CI135 Introduction to Web Development

Lecture 5

CSS box model and measurements

This lecture will cover…

- The cascade – "specificity"
- The CSS box model
  - Styling the box
  - The `background` property
- CSS relative length units – pixel, %, em, rem, vw, vh
- This week's lab tutorial
- This week's reading

The cascade

- Why ***cascading*** style sheets?
- Your style sheets are about to get complex – styling web pages involves working out the best – *i.e.* most economical and specific - way to match an **element** with a **selector** (see week 4)
- The **cascade** is how browsers determine which rule, or rules apply to which element – and how to handle conflicts between rules
- For example…

The cascade

- Browser style sheet –

```
body {color: #000; } /*font colour black*/
```

- External style sheet

```
body {color: #333; }
p    {color: #F30; }
```

- Your external style sheet rule targets the **<body>** element and specifies font colour; this over-rides the browser CSS
- The rule that targets the **<p>** element and specifies font colour over-rides the **body** rule
- The order in which rules are written is important – a lower rule in the style sheet over-rides the rules that precede it

The cascade - specificity

- Lab tutorial 4 – used a **descendant selector** as a way of matching the main navigation with **specificity**

```
<nav class="menu">
  <ul>
    <li><a href="">Home</a></li>
    <li><a href="">Guide me</a></li>
    <li><a href="">Contact</a></li>
  </ul>
</nav>
```

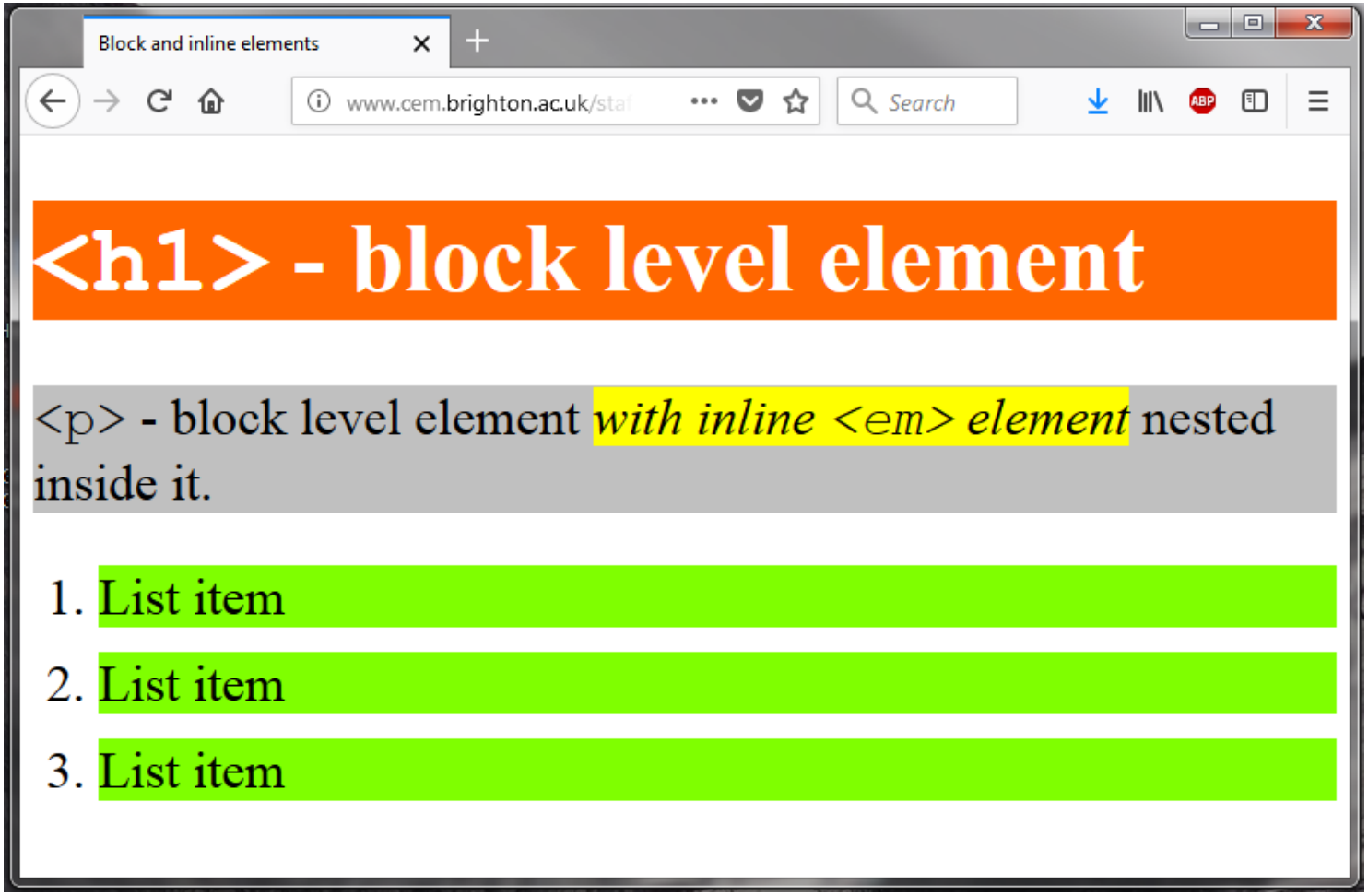
```
.menu li {
        list-style-type: none;
        font-size: 18px;
        margin-left: 0px;
        margin-bottom: 20px; }
```

