CSY2028 Web Programming

Tom Butler
thomas.butler@northampton.ac.uk

Topic 1 – Introduction/Recap

- About me
- Module Overview
- Module Objectives
- HTML & CSS recap

About me

- Graduated from UoN in 2007
- Worked as a software developer since
- Been teaching here 2 years part time
- Working on my PhD into software development best practices
- Write infrequently online:
  - My blog http://r.je
  - Sitepoint.com

Module Overview

- Module structure
  - 20 Credits
  - 22 Weeks
    - 2 hour session once a week
    - Ensure you come to your timetabled slot
  - Assignments
    - One assignment due in January (50%)
    - One assignment due in April (50%)

Module Objectives

- Develop an understanding of web programming including
  - Development tools and techniques used in industry
  - Best practices
  - PHP Programming

- You should already have the following:
  - Basic programming skills
  - Basic website building skills (HTML)
  - Basic database skills (MySQL)

How do the sessions work?

- 2 hour practical session
  - Notes will be broken up into **topics**
  - **Each topic may take one or more lectures**
  - Exercises will be mixed in with the notes
  - We will go at the speed of the class, if something takes longer than expected we'll move some of the lecture to the following week
  - Feel free to ask questions
  - Slides will be on NILE
  - Solutions will be on NILE on Saturday the following week

How do the sessions work?

Exercises

- Sometimes exercises will be flagged with a difficulty
- Everyone must do the grade D-C exercises
- Others are optional and may be a lot more difficult
- If there is no grade associated with an exercise, everyone should do it

– The exercises may require more time than we have in the seminar, be prepared to work outside the class

– You may work in small groups (**Except for assignments!)**

– Ask questions

Topics Covered

1) Introduction + Basic HTML/CSS recap
2) HTML5 + Advanced CSS tips and tricks
3) Setting up a development environment for web programming
4) PHP - Introduction
5) PHP – Functions and Control Structures
6) PHP – Arrays and data structures
7) PHP – Handling User Input
8) Object-Oriented PHP Code
9) MySQL – Introduction + Connecting with PHP
10)MySQL – Manipulating databases with PHP
11)PHP, MySQL and Database Abstraction
12) PHP- Maintaining state across requests + user log ins

Topics Covered

13) Revision session
14) Object Oriented PHP part 2
15) Object Oriented PHP part 3
16) URL rewriting with mod_rewrite, URL Routing, & Single point of entry
17) Separation of concerns + HTML Templates
18) Complex HTML forms + automation
19) MVC (Model-View-Controller)
20) Building reusable components
21) Deploying your website with Git
22) Javascript – jQuery
23) Javascript – Ajax
24) PHP – Unit Testing/Test Driven Development

Module Focus

- Learn how to build websites using PHP and MySQL
- Object-Oriented Programming in PHP
- Best Practices- Why should you structure your code in a certain way?
- Making code reusable, removing duplication of effort

Today's Lecture

- Today's lecture will cover:
  - HTML
  - CSS
  - Creating a basic web page

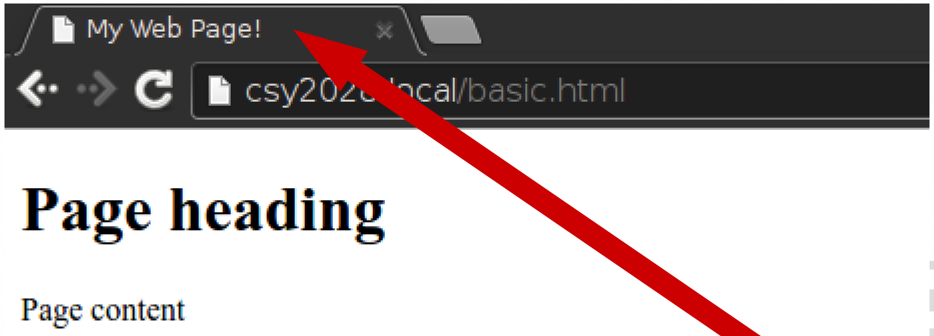
HTML

- Basic HTML rules:
  - All HTML files should start with **<!DOCTYPE html>** (This is the only doctype you should use, all others are now very out of date!)
  - All HTML files should have a **<html>** tag which wraps everything else
  - All HTML should have a **<body>** tag which wraps all the content
  - Most HTML files should have a **<head>** tag which contains *metadata*

HTML

- A basic HTML file looks like this:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
  </head>

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



Page heading

Page content

Nothing in <head>
appears on the page, but the info is
used elsewhere

HTML

- HTML is forgiving
  - You can make mistakes and the browser will try to fix it
- But you shouldn't rely on this behaviour!
  - Try to always write valid HTML
  - You can (and should!) check your HTML for errors using the W3C HTML Validator http://validator.w3.org/
    - Alternatively, most editors (e.g. Sublime) have plugins to do this inside the program which makes it trivial to do
  - If your page isn't displaying as you expect, it's probably got an error and the validator will help you find out how to fix it by telling you where to look (e.g. tag name or line number)

Useful HTML tags

- Link to another page or website
  - <a href="http://www.google.com">Click here to go to google</a>
- Display an image:
  - <img src="logo.gif" alt="Logo" />
    - Always supply an alt attribute, this is used by screen readers and when the image cannot be loaded
- A paragraph of text
  - <p>text</p>
- Headings of various levels (h1 is the top "biggest" header)
  - <h1>Heading</h1>, <h2>Heading 2</h2>,….. <h6>Heading 6</h6>

CSS

- CSS is a separate language which is used to describe how a HTML file looks when it's opened in a browser
- CSS should be in its own file (or multiple files) rather than in the HTML file

Inline vs External Styles

- You can store CSS rules inside the HTML file
- Although this is discouraged
- You should store CSS in its own file:
  - This allows you to use the same CSS rules on multiple HTML files
  - OR present some HTML in more than one way using different CSS files

CSS

- External stylesheets are structured in the following way:

```
selector {
        rule-name: value;
        rule-name: value;
}
```

- The selector is a way of 'targeting' one or more HTML elements on the page and applying properties to it
- Each stylesheet can contain one or more selectors with rules

CSS Rules

- Each rule is used to declare the way the element should be presented. For example

```
h1 {
        color: red;
}
```

- Targets any H1 element and sets the colour of the text to red
- *Notice the American spelling of colour without a u!*

External Stylsheets

- To attach a css file to a HTML file, save the file with a .css extension in the same directory as the HTML file and you can reference it in the HTML using a <link> tag inside the <head> tag

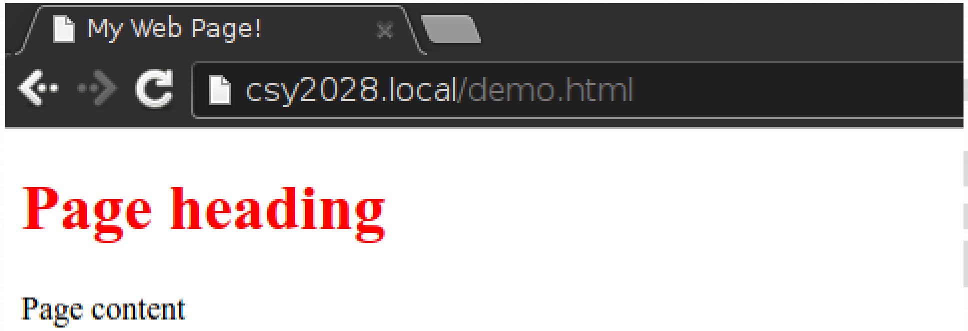
```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

CSS

- Now, when you open up the page in a browser it will pull in both the HTML and CSS then apply the CSS rules to the HTML

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

```css
h1 {
        color: red;
}
```



Page heading

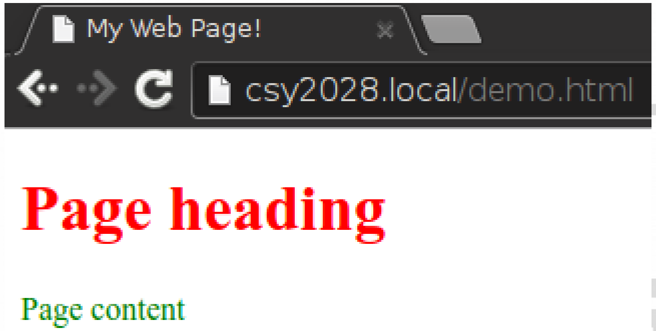
Page content

CSS

- You can add as many rules to a CSS file as you like.
- Each rule has a selector, an opening brace, some properties and a closing brace

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

```
h1 {
        color: red;
}
p {
        color: green;
}
```



Page heading

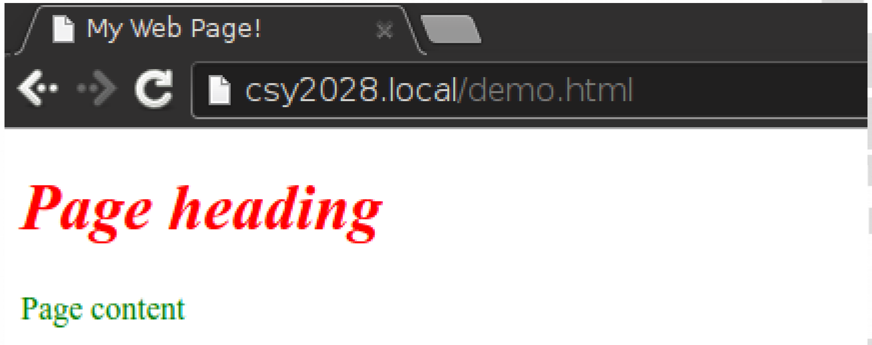
Page content

CSS

- Each rule can have one or more properties. For example:
- To change the font to italic you can use the property *font-style* with the value *italic*

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

```css
h1 {
       color: red;
       font-style: italic;
}
p {
       color: green;
}
```



Page heading

Page content

CSS Selectors

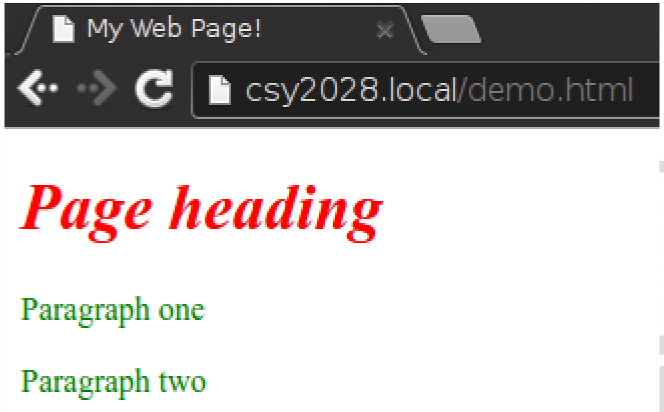
- There are three main css selectors
  - Element name
  - Class name
  - ID

Element Name selector

- The element name selector selects all elements with the same tag name

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <h1>Page heading</h1>
    <p>Paragraph one</p>
    <p>Paragraph two</p>
  </body>
</html>
```

```
h1 {
        color: red;
        font-style: italic;
}
p {
        color: green;
}
```



Page heading

Paragraph one

Paragraph two

Notice that both paragraphs are now green

Class name selector

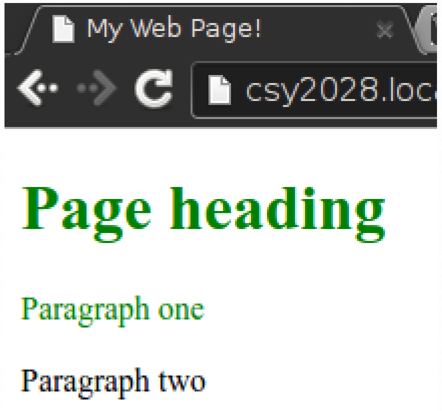
- Any element can be given a *class* this is an HTML attribute like *src* in images or *href* in links
- The class name selector works in the format
  - .name
  - A dot followed by the name of the class you want to match
- This will match any element that has *class="name"*
- Note that the attribute does not include the dot in the HTML!

