# Padding

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

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

Element 1
Element 2

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

Element 1

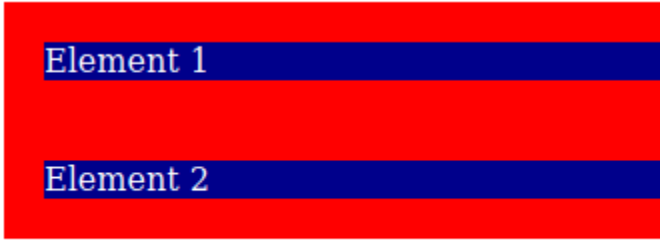
Element 2

# 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;
}
```



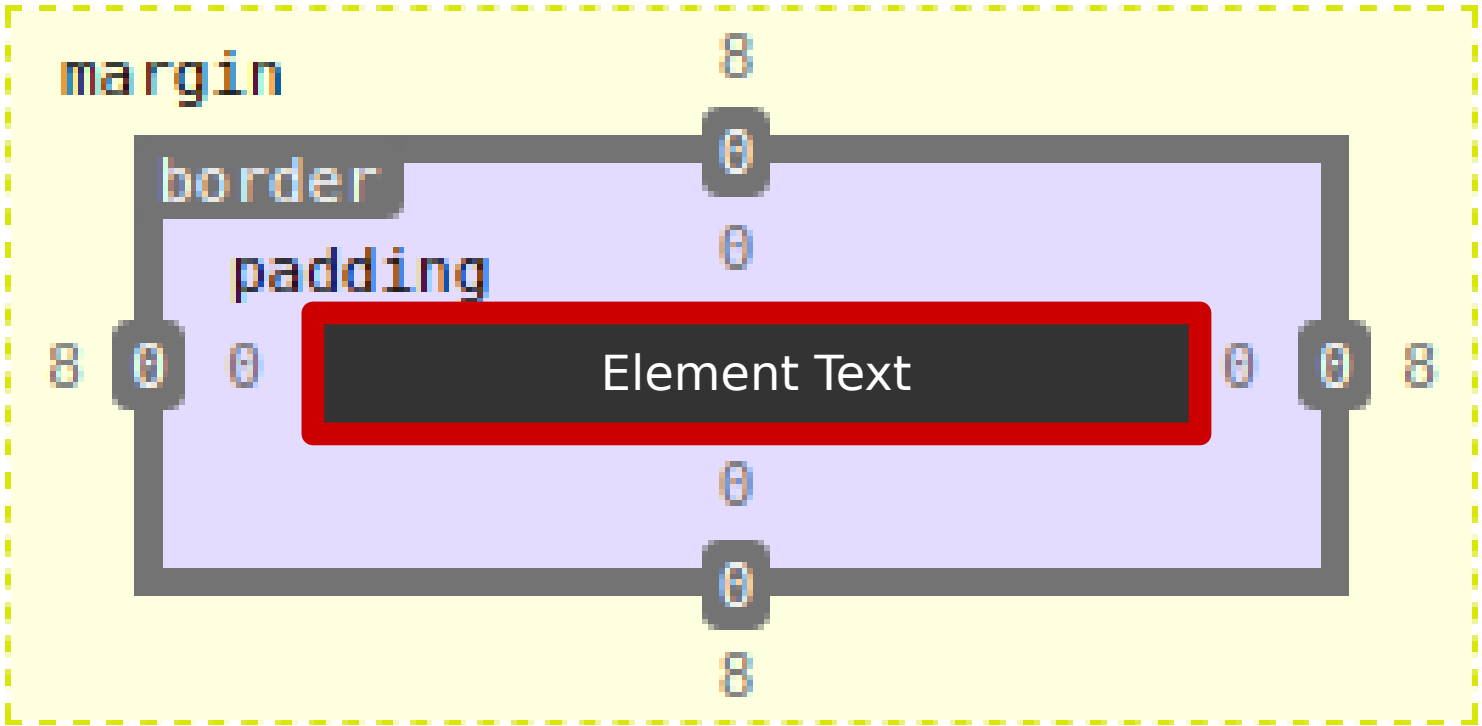
# Box model

- These can all be applied to the same element:

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

Element 1
Element 2

```
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/

# CSS Colours

- You can use several colour names in your css e.g.
  - background-color: blue
  - background-color: red
- But these are very limited and there are only a handful available.
- Modern monitors are capable of displaying 16 million different colours
  - It would be impossible to give each of them a unique name!