- This selector matches *only* the list items that are **descendants** of – *i.e.* nested within - the element with the class of "menu"

CSS box model

- The **CSS box model** controls how elements are displayed and positioned on a web page
- Every HTML element - including the **<body>** - is treated as if it is a **box**
- Some elements/boxes are **block level** – these take up a whole line on the page
- Other elements/boxes are **inline level** – these are nested within a block element...
- Boxes can be organised in a sequence, one after another, or be nested within other boxes

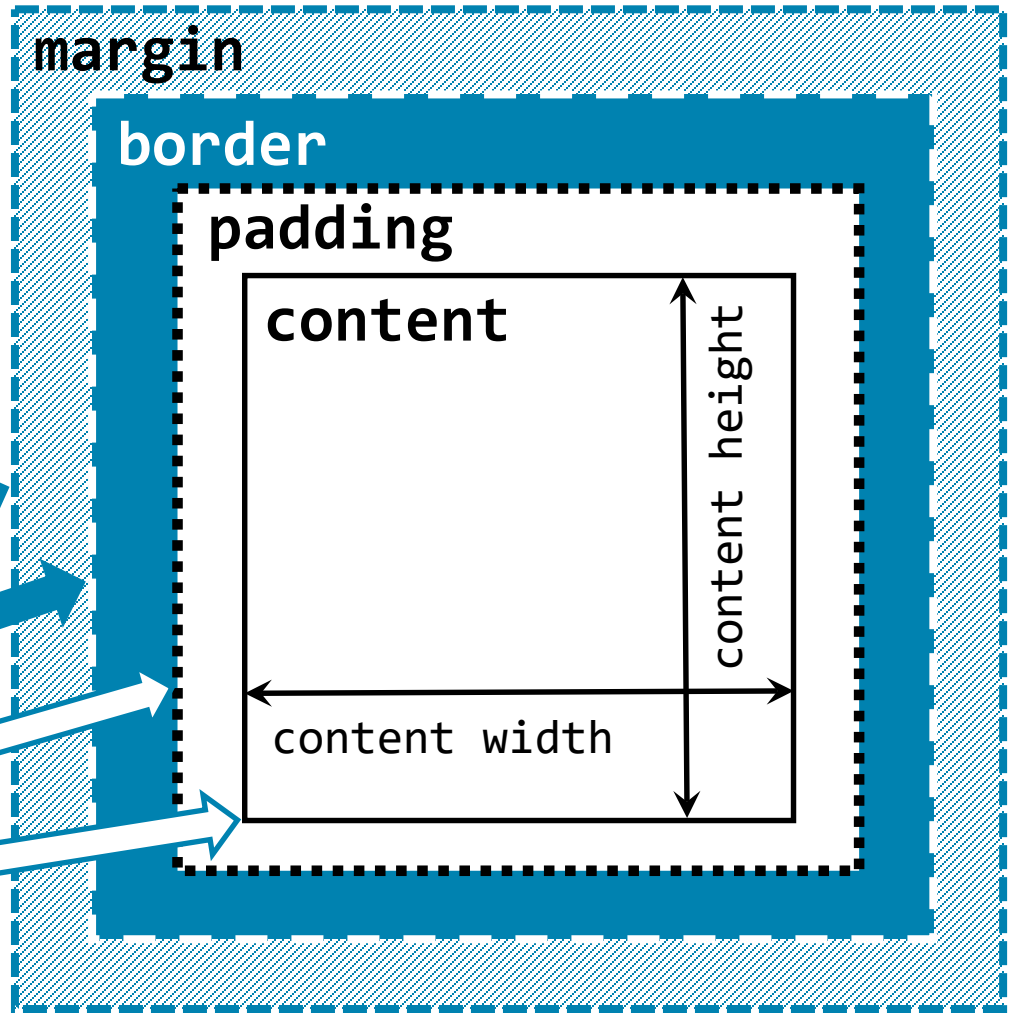
Block and inline elements