Class name selector

- The class name selector is a name prefixed with a dot and selects any element with a class attribute set to that value

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <h1 class="myclass">Page heading</h1>
    <p class="myclass">Paragraph one</p>
    <p>Paragraph two</p>
  </body>
</html>
```

```
.myclass {
        color: green;
}
```



Page heading

Paragraph one

Paragraph two

Notice that both elements with the class “myclass” are green

ID Selector

- The ID selector works very similarly to the class name selector
  - Uses a # prefix
  - #myid
  - Matches any element with id="myid" set as an attribute
- Note that the HTML does not contain the # symbol!
- IDs are different to class names. You can only use an ID once: Each ID should apply to a single element on the page

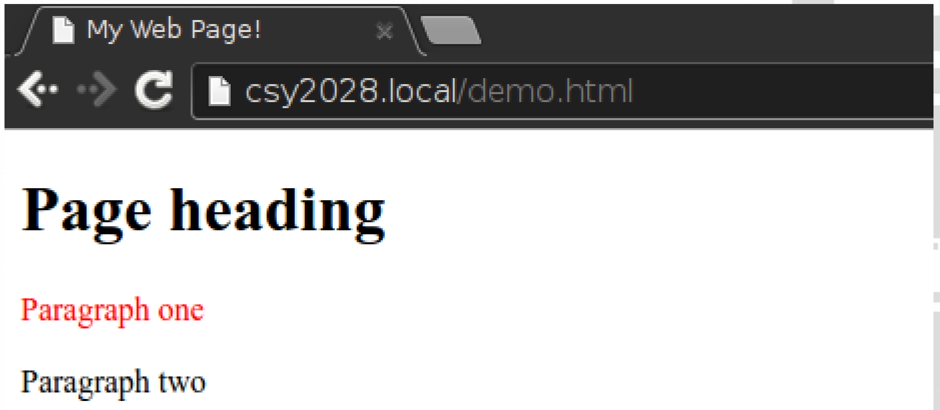
ID selector

- The ID name selector is a name prefixed with a hash and selects the element with that ID

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <h1>Page heading</h1>
    <p id="myid">Paragraph one</p>
    <p>Paragraph two</p>
  </body>
</html>
```

```
#myid {
       color: red;
}
```

You may only have one element with each ID



Page heading

Paragraph one

Paragraph two

Which should I use?

- In a lot of cases you can use a class name, or an element name or an ID to achieve exactly the same result!
- You should avoid IDs because you cannot easily reuse the style (if you want to make two elements red, you need to add two css rules and two IDs!)
- You should usually avoid element selectors as they're very imprecise
  - They will affect ALL elements of that type!
  - Sometimes, this is what you want
  - But if you want a specific style on a single part of the page, you might get unexpected results as the page grows and you add more elements

Combining Selectors

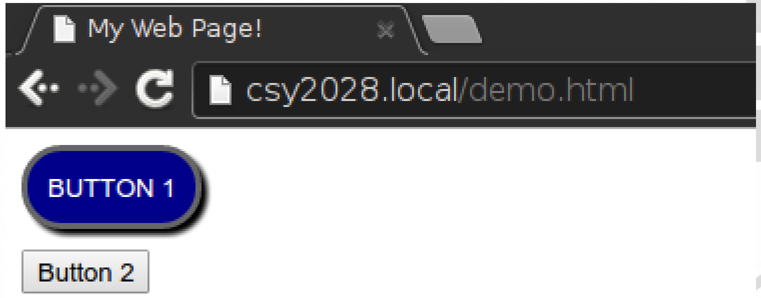
- HTML is a *nested* notation
  - (Some) Elements can exist inside (some) other elements
- You can use css to target nested elements. By combining a selector with a space you can target specific elements in the HTML DOM Tree
  - article h1 {font-weight: bold} – Sets any <h1> element inside an <article> element to bold
  - .myclass span {font-style: italic} – Sets any <span> element inside an element with *class="myclass"* to italic
- This will target elements any *depth!*

Combining selectors

- You can combine selectors to be more specific

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="myform">
      <input type="button" value="Button 1" />
    </div>

    <input type="button" value="Button 2" />
  </body>
</html>
```

```css
.myform input {
  border-radius: 20px;
  border: 3px solid #666;
  box-shadow: 3px 3px 3px #000;
  margin-bottom: 10px;
  background-color: darkblue;
  padding: 10px;
  text-transform: uppercase;
  color: white;
}
```



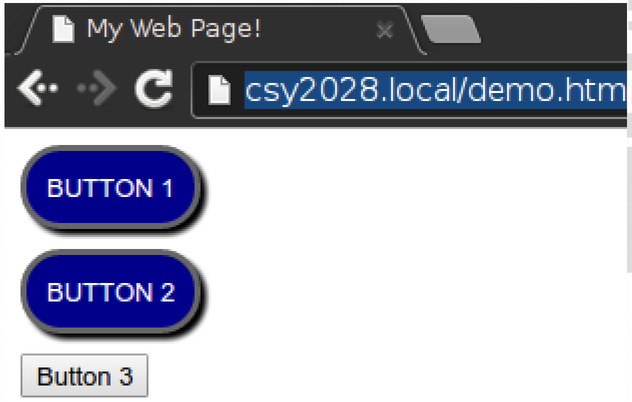
Notice that only the button inside the element with the class *myform* has been styled

Combining selectors

- This will target elements of any *depth*

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="myform">
       <input type="button" value="Button 1" />
      <section>
        <form>
          <input type="button" value="Button 2" />
        </form>
      </section>
    </div>

    <input type="button" value="Button 3" />
  </body>
</html>
```

```css
.myform input {
  border-radius: 20px;
  border: 3px solid #666;
  box-shadow: 3px 3px 3px #000;
  margin-bottom: 10px;
  background-color: darkblue;
  padding: 10px;
  text-transform: uppercase;
  color: white;
}
```



Notice that only the button inside the element with the class *myform*

Combining selecors

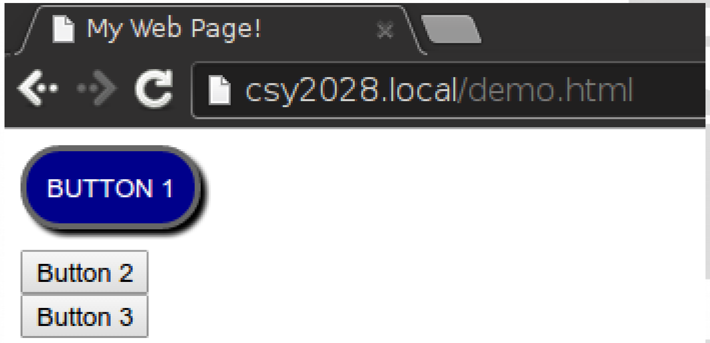
- You can use the *direct decedent* selector to filter to only elements that are only one level down
- The direct decedent selector uses the greater than symbol >

Combining selectors

- This will target elements only one level down

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="myform">
       <input type="button" value="Button 1" />
      <section>
        <form>
          <input type="button" value="Button 2" />
        </form>
      </section>
    </div>

    <input type="button" value="Button 3" />
  </body>
</html>
```

```
.myform > input {
  border-radius: 20px;
  border: 3px solid #666;
  box-shadow: 3px 3px 3px #000;
  margin-bottom: 10px;
  background-color: darkblue;
  padding: 10px;
  text-transform: uppercase;
  color: white;
}
```



Notice that only the button inside the element with the class *myform*

Combining selectors

- You can combine selectors into very complex expressions e.g.:
- `#myelement > .myclass article section header h1`
- However, this is generally discouraged
- We call this *tightly coupled* to the HTML: A slight change to the HTML will stop the h1 tag being styled
- As a rule of thumb you shouldn't need more than three levels of selector

Layouts

- CSS is used for describing how elements look on the screen:
  - Background colours, font colours, font sizes, font faces, spacing (paddings/margins)
- It's other use is describing how the page is laid out: *where* on the screen the element should appear and how large it is

Layouts

- Don't use tables for layouts!
- Back in 1999 when browser support for CSS was limited, tables were the best available choice for positioning elements on the screen
- Using tables for this purpose has been labelled *bad practice* since the early 2000s!
- Don't do it!
- I will mark you down for doing this!
- Further reading:
  - https://www.hotdesign.com/seybold/everything.html
  - http://phrogz.net/css/WhyTablesAreBadForLayout.html
  - http://webdesign.about.com/od/layout/a/aa111102a.htm
  - http://www.htmlgoodies.com/beyond/css/article.php/3642151

Layout properties

- Knowing how these 3 properties work will let you build almost any layout you want
  - Display
  - Float
  - Overflow

Display

- There are two main properties:
  - Inline
  - Block
- Inline elements will display on the same line as other inline elements
- Block elements will fill all the available width

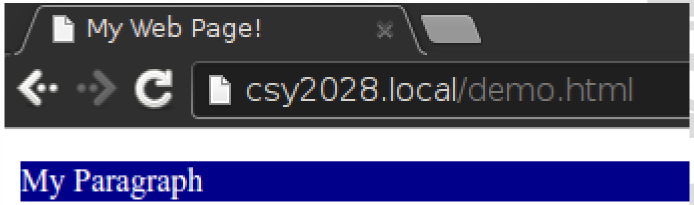
Block vs inline

- Every element has a default display property of block or inline.
- Example inline elements:
  - <img>
  - <span>
  - <input>
- Example block elements:
  - <p>
  - <div>
  - <h1>
- These are not fixed and can be easily overridden with CSS

Block elements

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

```
p {
  background-color: darkblue;
  color: white;
  display: block;
}
```



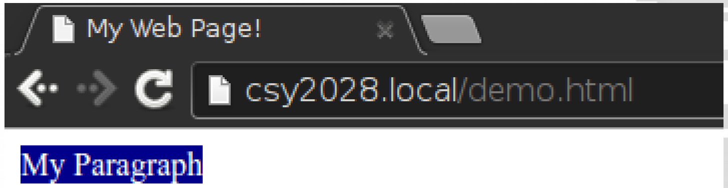
My Paragraph

Notice that the blue background goes all the way across the page

Inline elements

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

```
p {
  background-color: darkblue;
  color: white;
  display: inline;
}
```



My Paragraph

Notice that the blue background only extends to the width of the text

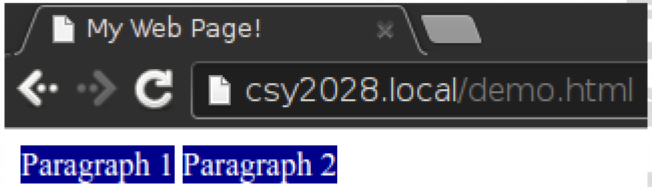
Block vs inline

- Block and inline affects how elements interact with each other
- Block elements will always be on their own line
- Inline elements will appear on the same line if there is enough space

Inline elements

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: inline;
}
```



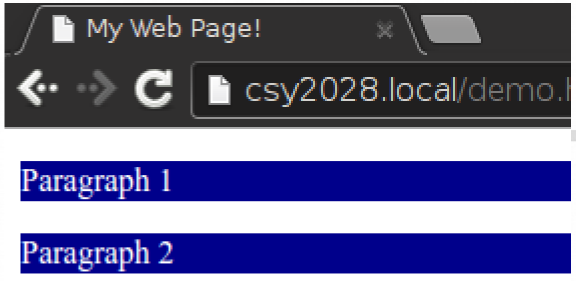
Paragraph 1 Paragraph 2

The elements appear on the same line

Block elements

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
}
```



Paragraph 1

Paragraph 2

The elements appear on the same line

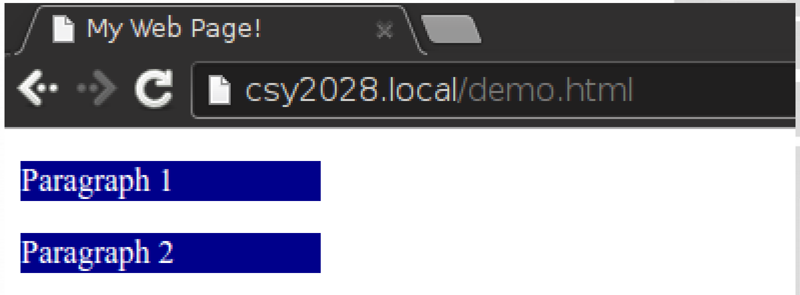
Block vs inline

- Only block elements can have a width and height!
- Setting a width and a height on inline elements will have no effect

Block elements

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
}
```



