The University Of Sheffield.

COM1008:

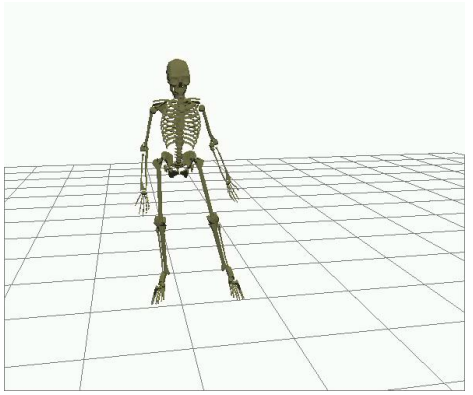
Web and Internet Technology

Lecture 2: HTML and CSS

Dr. Steve Maddock
s.maddock@sheffield.ac.uk

1. Introduction

- When creating a web page, separate the structure and appearance
  - HTML – structure (first)
  - CSS – appearance (second)
- Markup language
- We will use HTML5 and CSS
  - CSS3 features in a later lecture



2. Markup language

- Annotate a document to show its structure
- eXtensible Markup Language (XML)
  - Strict set of rules for defining the encoding of a document or an arbitrary data structure
  - metalanguage – a language for defining other markup languages

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE recipe PUBLIC "-//Happy-Monkey//DTD RecipeBook//EN"
"http://www.happy-monkey.net/recipebook/recipebook.dtd">

<recipe>

    <title>Peanut-butter On A Spoon</title>

    <ingredientlist>
        <ingredient>Peanut-butter</ingredient>
    </ingredientlist>

    <preparation>
        Stick a spoon in a jar of peanut-butter,
        scoop and pull out a big glob of peanut-butter.
    </preparation>

</recipe>
```

http://en.wikipedia.org/wiki/File:RecipeBook_XML_Example.png

2.1 HTML

- HTML is a markup language with pre-defined presentation semantics

start tag	mix of text and elements	matching end tag
<??>		</??>

```
<!DOCTYPE html>
<html>
 <head>
  <title>Sample page</title>
 </head>
 <body>
  <h1>Sample page</h1>
  <p>This is a <a href="demo.html">simple</a> sample.</p>
  <!-- this is a comment -->
 </body>
</html>
```

Can also have empty elements, e.g.

A general document is made up of elements

Attributes are named properties of elements

attribute_name = “value inside quotes”

2.1 HTML

- HTML5 being developed simultaneously by WHATWG and W3C
- (Web Hypertext Application Technology Group, World Wide Web Consortium)
- 'living HTML spec'
- Current spec is over 900 pages

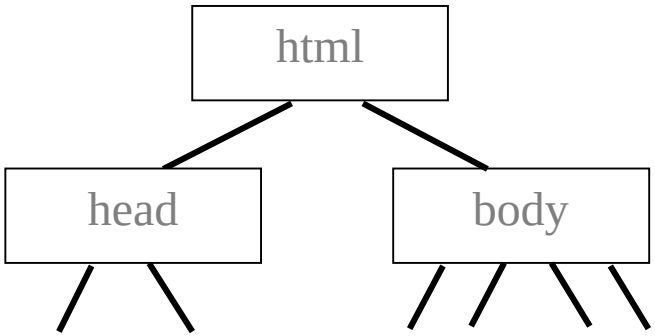
www.w3.org/TR/html5
https://html.spec.whatwg.org/multipage/

An example…

- We'll look at the HTML involved in structuring a page with:
  - A heading
  - A short piece of text
  - A bulleted list containing 3 items
  - A hyperlink attached to one of the items in the list
- Then, we'll use a stylesheet to change the appearance.

3. Overall structure of the page

- There are two main parts to the document:
  - head and body
- The structure of the document can be represented as a tree
  - html element is the root



Specifying the doctype triggers browsers that need it to operate in html standards mode

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>Team X</title>
</head>

<body>
  <!- a comment in place
       of some contents -->
</body>

</html>
```

3.1 History: XHTML document type declaration

- XHTML is an XML-based language, defined by a document type definition (DTD)
- An XHTML document must include a *DTD*

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- Modern Web browsers have the DTD incorporated into their code.