http://jh1033.brighton.domains/ci435/tutorials/block-inline.html

CSS standard box model

- The box model describes the space taken up by an element
- A box has 4 edges
  - `margin` edge
  - `border` edge
  - `padding` edge
  - `content` edge



CSS box model: box-sizing

- W3C CSS box model standard is –

  `{ box-sizing:content-box; }`

  i.e. The `width` and `height` properties of the box include only the content box – they do not include the `padding`, `border-width` or `margin` lengths
- This is the default setting of browsers that support the W3C standard

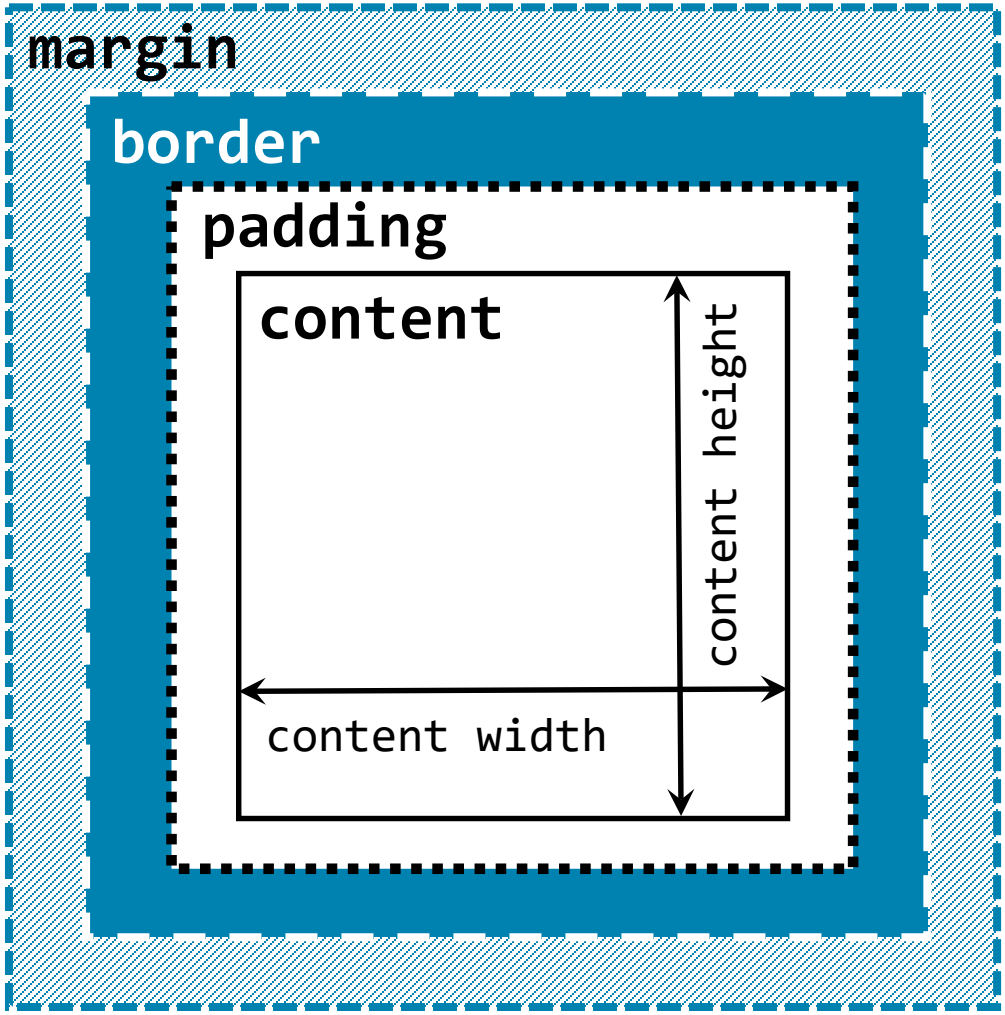
N.B. Older versions of *Internet Explorer* used a non-standard box model -