Paragraph 1

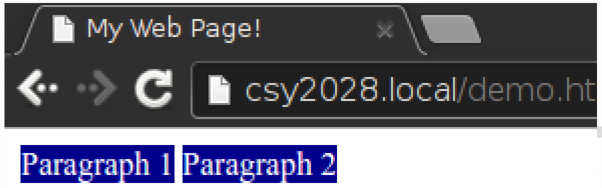
Paragraph 2

The elements appear on their own lines but now don't span the entire width

Block elements

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: inline;
  width: 150px;
}
```



Paragraph 1 Paragraph 2

Width has no effect, they both stretch only as wide as the content

Float

- How do you put elements with a fixed with/height on the same line?
- *Float* can be applied to *block* elements and have them appear on the same line but with fixed dimensions
- Float has three options:
  - None (default, no float)
  - Left
  - Right

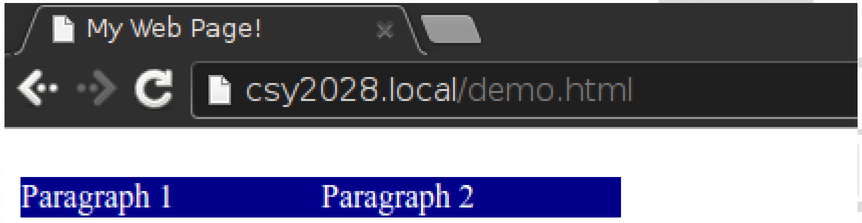
Float: left

- Float left will have the affect of stacking block elements left to right. When elements get too wide for the browser they will wrap to the next line

Float: left

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: left;
}
```



Paragraph 1 Paragraph 2

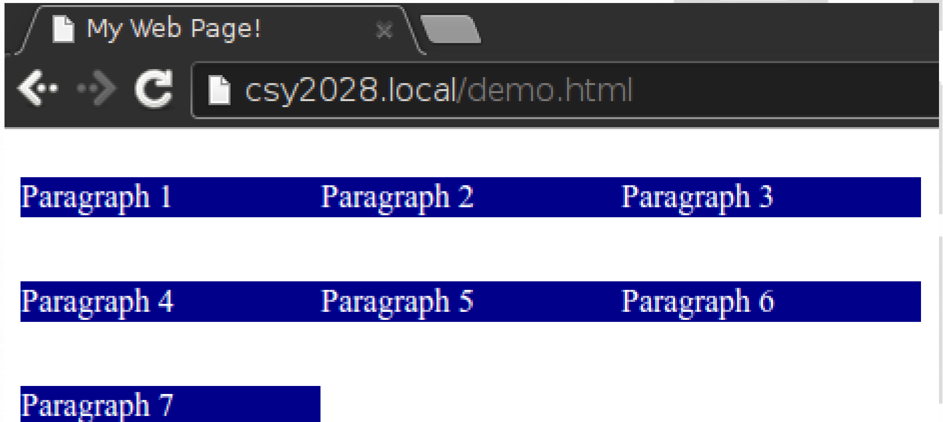
Float: left

- As more elements are added, they will be stacked left to right until there is no more room on the line

Float: left

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
     <p>Paragraph 6</p>
     <p>Paragraph 7</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: left;
}
```



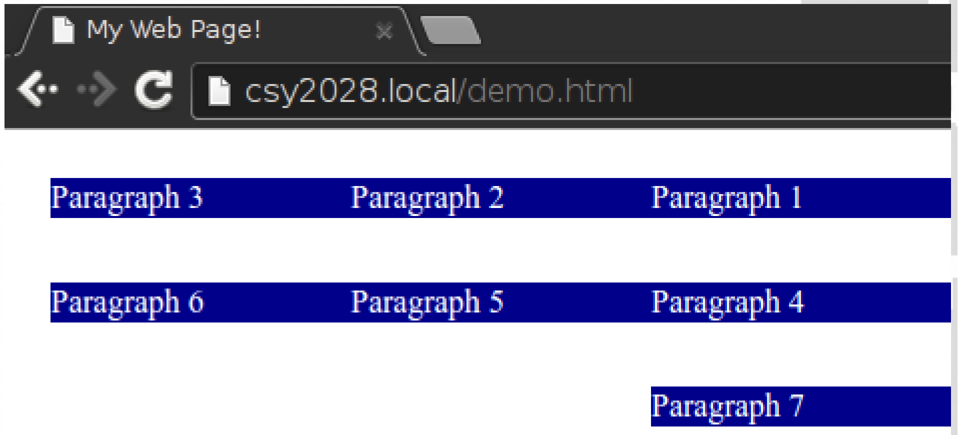
```
Paragraph 1    Paragraph 2    Paragraph 3

Paragraph 4    Paragraph 5    Paragraph 6

Paragraph 7
```

Float: right

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
     <p>Paragraph 6</p>
     <p>Paragraph 7</p>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: right;
}
```

Float: right stacks the elements from right to left



```
Paragraph 3     Paragraph 2     Paragraph 1

Paragraph 6     Paragraph 5     Paragraph 4

                                Paragraph 7
```

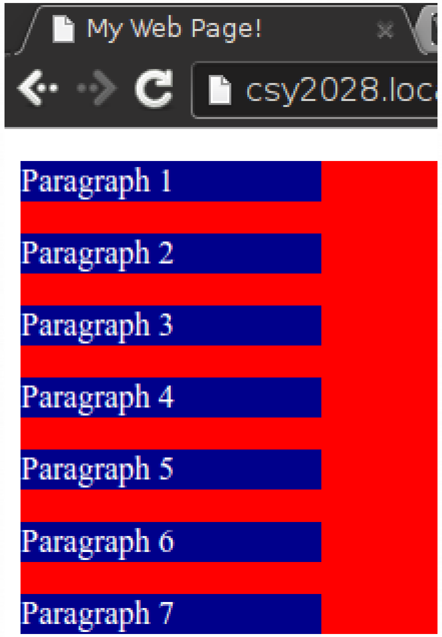
Floats

- Floats have some unexpected results
- Usually when an element contains another, the background will be visible

Float

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="red">
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
     <p>Paragraph 6</p>
     <p>Paragraph 7</p>
    </div>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;

}

.red {
  background-color: red;
}
```



Paragraph 1

Paragraph 2

Paragraph 3

Paragraph 4

Paragraph 5

Paragraph 6

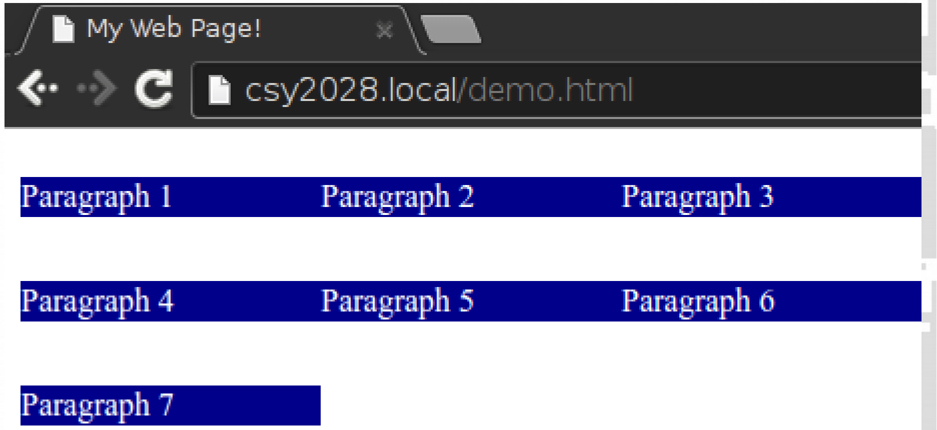
Paragraph 7

As you expect, the paragraphs sit in a container with a red background

Float

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="red">
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
     <p>Paragraph 6</p>
     <p>Paragraph 7</p>
    </div>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: left;

}

.red {
  background-color: red;
}
```



Paragraph 1 Paragraph 2 Paragraph 3

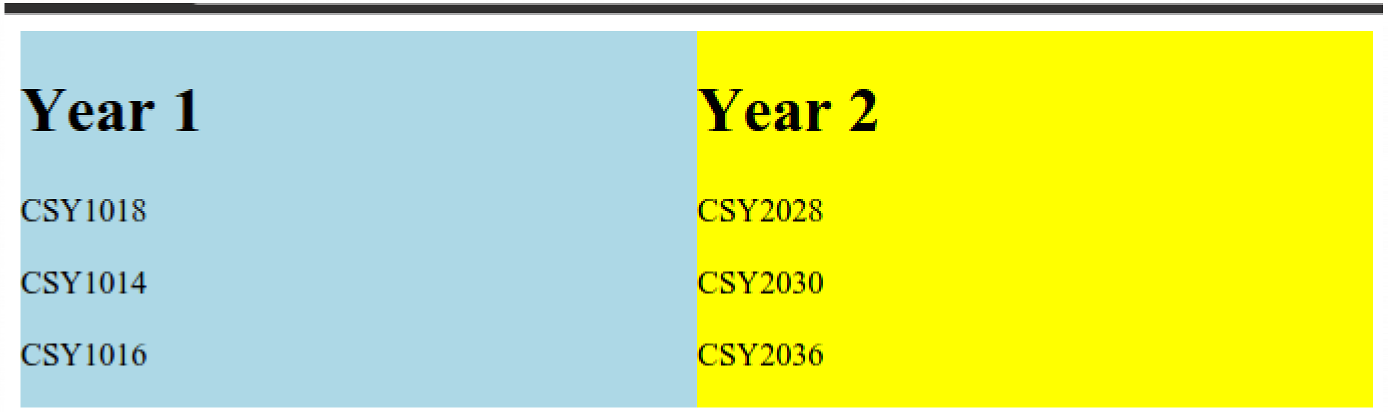
Paragraph 4 Paragraph 5 Paragraph 6

Paragraph 7

But when float: left is added,
The background disappears!

Exercise 1

Create a web page with two columns with a width of 50% each. In column 1 put all your first year modules, in column 2 put all your second year modules. You should give each column a background colour. It should look something like this:



Year 1	Year 2
CSY1018	CSY2028
CSY1014	CSY2030
CSY1016	CSY2036

Hint: You can use 50% widths on the Elements!