# CSS

- CSS uses numerical colours that describe an amount of Red, Green and Blue (RGB)
- The range is from 0-255 for each colour.
- There are three numbers in the order Red, Green, Blue.
- The range 0-255 describes the amount of each color. The value
  - 255, 0, 0 describes pure red (100% red, no blue, no green)
  - 0, 255, 0 describes pure green
  - 0, 0, 255 describes pure blue

# CSS colours

- Other colours are made up by combining different amounts of red, green and blue
- White is all three colours combined:
  - 255, 255, 255
- Black is none of each colour:
  - 0, 0, 0
- Any value where all three colours are the same will be grey:
- 50, 50, 50 (dark grey)
- 200, 200, 200 (light grey)

# CSS colours

- Other colours are made by combining different amounts of the other colours
- Yellow is 255, 255, 0 (mixing red and green)
- Pink is 255, 0, 255 (mixing red and blue)

# Colours in CSS

- In CSS you can set an RGB colour using the following code:

```
h1 {
    background-color: rgb(0, 255, 0);
    color: rgb(255, 0, 0);
}
```

- This sets the h1’s background colour to green and font colour to red

# Hexadecimal

- In addition to the RGB notation, CSS colours are more frequently written using a hexadecmical notation.
- The numbers we use daily use the *decimal* counting system where there are ten different symbols (0-9)
- Hexademical is a counting system which uses 16 different symbols
  - 0-9 then A-F

# Hexadecimal

| | |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | A |
| 11 | B |
| 12 | C |
| 13 | D |
| 14 | E |
| 15 | F |
| 16 | 10 |
| 17 | 11 |

# Hexadecimal

- In Hexadecimal, 255 is expressed as FF
- HTML/CSS colours are described using a # symbol, then 6 hexadecimal digits
- The 6 digits are three pairs of numbers from 0-255
- The first pair is red,  second is green, third is blue. Therefore:
  - #FF0000 is equivalent to red ( 255, 0, 0)
  - #00FF00 is equivalent to green (0, 255, 0)

# Hexadecimal

- Hexadecimal is the most common colour notation used in CSS
- You don’t really need to know the values as you can always look them up
- If you do a lot of work in HTML/CSS you will be able to infer the colour from the number
- This isn’t that important as you can just look up colours when you see them
- The important part is that you know what to look for.
- When you see #020445 you should know it is referring to a colour, even if you’re not sure what that colour is

# Hexadecimal

- Hexadecimal colours can be used in your CSS like so:
-

```
h1 {
    background-color: #00FF00;
    color: #FF0000;
}
```

# Finding colour codes and colour combinations

- There are lots of free colour pickers available online which will generate the colour code for you
- e.g. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Colors/Color_picker_tool
- When designing a site, you want to choose colours that work well together. A very useful resource for finding groups of colours for a design is:
- https://www.colourlovers.com/

# Hexadecimal

- Hexadecimal is the most common colour notation used in CSS
- You don’t really need to know the values as you can always look them up
- If you do a lot of work in HTML/CSS you will be able to infer the colour from the number
- This isn’t that important as you can just look up colours when you see them
- The important part is that you know what to look for.
- When you see #020445 you should know it is referring to a colour, even if you’re not sure what that colour is

# Hexadecimal

- Hexadecimal is the most common colour notation used in CSS
- You don’t really need to know the values as you can always look them up
- If you do a lot of work in HTML/CSS you will be able to infer the colour from the number
- This isn’t that important as you can just look up colours when you see them
- The important part is that you know what to look for.
- When you see #020445 you should know it is referring to a colour, even if you’re not sure what that colour is

# Fonts

- You can set the font used by an element using the font-family css property followed by the name of the font you wish to use:

```
h1 {
    font-family: Verdana;
}

p {
    font-family: Arial;
}
```

# Fonts

- For this to work, the user has to have the font you have chosen installed on their computer
- Different operating systems include different sets of fonts (e.g. MacOS does not include most of the Microsoft fonts!)
- If you have Microsoft Office installed, it installs a considerable number of additional fonts which aren’t included in a normal Windows installation

# Fonts

- You cannot rely on the user having the fonts installed on their computer
- If you want to use a font beyond a very basic selection (Arial, Verdana, Times New Roman) you can include the font on the web page
- When the user views your page, the browser downloads the font file along with the HTML and CSS
- The font can then be used by the page

# Web Fonts

- Doing so requires providing the font in a special format called *web font*
- You can convert any font you like to a webfont  using FontSquirrel’s generator:
  - https://www.fontsquirrel.com/tools/webfont-generator
- You upload the font and it converts it to a webfont for use on your pages