`{ box-sizing:border-box; }`

This included the width of the content, padding and border in the box width and height measurements

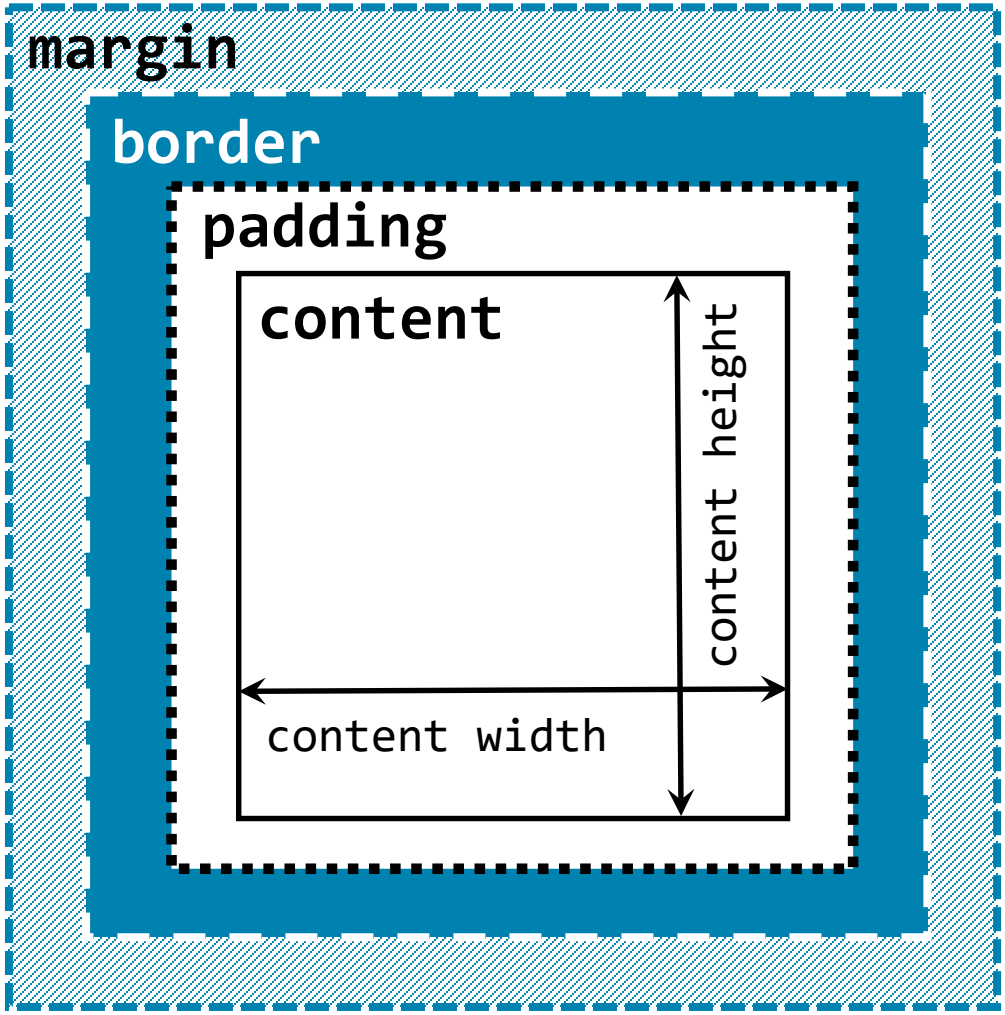
CSS box model: `width` and `height`

- If `box-sizing` is the default, these CSS properties control the **`content-box`** size –
  - `width`
  - `min-width`
  - `max-width`