Exercise 1 - Solution

```html
<!doctype html>
<html>
    <head>
        <title>Exercise 1</title>
        <link rel="stylesheet" href="ex1.css" />
    </head>

    <body>
        <div class="left">
            <h2>Year 1</h2>
            <p>CSY1018</p>
            <p>CSY1014</p>
            <p>CSY1016</p>
        </div>

        <div class="right">
            <h2>Year 2</h2>
            <p>CSY2028</p>
            <p>CSY2030</p>
            <p>CSY2036</p>
        </div>

    </body>
</html>
```

```css
.left {
     float: left;
     width: 50%;
     background-color: lightblue;
}
.right {
     float: left;
     width: 50%;
     background-color: yellow;
}
```

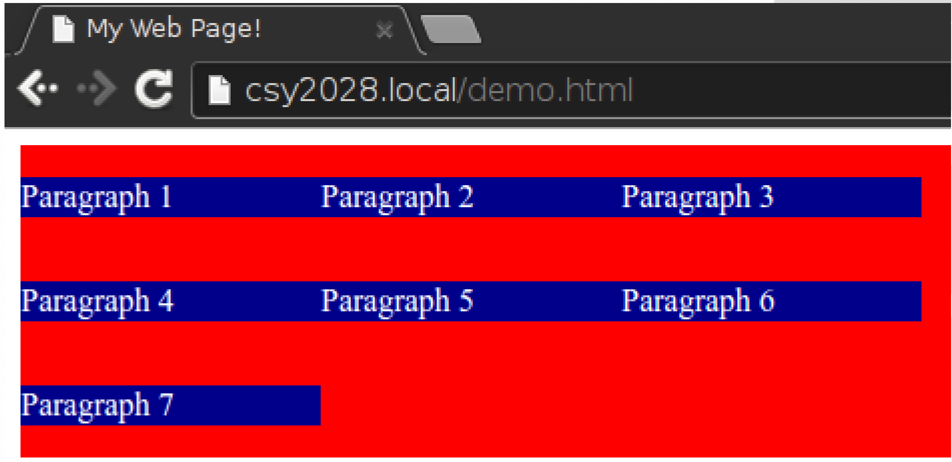
Overflow

- The overflow property can be used to make the background appear
- Overflow sets whether the element should be stretched to fill floated (or otherwise moved) child elements
- By adding overflow: auto to the .red element the background will appear

Float

```html
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="red">
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
     <p>Paragraph 6</p>
     <p>Paragraph 7</p>
    </div>
  </body>
</html>
```

```css
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: left;
}

.red {
  background-color: red;
  overflow: auto;
}
```



```
Paragraph 1        Paragraph 2        Paragraph 3

Paragraph 4        Paragraph 5        Paragraph 6

Paragraph 7
```

By adding overflow: auto,
The background appears

Clear: both

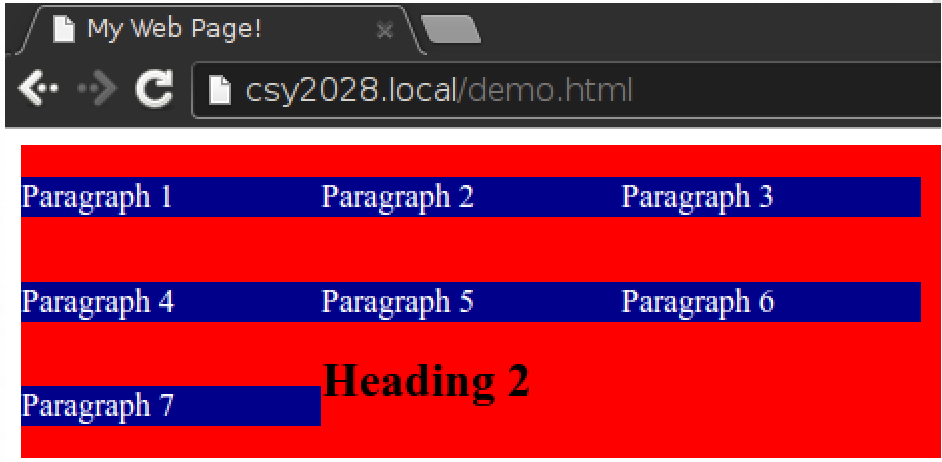
- When an element without a float is next to an element with a float, it will be floated anyway!
- To stop this, you can apply the style clear: both

Clear: both

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="red">
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
     <p>Paragraph 6</p>
     <p>Paragraph 7</p>
     <h2>Heading 2</h2>
    </div>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: left;
}

.red {
  background-color: red;
  overflow: auto;
}
```

The H2 is floated even though it doesn't have float set

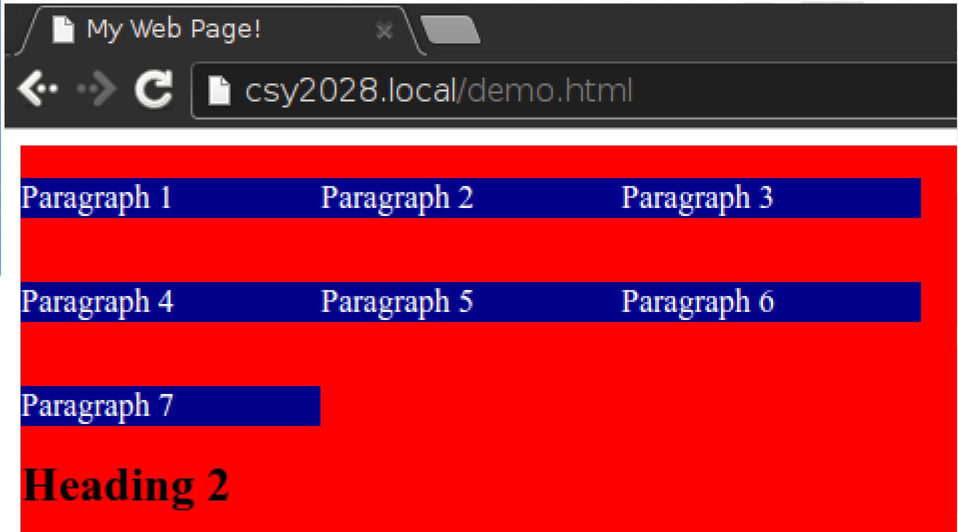


Paragraph 1 Paragraph 2 Paragraph 3
Paragraph 4 Paragraph 5 Paragraph 6
Paragraph 7 **Heading 2**

Clear: both

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Web Page!</title>
    <link rel="stylesheet" href="demo.css" />
  </head>
  <body>
    <div class="red">
     <p>Paragraph 1</p>
     <p>Paragraph 2</p>
     <p>Paragraph 3</p>
     <p>Paragraph 4</p>
     <p>Paragraph 5</p>
     <p>Paragraph 6</p>
     <p>Paragraph 7</p>
     <h2>Heading 2</h2>
    </div>
  </body>
</html>
```

```
p {
  background-color: darkblue;
  color: white;
  display: block;
  width: 150px;
  float: left;
}
.red {
  background-color: red;
  overflow: auto;
}
h2 {
  clear: both;
}
```



Paragraph 1 Paragraph 2 Paragraph 3

Paragraph 4 Paragraph 5 Paragraph 6

Paragraph 7

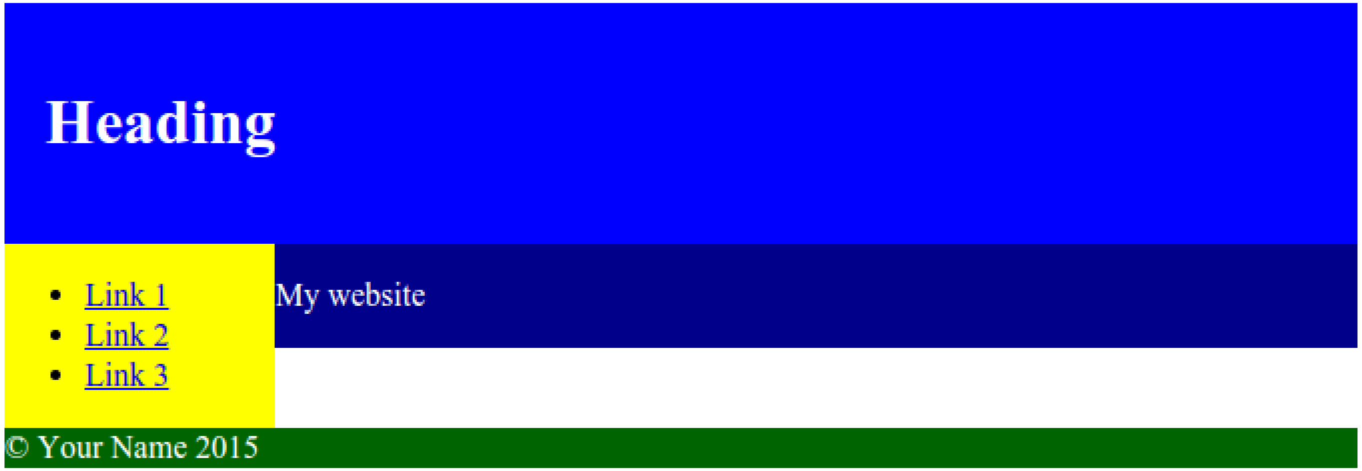
Heading 2

By applying clear: both, the heading will not float

Exercise 2

1) Download ex2.html from NILE and create **demo.css** to store your CSS

2) **Without modifying ex2.html**, use CSS to make it appear as the supplied layout. You can choose your own colours but the layout (heading at top, footer at bottom, navigation on the left and main content in the middle) should be like the screenshot below

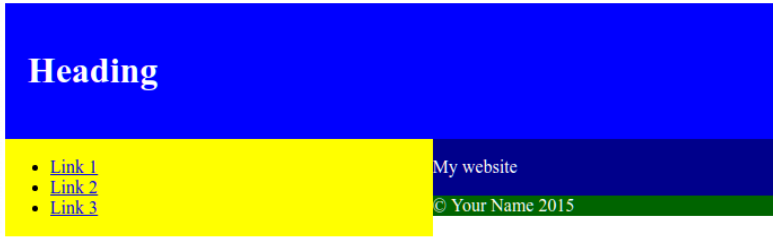


Hint: Navigation is 20% wide

Exercise 2 - Solution

```
header {
    display: block;
    background-color: blue;
    color: white;
    padding: 20px;
}
nav {
    float: left;
    width: 20%;
    background-color: yellow;
}
main {
    float: left;
    width: 80%;
    background-color: darkblue;
    color: white;
}
footer {
    clear: both;
    background-color: darkgreen;
    color: white;
}
```

Note clear: both, without it
The footer appears in the wrong place!

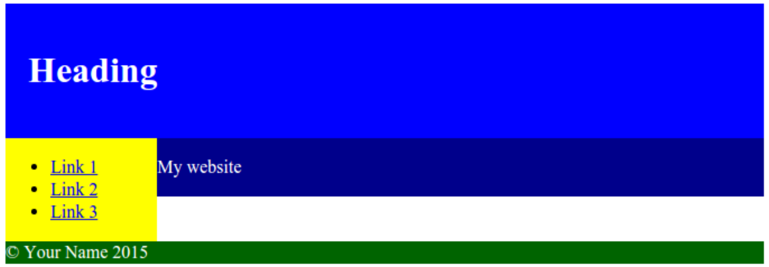


Heading

- Link 1
- Link 2
- Link 3

My website

© Your Name 2015



Heading

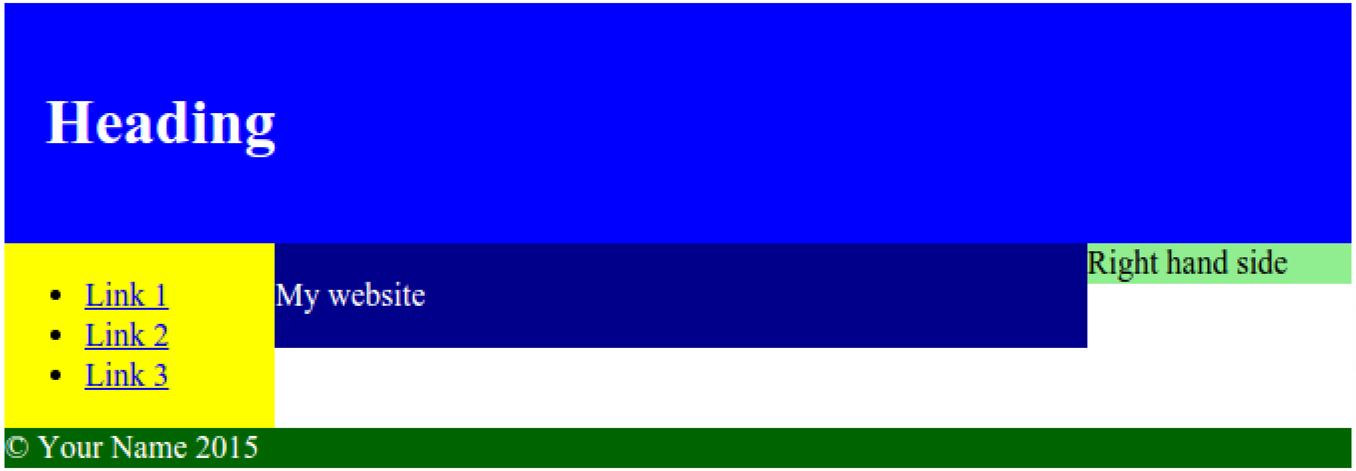
- Link 1
- Link 2
- Link 3

My website

© Your Name 2015

Exercise 3

1) Change ex2.html to include a <div> tag that represents a right hand column. Then adjust the CSS to make it appear like below:



Heading

- Link 1
- Link 2
- Link 3

My website

Right hand side

© Your Name 2015

Hint: Right hand column is 20% wide

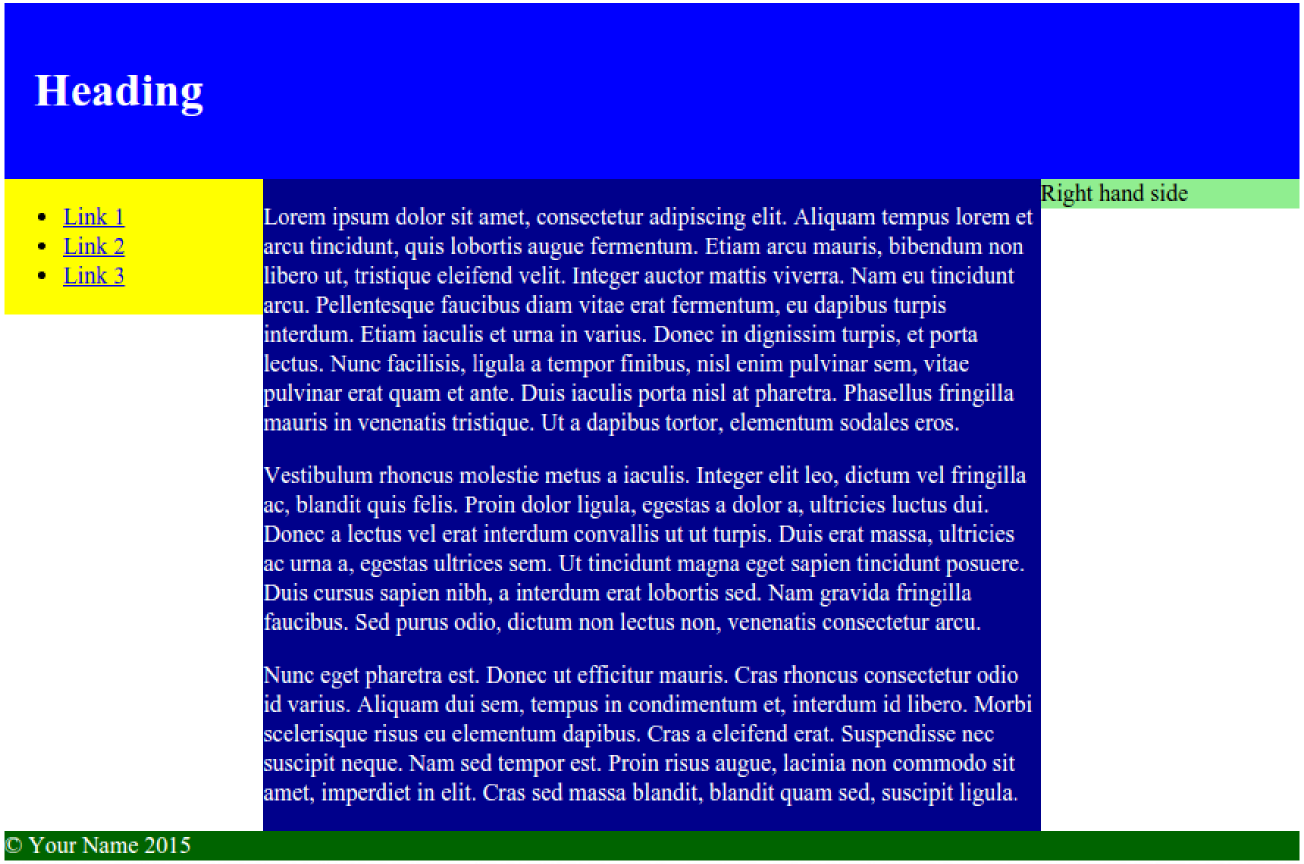
Exercise 3 - Solution

```
...

    </nav>
    <main>
      <p>My website</p>
    </main>

    <div>
      right hand side
    </div>

    <footer>
      &copy; Your Name 2015
    </footer>
  </body>
</html>
```

```
...
main {
      float: left;
      width: 60%;
      background-color: darkblue;
      color: white;
}

div {
      float: left;
      Width: 20%;
      background-color: lightgreen;
}
```

Web Layouts

- One problem with this is that when you add content the columns don't all stretch down to fill the available height:



Heading

- Link 1
- Link 2
- Link 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam tempus lorem et arcu tincidunt, quis lobortis augue fermentum. Etiam arcu mauris, bibendum non libero ut, tristique eleifend velit. Integer auctor mattis viverra. Nam eu tincidunt arcu. Pellentesque faucibus diam vitae erat fermentum, eu dapibus turpis interdum. Etiam iaculis et urna in varius. Donec in dignissim turpis, et porta lectus. Nunc facilisis, ligula a tempor finibus, nisl enim pulvinar sem, vitae pulvinar erat quam et ante. Duis iaculis porta nisl at pharetra. Phasellus fringilla mauris in venenatis tristique. Ut a dapibus tortor, elementum sodales eros.

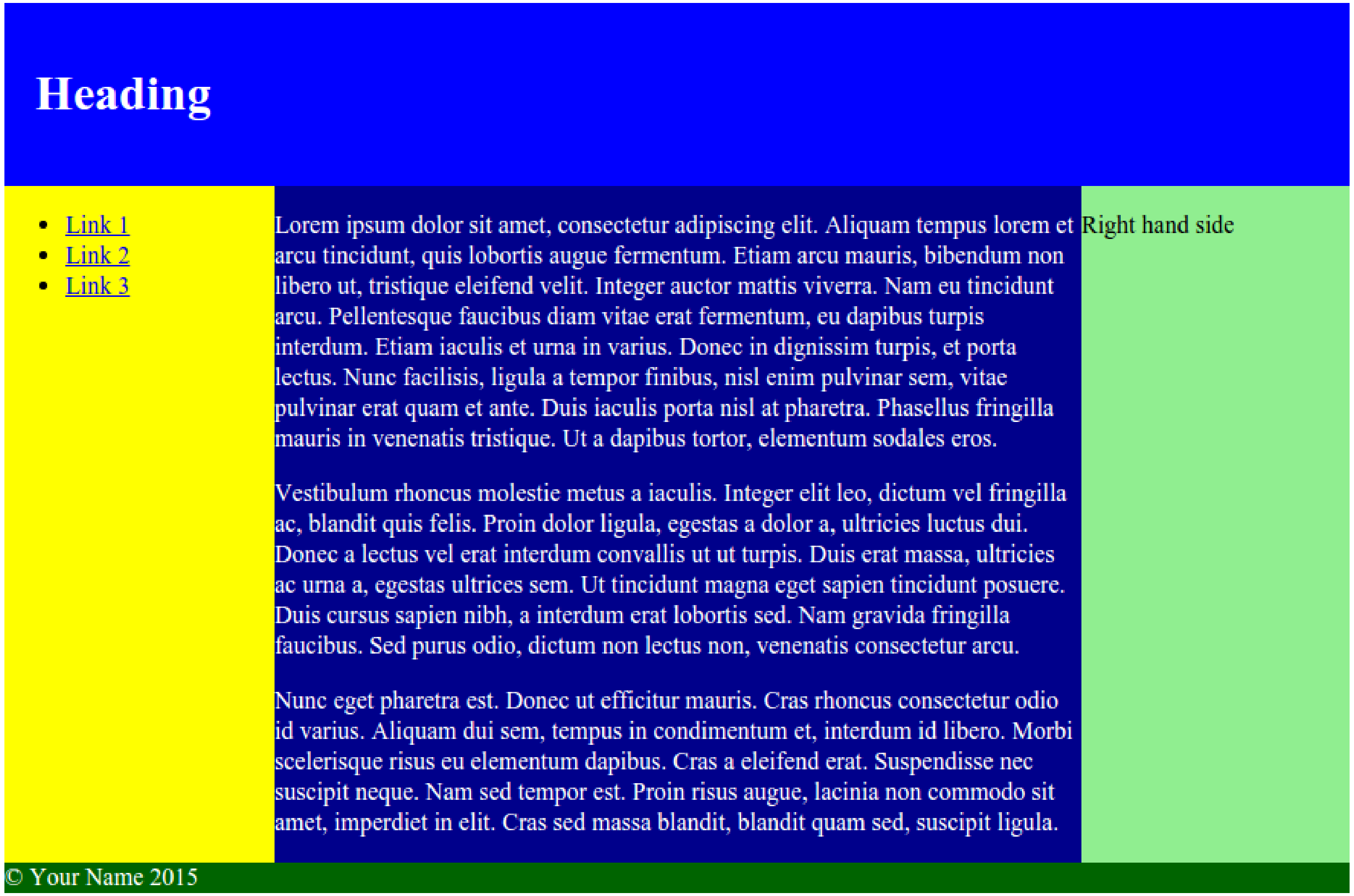
Vestibulum rhoncus molestie metus a iaculis. Integer elit leo, dictum vel fringilla ac, blandit quis felis. Proin dolor ligula, egestas a dolor a, ultricies luctus dui. Donec a lectus vel erat interdum convallis ut ut turpis. Duis erat massa, ultricies ac urna a, egestas ultrices sem. Ut tincidunt magna eget sapien tincidunt posuere. Duis cursus sapien nibh, a interdum erat lobortis sed. Nam gravida fringilla faucibus. Sed purus odio, dictum non lectus non, venenatis consectetur arcu.

Nunc eget pharetra est. Donec ut efficitur mauris. Cras rhoncus consectetur odio id varius. Aliquam dui sem, tempus in condimentum et, interdum id libero. Morbi scelerisque risus eu elementum dapibus. Cras a eleifend erat. Suspendisse nec suscipit neque. Nam sed tempor est. Proin risus augue, lacinia non commodo sit amet, imperdiet in elit. Cras sed massa blandit, blandit quam sed, suscipit ligula.

Right hand side

© Your Name 2015

Web Layouts



Heading

- Link 1
- Link 2
- Link 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam tempus lorem et arcu tincidunt, quis lobortis augue fermentum. Etiam arcu mauris, bibendum non libero ut, tristique eleifend velit. Integer auctor mattis viverra. Nam eu tincidunt arcu. Pellentesque faucibus diam vitae erat fermentum, eu dapibus turpis interdum. Etiam iaculis et urna in varius. Donec in dignissim turpis, et porta lectus. Nunc facilisis, ligula a tempor finibus, nisl enim pulvinar sem, vitae pulvinar erat quam et ante. Duis iaculis porta nisl at pharetra. Phasellus fringilla mauris in venenatis tristique. Ut a dapibus tortor, elementum sodales eros.