DTD	Public Identifier	URL
Strict	-//W3C//DTD XHTML 1.0 Strict//EN	http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd
Transitional	-//W3C//DTD XHTML 1.0 Transitional//EN	http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd
Frameset	-//W3C//DTD XHTML 1.0 Frameset//EN	http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd

3.2 The document head

- Content of head element is not rendered in browser window
- **title element** is compulsory and is displayed in the title bar
- **meta** element provides a general-purpose mechanism for adding metadata to HTML documents
- **charset** defines the document's character encoding
  - Security risk of not setting it
  - Must be in first 512 bytes
  - multibyte character encoding for Unicode.


```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>Team X</title>
</head>

<body>
   <!-- content -->
</body>

</html>
```

4.2 The document head

- Other meta data elements use name and content attributes
- Other elements
  - link – stylesheets (see later in this lecture)
  - script – JavaScript (see a later lecture)

```
<head>
  <meta charset="utf-8" />
  <title>Team X</title>
  <link rel="stylesheet" href="teamx1.css" />
  <meta name="author" content="Steve Maddock" />
  <meta name="description" content="Team X web site for COM1008" />
  <meta name="keywords" content="Team X, sports" />
</head>
```

4.3 The body of the document


```
<body>
  <h1>Team X</h1>
  <p>We are Team X.</p>
  <ul>
    <li>Art</li>
    <li><a href="http://www.thesimpsons.com/">Bart</a></li>
    <li>Cara</li>
  </ul>
</body>
```

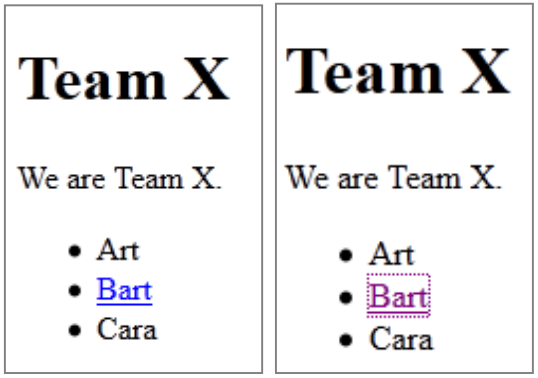
- Six heading elements: **h1**, h2, h3, h4, h5, h6
- List types:
  - Unordered lists (**ul**); items (**li**)
  - Ordered lists (ol); items (li)
  - Definition lists (dl); items (dt term and dd definition)
- The hyperlink (anchor) element is **a**

5. Hyperlinks

- The ends of hyperlinks are called *anchors*
  - *They link a source* and a *destination*

```
<a href="URL">link text</a>
```

- Default display: blue and underlined
- Once visited: purple and underlined




```
<body>
  <h1>Team X</h1>
  <p>We are Team X.</p>
  <ul>
    <li>Art</li>
    <li><a href="http://www.thesimpsons.com/">Bart</a></li>
    <li>Cara</li>
  </ul>
</body>
```