  - `height`
  - `min-height`
  - `max-height`



CSS box model: `margin` and `padding`

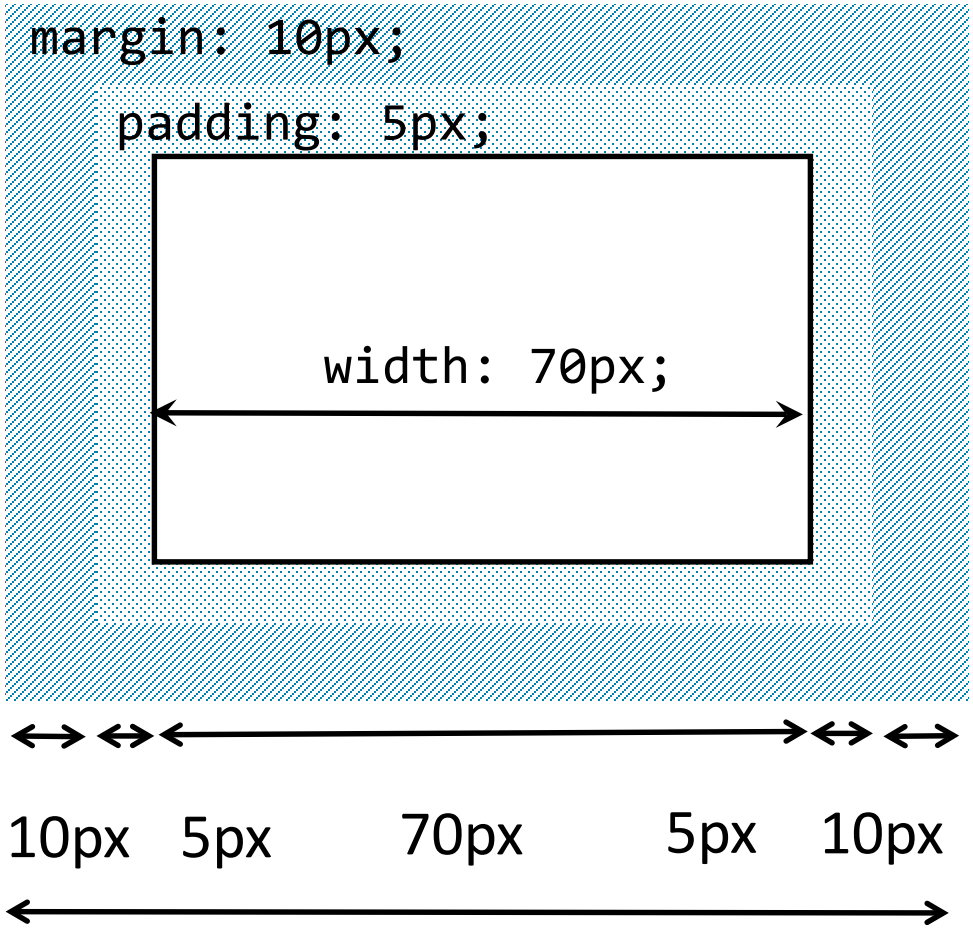
- Use the **`padding`** property to create space *inside* the element – between the content and the edge of the padding
- Use the **`margin`** property to create space *outside* the element, between it and adjacent elements