Vestibulum rhoncus molestie metus a iaculis. Integer elit leo, dictum vel fringilla ac, blandit quis felis. Proin dolor ligula, egestas a dolor a, ultricies luctus dui. Donec a lectus vel erat interdum convallis ut ut turpis. Duis erat massa, ultricies ac urna a, egestas ultrices sem. Ut tincidunt magna eget sapien tincidunt posuere. Duis cursus sapien nibh, a interdum erat lobortis sed. Nam gravida fringilla faucibus. Sed purus odio, dictum non lectus non, venenatis consectetur arcu.

Nunc eget pharetra est. Donec ut efficitur mauris. Cras rhoncus consectetur odio id varius. Aliquam dui sem, tempus in condimentum et, interdum id libero. Morbi scelerisque risus eu elementum dapibus. Cras a eleifend erat. Suspendisse nec suscipit neque. Nam sed tempor est. Proin risus augue, lacinia non commodo sit amet, imperdiet in elit. Cras sed massa blandit, blandit quam sed, suscipit ligula.

Right hand side

© Your Name 2015

Web Layouts

- This is a problem with float in CSS
- Although float is useful, it cannot be used to solve this problem
- There are several difficult/awkward solutions:
  - Fixed height on each element
  - Nesting each element (div inside nav, nav inside main, etc)
  - Background images

Display: table

- This can be achieved using *display: table* and *display: table-cell*
- You must have a container element with *display: table*
- Then child elements can be given *display: table-cell*
- This will use the browser's table rendering functionality to display the elements
- Note: Although table elements for layouts are bad practice, display: table is not!

Exercise 4

- 1) Add a container div around all 3 columns such as <div class=”.content”>
- 2) Use display: table and display: table-cell to make the columns stretch down the screen
- Hint: You will need to remove the float CSS properties!

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

    <div>
      Right hand side
    </div>
    </div>

    <footer>
      &copy; Your Name 2015
    </footer>
  </body>
</html>
```

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

.content {
 display: table;
 Width: 100%;
}

nav {
 display: table-cell;
 width: 20%;
 background-color: yellow;
}

main {
 display: table-cell;
 Width:60%;
 background-color: darkblue;
 color: white;
}

div {
 display: table-cell;
 Width: 20%;
 background-color: lightgreen;
}

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

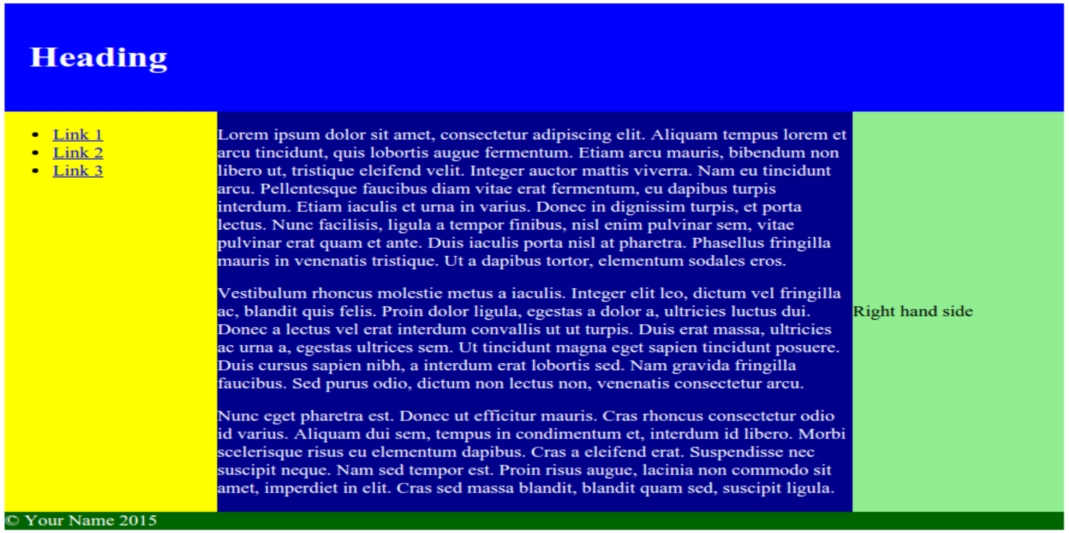
display: table

- You do not need to use float: left with display table
- You should specify a width for the container (in this example *<div class="content">* when setting *display: table*
- You can set widths for each table-cell

Display: table

- When using *display: table-cell* you can set vertical-align (this doesn't work with other display properties!) which aligns the content vertically

```
div {
    display: table-cell;
    width: 20%;
    background-color: lightgreen;
    vertical-align: middle;
}
```



Heading

- Link 1
- Link 2
- Link 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam tempus lorem et arcu tincidunt, quis lobortis augue fermentum. Etiam arcu mauris, bibendum non libero ut, tristique eleifend velit. Integer auctor mattis viverra. Nam eu tincidunt arcu. Pellentesque faucibus diam vitae erat fermentum, eu dapibus turpis interdum. Etiam iaculis et urna in varius. Donec in dignissim turpis, et porta lectus. Nunc facilisis, ligula a tempor finibus, nisl enim pulvinar sem, vitae pulvinar erat quam et ante. Duis iaculis porta nisl at pharetra. Phasellus fringilla mauris in venenatis tristique. Ut a dapibus tortor, elementum sodales eros.

Vestibulum rhoncus molestie metus a iaculis. Integer elit leo, dictum vel fringilla ac, blandit quis felis. Proin dolor ligula, egestas a dolor a, ultricies luctus dui. Donec a lectus vel erat interdum convallis ut ut turpis. Duis erat massa, ultricies ac urna a, egestas ultrices sem. Ut tincidunt magna eget sapien tincidunt posuere. Duis cursus sapien nibh, a interdum erat lobortis sed. Nam gravida fringilla faucibus. Sed purus odio, dictum non lectus non, venenatis consectetur arcu.

Nunc eget pharetra est. Donec ut efficitur mauris. Cras rhoncus consectetur odio id varius. Aliquam dui sem, tempus in condimentum et, interdum id libero. Morbi scelerisque risus eu elementum dapibus. Cras a eleifend erat. Suspendisse nec suscipit neque. Nam sed tempor est. Proin risus augue, lacinia non commodo sit amet, imperdiet in elit. Cras sed massa blandit, blandit quam sed, suscipit ligula.

Right hand side

© Your Name 2015

vertical-align: middle

- *vertical-align: middle* and *display: table-cell* is by far the simplest way to vertically centre content
- There are many options for vertical align, but the main ones you'll need are:
  - Top
  - Middle
  - Bottom

Web Layouts

- By combining display: table, and CSS floats for different parts of your web page you can achieve almost any layout you can think of
- It is generally bad practice to position things exactly with X/Y coordinates using position: absolute
- Similarly, in most cases it's a bad idea to give elements fixed heights: If the content changes it could break the layout

HTML5

- HTML5 is the current version of HTML
- You should use HTML5 and not XHTML or HTML4
- A HTML5 page starts with the doctype

```
<!DOCTYPE html>
```

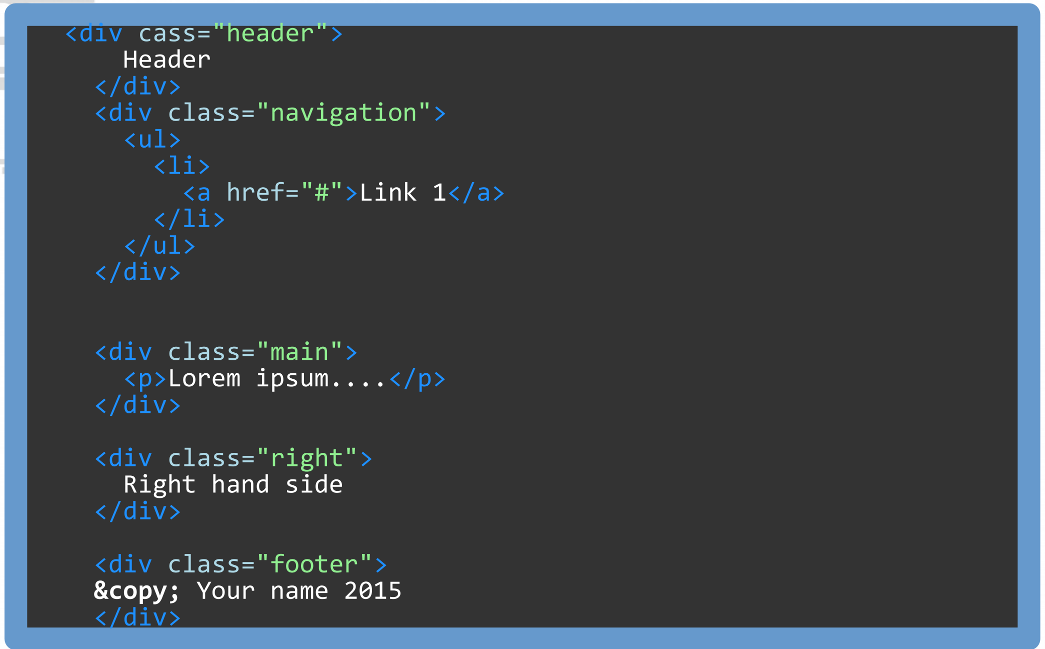
- If you are using any other doctype (or no doctype) browsers will not see your page as HTML5

HTML5

- What is HTML5?
  - HTML5 is the fifth version of HTML and introduces several new features over HTML4 and XHTML
  - For our purposes, this means some new tags available for use on our web pages

HTML5 Tags

- When the W3C (Worldwide Web Consortium) were drafting HTML5 and deciding what to implement they looked at the most commonly used CSS classes. E.g. pages like this:
-



```
  <div cass="header">
      Header
    </div>
    <div class="navigation">
      <ul>
        <li>
          <a href="#">Link 1</a>
        </li>
      </ul>
    </div>


    <div class="main">
      <p>Lorem ipsum....</p>
    </div>

    <div class="right">
      Right hand side
    </div>

    <div class="footer">
    &copy; Your name 2015
    </div>
```

HTML5 Tags

- Using CSS classes to highlight common areas of the page caused several problems:
  - Extra typing for developers (duplication of effort)
  - Difficult for non-humans to understand which part of the page is which. E.g. search engines and screen readers
- The common CSS classes were turned into their own tags

HTML5 Tags

- New tags available in HTML5

<header>	Describes a header (can be of a page or a section)
<footer>	Describes a footer (can be of a page or a section)
<main>	The main content of a page (this is useful for screen readers, they can skip straight to the content). You can only have one main element per page!
<article>	Describes a section which would make sense on its own. A paragraph would not but a topic on a forum would.
<section>	Describes part of an article, e.g. an individual post on a forum
<nav>	Contains the navigation for the web page. You can have more than one <nav> tag. This is also useful for screen readers as they can jump straight to the navigation
<aside>	Describes content related to a section/article that will be displayed differently/elsewhere e.g. sidebars

HTML5

- You should use an HTML5 tag rather than a CSS class if it's applicable
- For example:
  - Article comments should each be a <section>
  - Page headers should be <header>
  - Blog content should be <article>

HTML5 Headers

- Each <section> in HTML5 should contain a <header>
- Each <header> should contain a <h1>-<h6> element
- Although HTML isn't strict so it doesn't really matter if you don't do this

Responsive Web Design

- *Responsive Web Design* is a term for designing a web page that works in browsers of any size
- From widescreen desktop monitors to narrow phones
- This is done by having more than one stylesheet that is applied to the page:
  - e.g. One for mobile phones, one for tablets, one for desktops

Responsive Web Design

- You can apply different style sheets for different browser *widths*
- A desktop browser is usually over 1000px wide
- A mobile browser is usually under 500px wide
- Tablets are in between
- You can use these figures to apply different styles for different type of devices

Mobile Site Considerations

- You cannot get as much on the screen
- Fingers are big and clumsy compared to cursors, make buttons and links easy to press
- Navigation should be accessible via a button rather than a always visible using up valuable space
- You may want to hide some content entirely for mobiles to reduce clutter

Mobile Sites

- When creating a site for mobiles you should set the viewport with. This is almost always:

```
<meta name="viewport" content="width=device-width, initial-scale=1" />
```

- This goes in the <head> tag

Conditional stylesheets

- The <link> tag where you include your css has an optional *media* attribute
- This can be
  - all (default - is always applied)
  - screen (only gets applied when viewed on screen)
  - print (a stylesheet used when the web page is printed)
  -

```
<link rel="stylesheet" href="demo.css" media="print" />
```

Conditional stylesheets

- Additionally, stylesheets can be applied based on a screen width
- There are two main options
- Max-width
- Min-width

```
<link rel="stylesheet" href="mobile.css" media="screen and (max-width: 1000px)" />
```

- This will only apply mobile.css if the browser width is less than 1000px

Conditional stylesheets

- This will only apply desktop.css to devices over 1000px

```
<link rel="stylesheet" href="desktop.css" media="screen and (min-width: 1000px)" />
```

range (>300 and < 800px)

```
<link rel="stylesheet" href="mobile.css"
     media="screen and (min-width: 300px) and (max-width: 800px)" />
```

Multiple stylesheets

- You can add more than one <link> tag to a page
- Each one can have a different media attribute
- It's a good idea to have at least three stylesheets:
  - One for styles that are shared by both desktop and mobile, e.g. fonts, background-colours, etc
  - One for styles relevant only to desktop browsers
  - One for styles relevant only to mobile browsers

Multiple Stylesheets

- For example

```html
<link rel="stylesheet" href="main.css"
    media="screen" />

<link rel="stylesheet" href="mobile.css"
    media="screen and (max-width: 800px)" />

<link rel="stylesheet" href="desktop.css"
    media="screen and (max-width: 800px)" />
```

Stylesheet precedence

- When you have more than one stylesheet, it's possible that they could contradict. For example
- Main.css

```
h1 {
        color: red;
}
```

- Mobile.css

```
h1 {
        color: green;
}
```

Stylesheet precedence

- When this happens, the last stylesheet included will override the first one

```
<link rel="stylesheet" href="one.css"
    media="screen" />


<link rel="stylesheet" href="two.css"
    media="screen" />
```

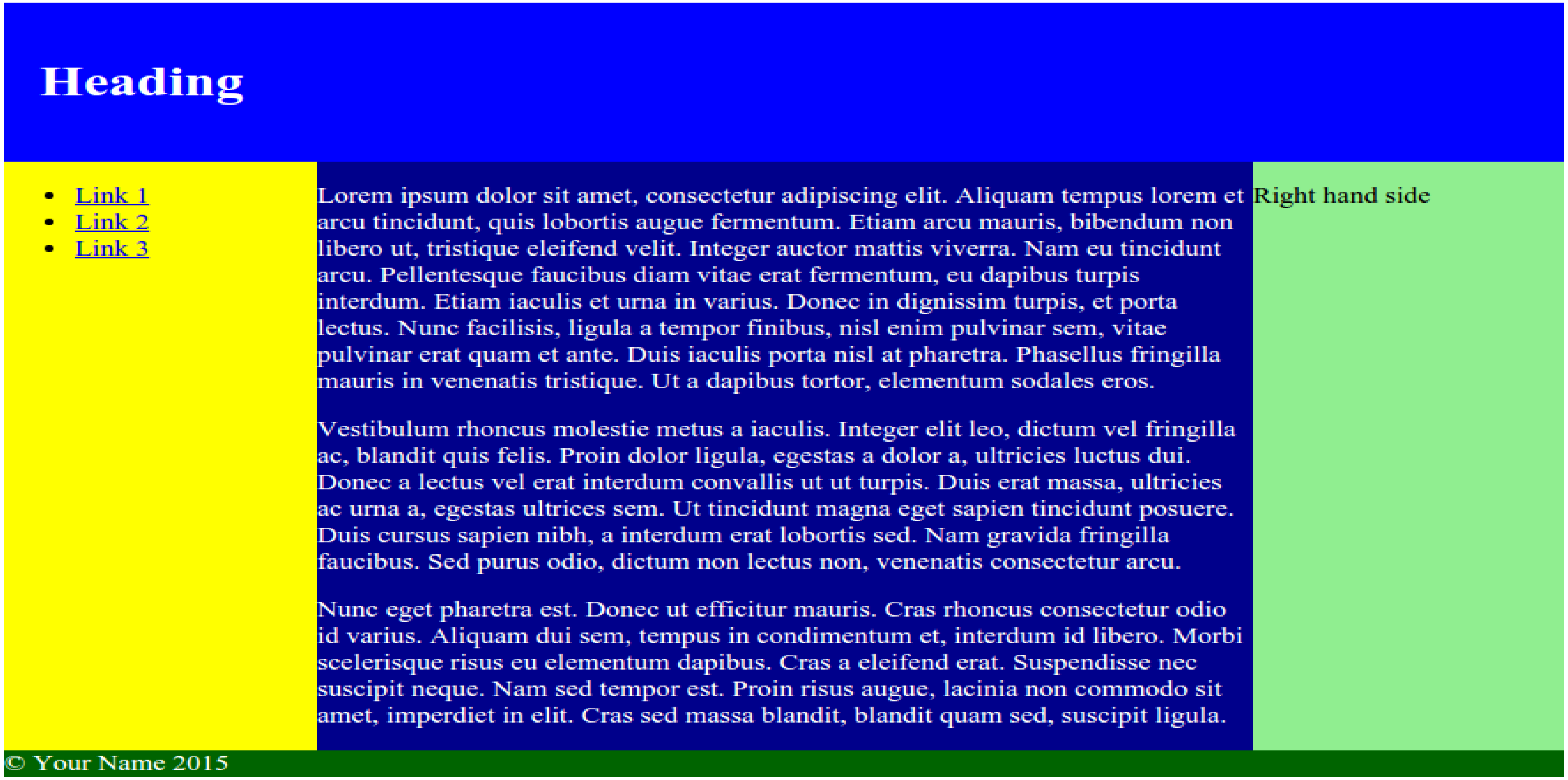
```
h1 {
        color: red;
}
```

```
h1 {
        color: green;
}
```

In this instance, the h1 wil be green because
two.css was included last

Mobile sites

- This site is not mobile friendly



Heading

- Link 1
- Link 2
- Link 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam tempus lorem et arcu tincidunt, quis lobortis augue fermentum. Etiam arcu mauris, bibendum non libero ut, tristique eleifend velit. Integer auctor mattis viverra. Nam eu tincidunt arcu. Pellentesque faucibus diam vitae erat fermentum, eu dapibus turpis interdum. Etiam iaculis et urna in varius. Donec in dignissim turpis, et porta lectus. Nunc facilisis, ligula a tempor finibus, nisl enim pulvinar sem, vitae pulvinar erat quam et ante. Duis iaculis porta nisl at pharetra. Phasellus fringilla mauris in venenatis tristique. Ut a dapibus tortor, elementum sodales eros.

Vestibulum rhoncus molestie metus a iaculis. Integer elit leo, dictum vel fringilla ac, blandit quis felis. Proin dolor ligula, egestas a dolor a, ultricies luctus dui. Donec a lectus vel erat interdum convallis ut ut turpis. Duis erat massa, ultricies ac urna a, egestas ultrices sem. Ut tincidunt magna eget sapien tincidunt posuere. Duis cursus sapien nibh, a interdum erat lobortis sed. Nam gravida fringilla faucibus. Sed purus odio, dictum non lectus non, venenatis consectetur arcu.

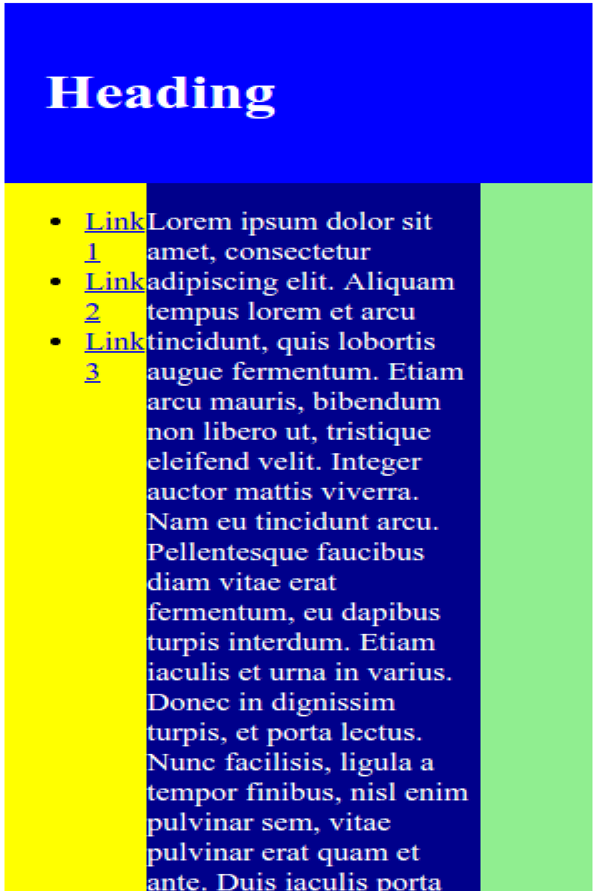
Nunc eget pharetra est. Donec ut efficitur mauris. Cras rhoncus consectetur odio id varius. Aliquam dui sem, tempus in condimentum et, interdum id libero. Morbi scelerisque risus eu elementum dapibus. Cras a eleifend erat. Suspendisse nec suscipit neque. Nam sed tempor est. Proin risus augue, lacinia non commodo sit amet, imperdiet in elit. Cras sed massa blandit, blandit quam sed, suscipit ligula.

Right hand side

© Your Name 2015

Mobile sites

- On a mobile it looks like this:



Heading

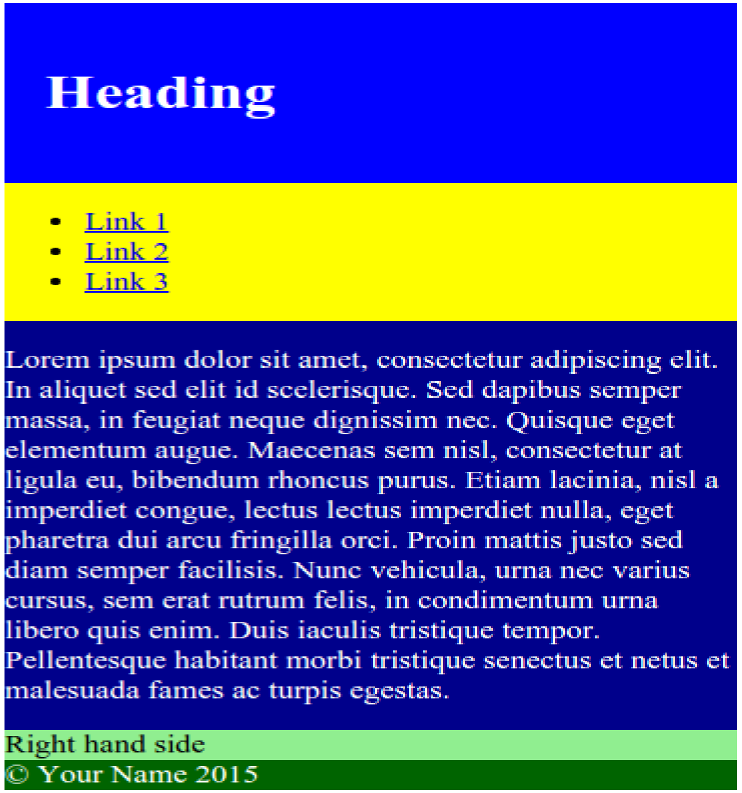
- Link 1
- Link 2
- Link 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam tempus lorem et arcu tincidunt, quis lobortis augue fermentum. Etiam arcu mauris, bibendum non libero ut, tristique eleifend velit. Integer auctor mattis viverra. Nam eu tincidunt arcu. Pellentesque faucibus diam vitae erat fermentum, eu dapibus turpis interdum. Etiam iaculis et urna in varius. Donec in dignissim turpis, et porta lectus. Nunc facilisis, ligula a tempor finibus, nisl enim pulvinar sem, vitae pulvinar erat quam et ante. Duis iaculis porta

Everything is squashed, the links are illegible

Mobile site

- By using a different stylesheet for mobile sites it's possible to "stack" the elements to look like this:



Heading

- Link 1
- Link 2
- Link 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In aliquet sed elit id scelerisque. Sed dapibus semper massa, in feugiat neque dignissim nec. Quisque eget elementum augue. Maecenas sem nisl, consectetur at ligula eu, bibendum rhoncus purus. Etiam lacinia, nisl a imperdiet congue, lectus lectus imperdiet nulla, eget pharetra dui arcu fringilla orci. Proin mattis justo sed diam semper facilisis. Nunc vehicula, urna nec varius cursus, sem erat rutrum felis, in condimentum urna libero quis enim. Duis iaculis tristique tempor. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas.

Right hand side

© Your Name 2015

Exercise 5

- 1) Add the meta viewport tag to the page
- 2) Add a media query so that a mobile version of the website is displayed for browsers narrower than 700px.
  - Hint: Use display: block for all elements to make them stretch across the page!
  - If you resize the browser window you should be able to toggle between the two versions!

“Hamburger” icon

- The “hamburger” icon is commonly used as an icon to display the menu on a mobile website
- This is so commonly used it's always good to follow convention and use the icon
- You can use an image for this, but it's possible to draw it using purely CSS

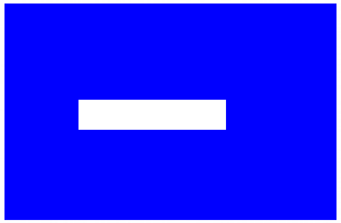
Create a Hamburger icon

- 1) Create a link <a> element in the html which will be used as the button (this doesn't need any text)