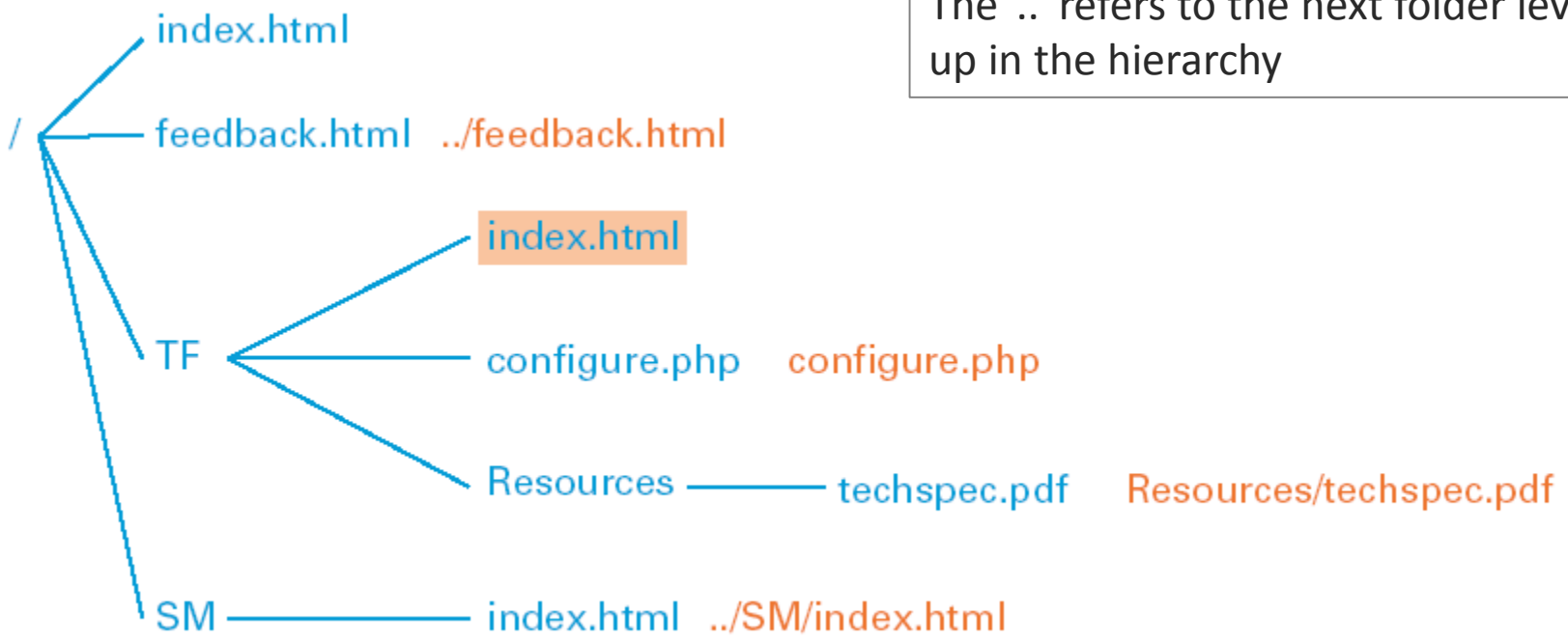
Uses an absolute URL

5.1 Relative URLs

- Orange gives the relative filenames from the start position index.html
- From within index.html :

```
<a href="../feedback.html">feedback</a>
```

The '..' refers to the next folder level up in the hierarchy



Absolute URL : feedback

Chapman, N and J. Chapman, Web Design: A complete introduction, John Wiley & Sons, 2006.

5.1 Relative URLs

- Useful because can easily move whole Web site to a different host machine, as the links are relative
- Fragment identifier can be used to link to a location within a document:

```
<a href="../feedback.html#Comments">Send a comment</a>
```

  - The location in the document is identified with an id:

```
<h1 id="Comments">Your comments</h1>
```

- Implicit destination anchor at the start of every document

```
<a href="#">Jump to top of page</a>
```