CSS box model: measurement

- The **width** and **height** properties refer to the **content area** of the box
- **margin** and **padding** increase the total size of the box

```
.box1 {
  width: 70px;
  padding: 5px;
  border: 0;
  margin: 10px;
  }
```



CSS box model: properties

- The **`height`** property is usually not defined – because the box content may be of variable size and needs to flow into the space it needs
  - *E.g.* the `<article>` elements in your Learning Journal will have different amounts of content and therefore will be different `height`
- You can define styles for each side of the box –

  `padding-top, padding-right, padding-bottom, padding-left`

  `margin-top, margin-right, margin-bottom, margin-left`
- Use the shorthand **`margin`** and **`padding`** properties if all 4 sides are the same value - *e.g.* `{margin: 10px;}`

Styling the box – block elements

** ** Cum sociis natoque penatibus et magnis. Nullam … ** **

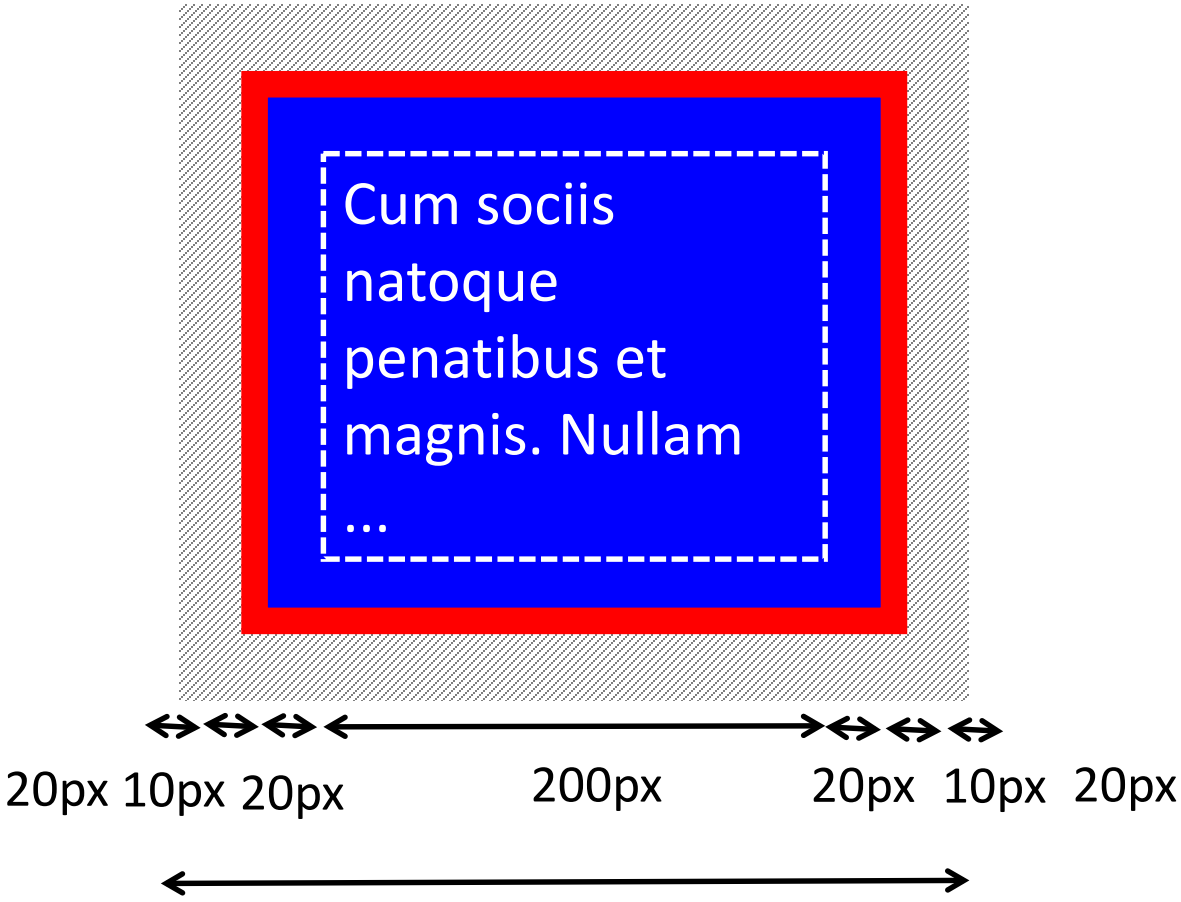
```
p {
width:200px;
background-color:blue;
padding:20px;
border:10px solid red;
margin:20px;
}
```