# Generating web fonts

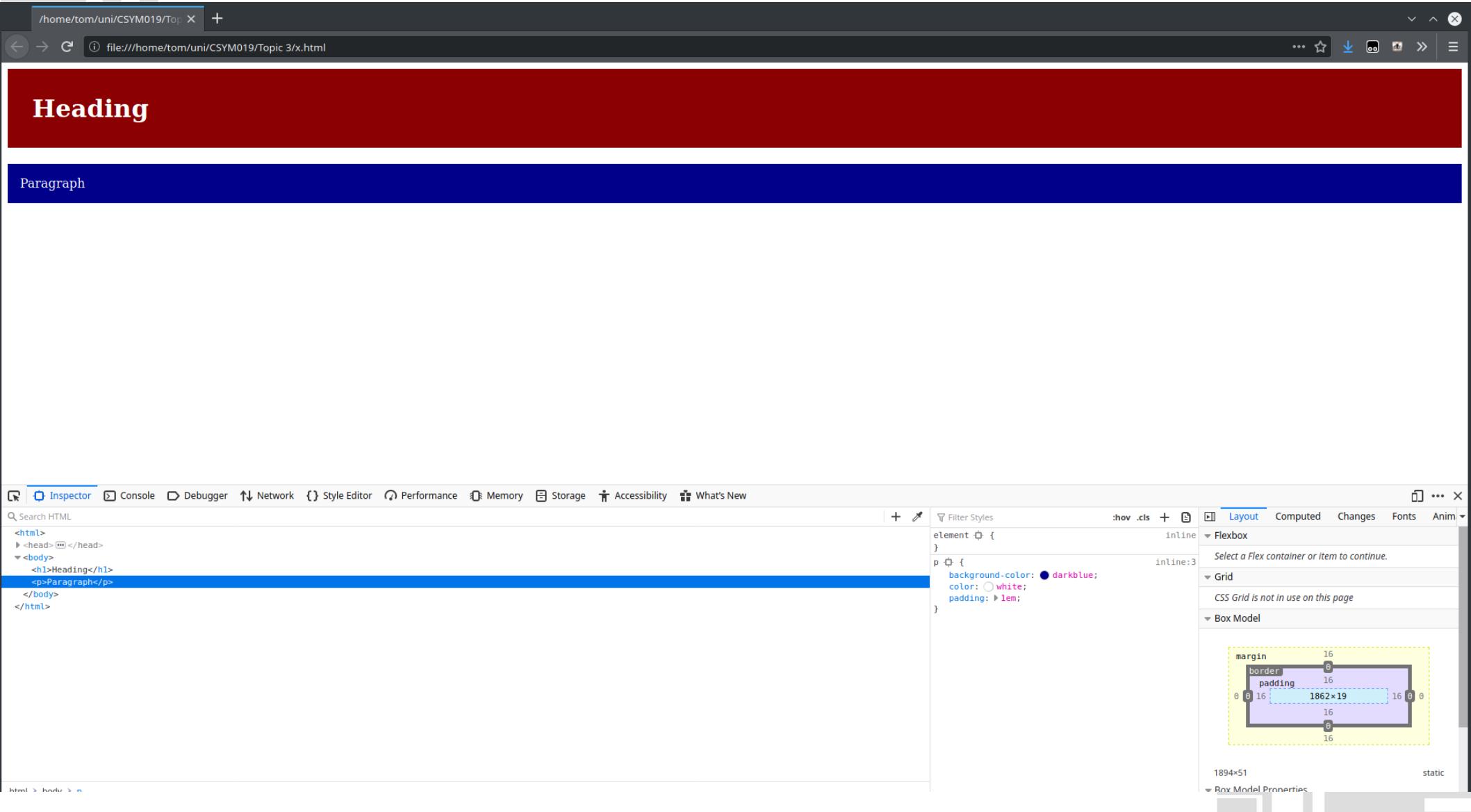
- There are several issues with this
  - 1. Legality. You cannot legally convert a font owned by someone else like Microsoft and distribute it on your website.
  - 2. Performance. If every website did this, then users would end up downloading the same fonts over and over again for each website they visit.