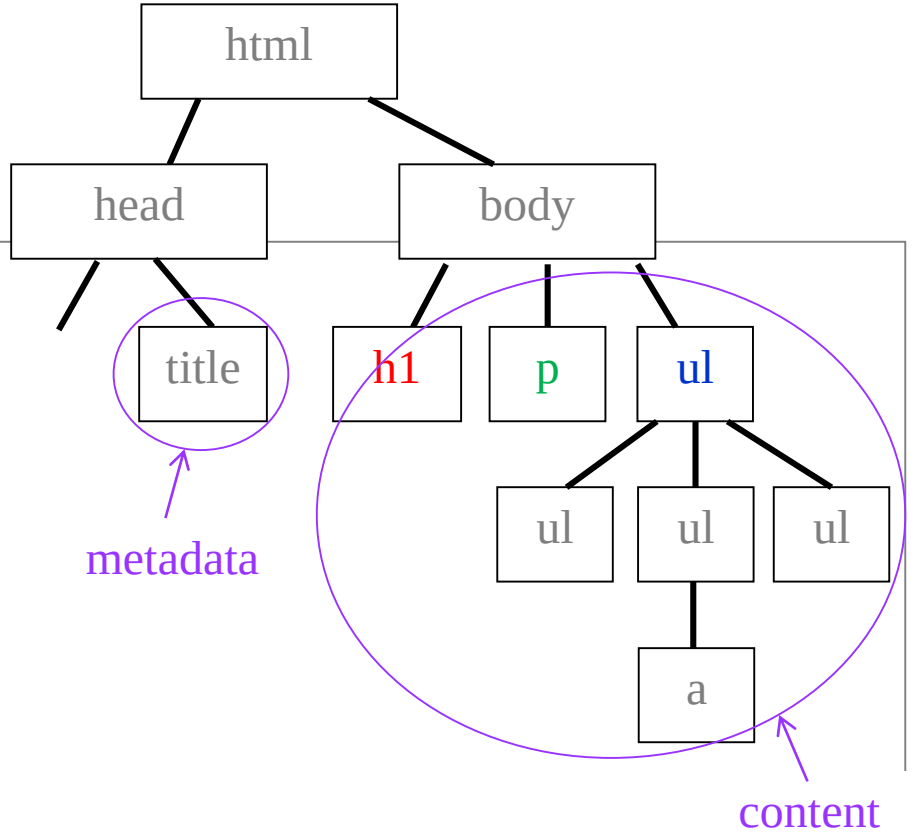
6. Document structure

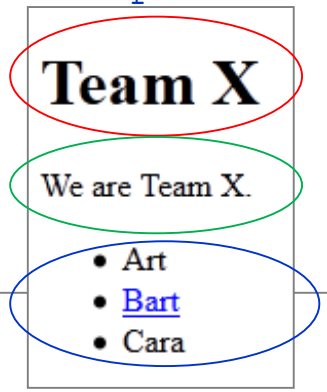
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>Team X</title>
</head>

<body>
  <h1>Team X</h1>
  <p>We are Team X.</p>
  <ul>
    <li>Art</li>
    <li><a href="http://www.thesimpsons.com/">Bart</a></li>
    <li>Cara</li>
  </ul>
</body>

</html>
```





6.1 Layout – should reflect document structure

- Some editors will layout the code for you
- There are also tools to tidy up your HTML, e.g. HTML Tidy (http://infohound.net/tidy/)

```
<body>
  <h1>Team X</h1>
  <p>We are Team X.</p>
  <ul>
    <li>Art</li>
    <li><a href="http://www.thesimpsons.com/">Bart</a></li>
    <li>Cara</li>
  </ul>
</body>
```

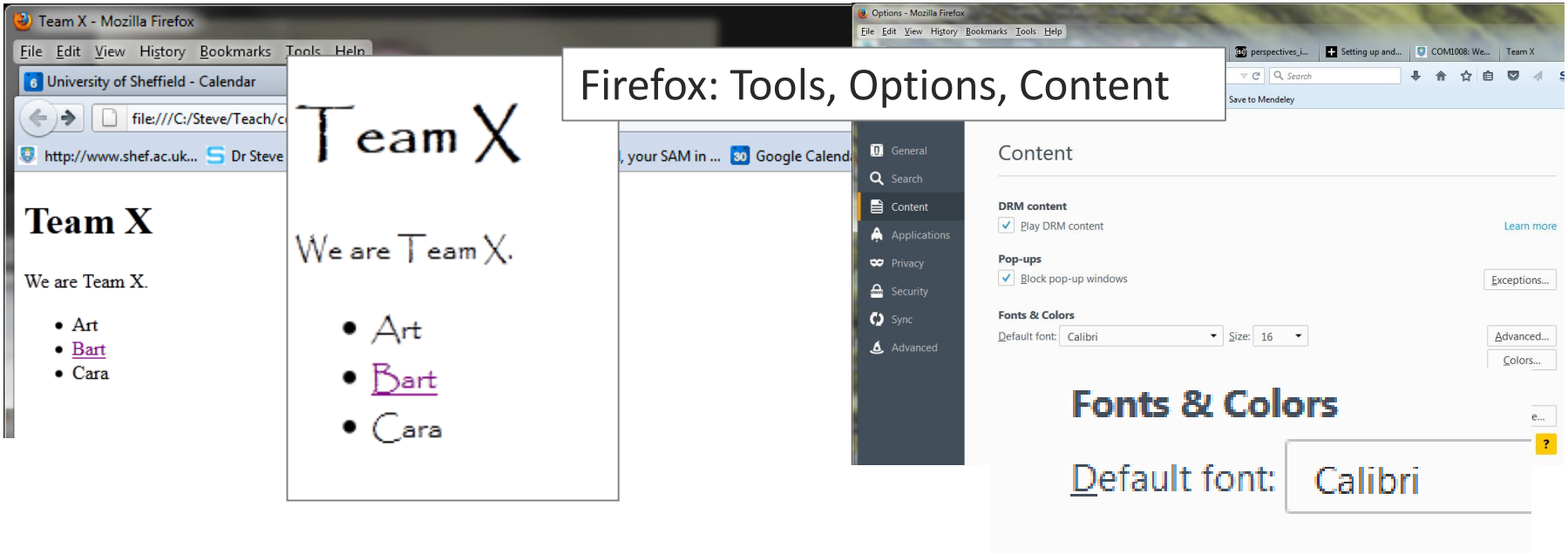
Indent 2 spaces