`border:width style color;`

When the content area has the **`background-color`** or **`background-image`** property this extends into the padding area*

*sometimes useful to put color in just for testing layout

Styling the box – block elements <p>



Adds up to **300 px** total size across

Styling the box – inline elements

```
<body>
<p>  Cum sociis natoque <em>  penatibus et magnis. </em>
   Nullam ... </p>
</body>

body {
  font-family: Arial, sans-serif;
  color: #000;
}
em {
  background-color: #FF0;
}
```

Cum sociis natoque penatibus et magnis. Nullam ...

Styling the box – inline elements <a>

```
<nav class="menu">
  <ul>
    <li>Learning Journal</li>
    <li><a href="">Guide</a></li>
    <li><a href="">Contact me</a></li>
  </ul>
</nav>
```


```
.menu a {
        color: #000;
        padding: 5px;
        text-decoration: none;
        border: 2px solid #0090D2;
        border-radius: 3px;}
```

```
.menu a:hover {
        background-color: #0090D2;
        color: #FFF;}
```

Styling the box – background

- 'background' properties can be styled for all regions of the box
- Default value is transparent – the background-color or background-image of the parent box in which an element is nested shows through
- The background-image property applies an image (PNG, JPG, GIF, SVG) or CSS3 gradient
- Small images can be used: these tile - *i.e.* repeat - to fill the browser window

```
background-repeat: repeat;
```

- Large images - if used – should be set to

```
background-repeat: no-repeat;
```

Styling the box - background

- This example web page shows how to style the box using these properties –
  - `background`
  - `background-color`
  - `background-image`
  - `background-repeat`
  - `background-position`
- Boxes have been laid out using a basic CSS3 grid
- Includes links to web resources to learn more

http://jh1033.brighton.domains/ci435/tutorials/background/background.html

CSS units

- CSS properties define the `length` *i.e.* distance value of elements - *e.g.* `font-size`; `line-height`; `border-width`; `margin` and `padding`; `width` and `height`
- **Absolute** and **relative** `length` units –

Absolute - fixed	**Relative** - to another length property
in - inches	**px** – computer screen pixels
cm - centimeters	**%** - percentage of another value
mm - millimeters	**rem** – font size of the root element
pt - points (=1/72 inches)	**em** – font size of the element
	vw - 1% of viewport width
	vh - 1% of viewport height

- https://css-tricks.com/fun-viewport-units/

Absolute and relative length units

- Absolute units are **fixed** measurements
- But a computer screen is not like a printed page or a fixed size canvas...
- Fixed measurements could only work if the browser knew precisely the display resolution of the screen and size of the device viewport
- Relative length units have no fixed value: their size is relative to a value specified for another property
- **Pixel**, **%**, **rem**, **em, vw** and **vh** are the units most commonly used in web development

Relative length units: **pixel**

- Up to now all the example CSS has used pixel measurements *e.g.* `body { font-size: 16px;}`
- **Pixels** give an *illusion* of control – *i.e.* that element measurements can be fixed and absolute
- In fact, pixel size is relative to screen resolution – the same font size will look large on a low resolution display, smaller on a high res display, tiny on a high-density retina display
- Any box property can be measured in pixels – `font-size`, `width` and `height`, `margin`, `padding`, `border` *etc.*
- Pixel width measurements can be a problem for responsive web design because elements sized in pixels are not fluid or flexible and will not adapt to different devices