# Google fonts

- Google provides a service for hosting royalty free fonts.
- You can embed fonts from google and any font on this service is freely available

# 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 2015

# Add colours to see what is what

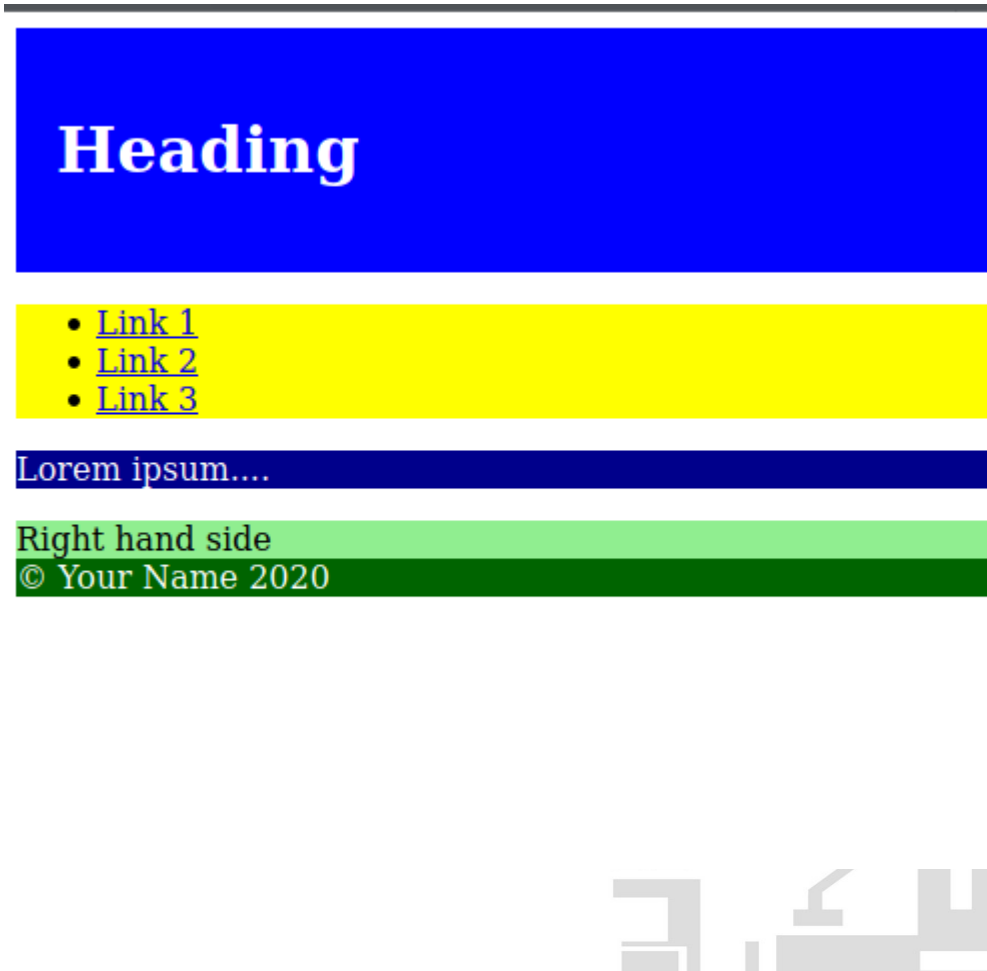
```css
header {
 background-color: blue;
 color: white;
 padding: 20px;
}


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;
}
```

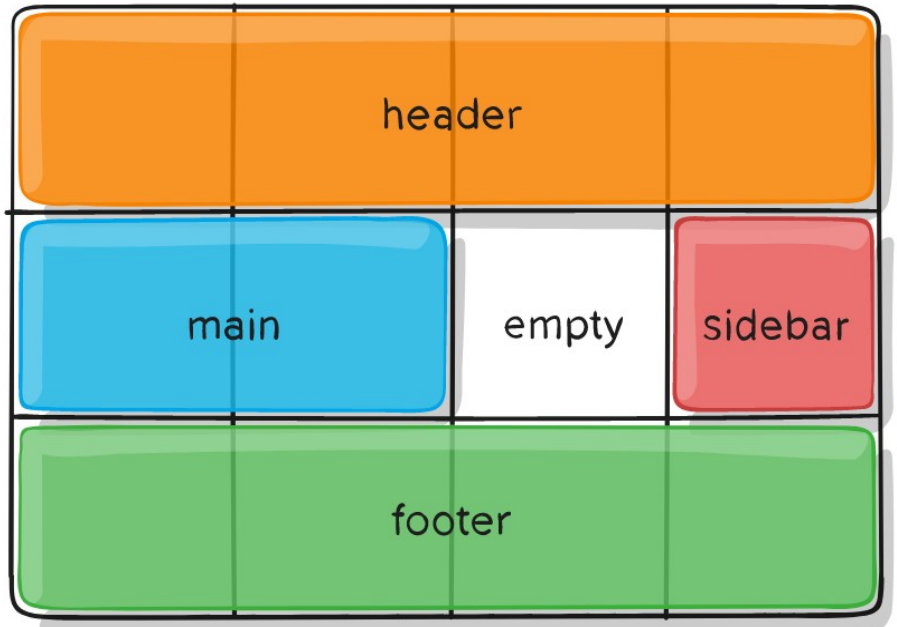
# columns

- 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 c
- 3 columns: 20% 60% 20%:

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

# grid

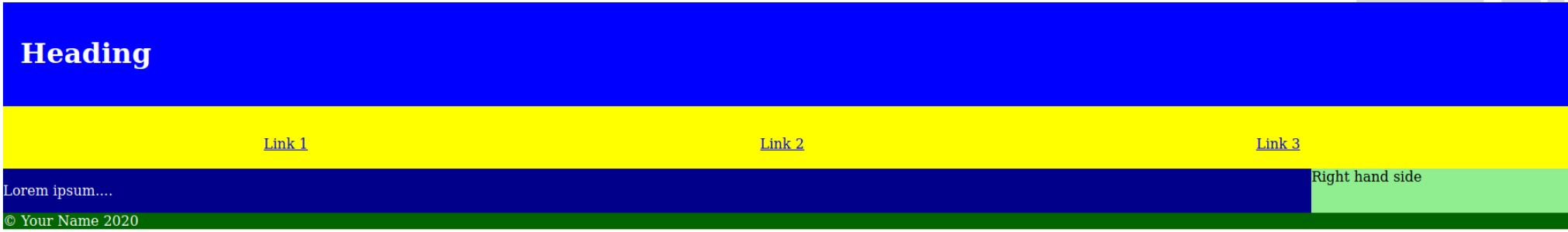
- This will create a grid with as many rows as necessary and 3 columns.
- You then specify which elements occupy which parts of the grid



```
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

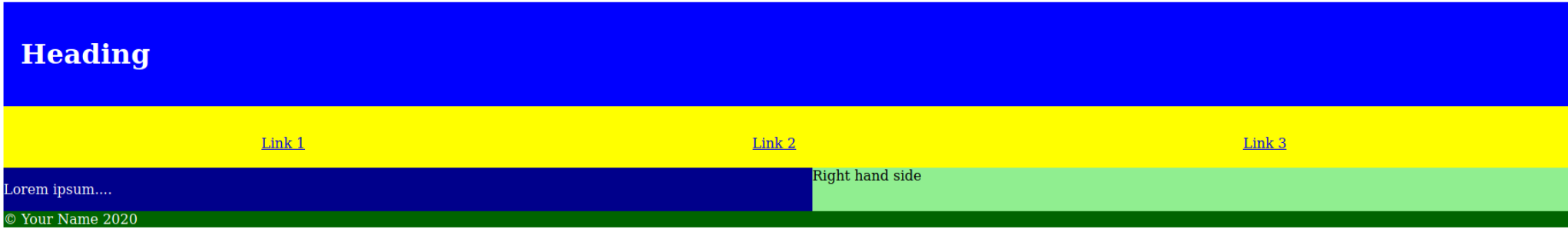
- 1. Follow the slides to create the 3 column layout
  - Make sure you use the correct HTML elements (<header>, <main>, <nav>, <footer> and <aside>)
- 2. Once you've got the basic layout, tweak it to meet the following designs:
- 1:



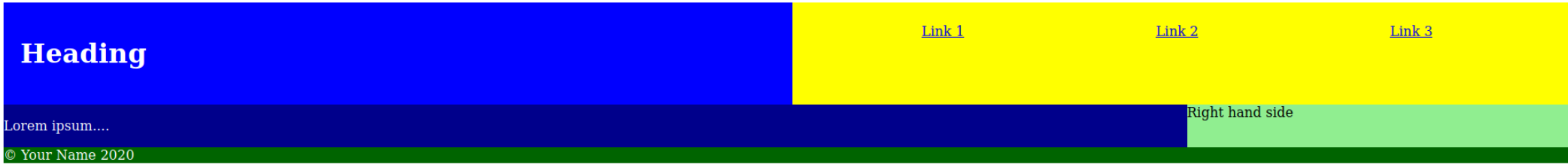
- 2:


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

- 3:


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

- 4:


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
  - Use a selection of colours from colour lovers and fonts from google fonts to see how nice you can make your page look

# Create a website

- Choose a topic that you are interested (anything: your favourite film, game, tv show, sport, etc) and create a web page about it
- Choose nice looking fonts and colours and experiment with making the page look good