```html
<a class="shownav"></a>
```

- 2) Style it as a line with display: block, a background-color and a width/height

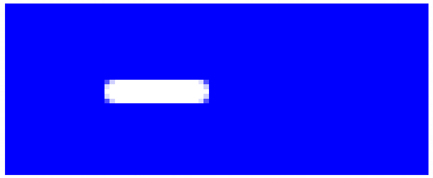
```css
.shownav {
    background-color: white;
    height: 5px;
    width: 20px;
    display: block;
}
```



Create a Hamburger Icon

- 3) Add border-radius: 5px if you want it curved

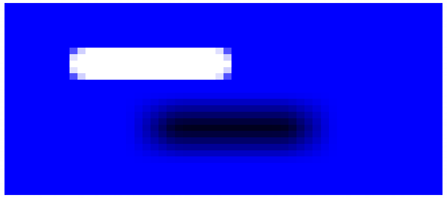
```
.shownav {
     background-color: white;
     height: 5px;
     width: 20px;
     display: block;
     border-radius: 5px;
}
```



- 4) You can use box-shadow to make an element cast a shadow. The syntax is:
  - Box-shadow: [X offset] [Y offset] [Blur amount] [colour]
  - e.g. box-shadow: 10px 10px 5px black will cast a black shadow 5px across, 10px down with a 5px blur

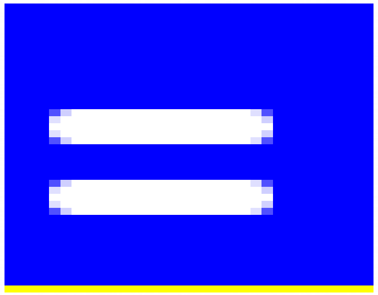
Hamburger icon

```
.shownav {
    background-color: white;
    height: 5px;
    width: 20px;
    display: block;
    border-radius: 5px;
    box-shadow: 10px 10px 5px black;
}
```



- By adding a shadow without a blur, below the line it's possible to draw the bottom part of the icon

```
.shownav {
    background-color: white;
    height: 5px;
    width: 20px;
    display: block;
    border-radius: 5px;
    box-shadow: 0px 10px 0px white;
}
```

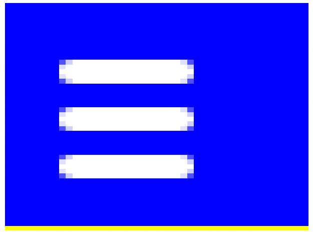


Hamburger icon

- You can add more than one shadow to an element. The syntax is:
  - box-shadow: [SHADOW 1], [SHADOW 2]

4) By drawing a second shadow above the line you can complete the hamburger icon

```
.shownav {
    background-color: white;
    height: 5px;
    width: 20px;
    display: block;
    border-radius: 5px;
    box-shadow: 0px 10px 0px white,
                0px -10px 0px white;
}
```



Exercise 6

Add the hamburger icon to the mobile website

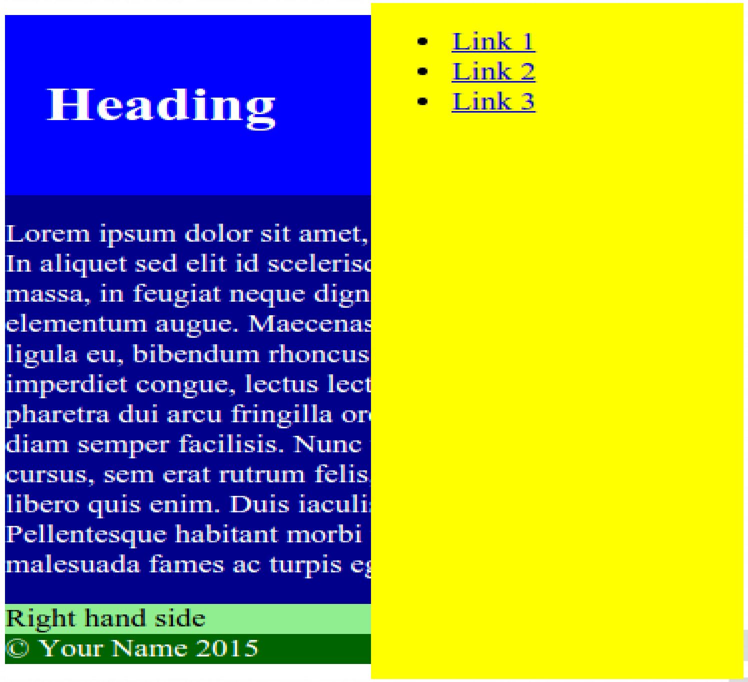
- Hint: You can use display: none to hide it from the desktop version!

Moving the menu

- Instead of having the menu on the page, it's possible to move it to cover up the page. This can be done using:
  - position: fixed which lets you position it on the screen using x/y co-ordinates
  - Setting top to 0
  - Setting left to 50% so it covers half the screen
  - Setting height to 100vh (100 vertical height)
  - Setting the width to 50% so it covers half the screen

Mobile menus

```
nav {
    display: block;
    background-color: yellow;
    position: fixed;
    top: 0;
    left: 50%;
    width: 50%;
    height: 100vh;
}
```



Heading

Lorem ipsum dolor sit amet,
In aliquet sed elit id sceleris
massa, in feugiat neque dign
elementum augue. Maecenas
ligula eu, bibendum rhoncus
imperdiet congue, lectus lect
pharetra dui arcu fringilla or
diam semper facilisis. Nunc
cursus, sem erat rutrum felis
libero quis enim. Duis iaculi
Pellentesque habitant morbi
malesuada fames ac turpis eg

Right hand side

© Your Name 2015

- Link 1
- Link 2
- Link 3

:target selector

- The CSS :target selector is a special CSS selector that's very useful for links:
  - It allows you to apply CSS *after* a link has been clicked

Anchors

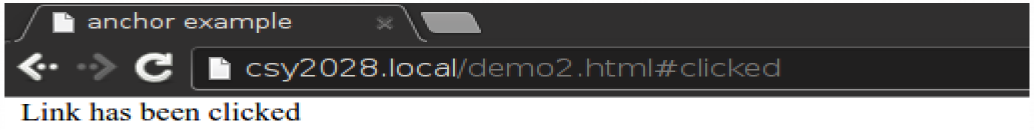
- Before using the :target element you must define an *anchor*
- This is an element on the page with an ID
- You can link to this ID, and when the link is clicked the browser will scroll to the element
- To link to an element use <a href=”#idname”>Click me</a>

Anchors

```
<!DOCTYPE html>
<html>
  <head>
    <title>anchor example</title>
    <link rel="stylesheet" href="demo2.css" />
  </head>

  <body>

    <a href="#clicked">Click me</a>

    <div id="clicked">
      Link has been clicked
    </div>

  </body>

</html>
```

Anchors

- When a link to an anchor is clicked, it will do two things:
- 1) Scroll down to the element (or scroll to the top if the element is at the top of the page)
- 2) Append the anchor name to the URI



```
csy2028.local/demo2.html#clicked
```

Link has been clicked

:target

- The CSS :target selector allows you to apply styles to an element only when it has been targeted (e.g. when #clicked is present in the URI)
-

```html
<!DOCTYPE html>
<html>
  <head>
    <title>anchor example</title>
    <link rel="stylesheet" href="demo3.css" />
  </head>

  <body id="clicked">
    <a href="#clicked">Click me</a>
  </body>

</html>
```

```css
/* Sets the background to red when the body
 isn't targeted */
body {
     background-color: red;
}

/* Sets the background to green when the body
 is targeted */
Body:target {
     background-color: green;
}
```

:target

- You can use CSS to select elements inside the targeted element
- This is useful if you want to have a toggle button
- Add two links:
  - One that sets the target
  - One that clears the target
- Hide one link at a time so only one is visible

:target

```html
<!DOCTYPE html>
<html>
  <head>
    <title>:target example</title>
    <link rel="stylesheet" href="demo4.css" />
  </head>

  <body id="clicked">
    <a href="#clicked" class="on">Target on</a>
    <a href="#" class="off">Target off</a>
  </body>

</html>
```

```css
/*
When the target isn't set, hide the "off"
button and show the "on" button
*/
#clicked .on {display: block;}
#clicked .off {display: none;}

/*
When the target is set, hide the "on"
button and show the "off" button
*/
#clicked:target .on {display: none;}
#clicked:target .off {display: block;}



#clicked {background-color: red;}
#clicked:target {background-color: green;}
```

:target

- :target is a very useful way of toggling on/off the menu on a mobile site
- Using :target, position: fixed and the hamburger icon allows you to create a mobile menu purely in HTML/CSS

Exercise 7

- Add a href to the nav menu so when clicked the menu appears. This is identical to how the "target on"/"target off" example works!

CSS Transitions

- Transitions are a fairly recent addition to CSS
  - Some older browsers don't support them, but almost all do now!
  - Browsers which don't support them will just display the start/end result without animating

CSS Transitions

- Transitions can be used to make elements move, fade in or change in some way.
- They are triggered on page load, or when a CSS class is changed via javascript or via the :target selector
- You need two CSS rules:
  - One for the before state
  - One for the after state

CSS Transitions

- You can use the transition property. The syntax is:
  - transition: [PROPERTY TO TRANSITION] [DURATION]
  -

```html
<!DOCTYPE html>
<html>
  <head>
    <title>:target example</title>
    <link rel="stylesheet" href="demo5.css" />
  </head>

  <body id="clicked">
    <a href="#clicked" class="on">Start animation</a>
    <a href="#" class="off">Reset</a>

    <div class="box"></div>
  </body>

</html>
```

```css
/*
 Before state + transition definition
*/
.box {
    background-color: red;
    width: 50px;
    height: 50px;
    margin-left: 0px;
    transition: margin-left 2s;
}

/*
 After state
*/
#clicked:target .box {
    margin-left: 400px;
    background-color: green;
```

Exercise 8

- Add a transition to the menu so that it slides in/out