Relative length units: **percentage**

- **Percentage** provides excellent flexibility in designing fluid web pages - elements that adjust to fit the browser viewport
- **%** values are *always relative* to a value that has been declared elsewhere – *e.g.* the `width` of a parent element
  - **`#wrapper { width: 80%; }`** = 80% of its parent element, the **`<body>`**, which fills 100% of the **viewport**
  - **`main { width: 70%; }`** – 70% of the wrapper div
- Because **`<main>`** (the child element) is nested inside **`<div id="wrapper">`** (the parent element), its measurement is a percentage of the wrapper width - 70%

Relative length units: **em**

- **Em** is the most **scaleable** relative length unit
- The unit can be used to specify the measurement of any CSS box property
- When a user resizes text in their browser (not zoomed) both font and layout box will resize to scale – elements are "elastic"
- An em unit is the width of the letter 'M' in the current font
  - If the `font-size` of the element is 16 px then 1 em=16 px
- The concept of ems is quite difficult to grasp - until you have experience of using the measurement

Relative length units: **em**

- Default font size defined by the browser settings is **16 px**
- Specify the baseline `font-size` in the rule for the **`<html>`** or **`<body>`** element – the first rule in the style sheet

```
body {font-size: 100%; } /* = 16px*/
```

100% of the browser default means 1 em = 16 pixels

- Define font size of all other elements in ems - *e.g.* **h1 – 2.5 em** (=40px); **h2 -2 em** (=32px); **p – 0.875 em** (=14px)
- These elements will have a font size relative to the body `font-size`

Relative length units: **em**

- When elements are nested within other elements using em units gets tricky

```
<body>
    <h1>Page heading</h1>
    <main>
        <h2>Sub-heading</h2>
    </main>
</body>
body {font-size: 100%;} /* = 16px */
h1 {font-size: 2.5 em;} /* =40px */
main {font-size: 1.25em;} /* = 20px */
h2 {font-size: 2em;} /* =40px; */
```

Because em is relative to the current font, the h2 font size is relative to the <main> font-size, not the <body font-size

http://jh1033.brighton.domains/ci435/tutorials/em.html

Relative length units: **rem**

- **rem** is much easier to understand than **em**
- The **root element** of any web document is the `html` element – `<html>` starts the document (after the `doctype` declaration) and ends it `</html>`
- The **rem** unit is equal to the value of the `font-size` of the **root element -**

```
html {font-size:100%;}
```

- **100% =16px**, the default `font-size` of browser style sheets

```
h1  {font-size: 2.5rem;}  /*16 x 2.5 = 40 px*/
h2  {font-size: 2rem;}  /*16 x 2 = 32 px*/
article {margin-bottom: 0.5rem;} 16 x 0.5= 8px
```

Viewport percentage lengths - vw, vh

- Viewport percentage lengths define the length of an element relative to the size of the **viewport**
  i.e. the visible portion of the web page

  `vw` - 1/100th viewport width

  `vh` - 1/100th viewport height

  `vmin` - 1/100th of the smallest side

  `vmax` - 1/100th of the largest side
- In portrait mode `1vmax=1vh`, `1vmin=1vw`
- In landscape mode `1vmax=1vw`, `1vmin=1vh`

Viewport percentage lengths - vw, vh

- vw is mainly used in web typography as it allows text to resize according to the viewport size
- The following example is from https://css-tricks.com/viewport-sized-typography/

CSS -

```
h1 { font-size: 5.7vw; }
h2 { font-size: 4.0vw; }
h3 { font-size: 2.8vw; }
p { font-size: 2.5vw; }
```

Demo

https://css-tricks.com/examples/ViewportTypography/

This week's reading

- Jon Duckett, *HTML & CSS: Design and build websites.* Chapter 12 (Text); Chapter 13 (Boxes).
- **Advanced** - *W3C CSS2.1 Specification* – Chapter 8, The box model http://www.w3.org/TR/CSS2/box.html
- Styling text in CSS - https://developer.mozilla.org/en-US/Learn/CSS/Basic_text_styling_in_CSS
- Background and border -

  https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Background_and_Borders

  https://css-tricks.com/almanac/properties/b/background-image/

  https://css-tricks.com/almanac/properties/b/background-repeat/

  https://css-tricks.com/almanac/properties/b/background-attachment/