```
<body>
    <h1>Team X</h1>
    <p>We are Team X.</p>
    <ul>
        <li>Art</li>
        <li><a href="http://www.thesimpsons.com/">Bart</a></li>
        <li>Cara</li>
    </ul>
</body>
```

Indent 4 spaces

7. Appearance

- A browser (sometimes called a User Agent) has defaults for the appearance of each element
- A browser can offer the user options to override these defaults
- *Alternative*: Glue a stylesheet



7.1 "Gluing" a stylesheet to a document

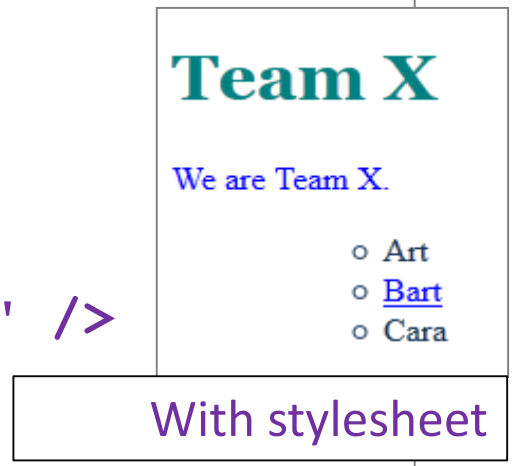
```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>Team X</title>
  <link rel="stylesheet" href="teamx1.css" />
</head>

<body>
  <h1>Team X</h1>
  <p>We are Team X.</p>
  <ul>
    <li>Art</li>
    <li><a href="http://www.thesimpsons.com/">Bart</a></li>
    <li>Cara</li>
  </ul>
</body>

</html>
```

Multiple stylesheets can be glued



With stylesheet

Without stylesheet

7.2 Alternative ways to “glue” a stylesheet to a document

- Link to a stylesheet which is in its own file: ← Preferred option

```
<head>
  <link rel="stylesheet" href="teamx1.css" />
</head>
```

- ‘Old style’:

```
<head>
  <link rel="stylesheet" type="text/css" href="teamx1.css" />
</head>
```

- Embed it in the head element:

```
<head>
  <style> ...style information  goes here... </style>
</head>
```

- Inline it:

```
<h1 style="color: teal;">My heading</h1>
```

teamx1.html

8. The CSS


```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>Team X</title>
  <link rel="stylesheet" href="te
</head>

<body>
  <h1>Team X</h1>
  <p>We are Team X.</p>
  <ul>
    <li>Art</li>
    <li><a href="http://www.thesi
    <li>Cara</li>
  </ul>
</body>

</html>
```

teamx1.css

```
h1 {
  color: teal;
  font-family: Georgia, serif;
  font-size: 200%;
}

p {
  color: blue;
}

ul {
  padding-left: 100px;
  list-style-type: circle;
}

li {
  color: #123456; /* hexadecimal */
}
```

A comment.

#123456 is the same as rgb(18,52,86)

8.1 The CSS

- A stylesheet is a set of *rules*

```
Selector { Property: Value; }
```

- This applies to all occurrences of the relevant element
  - (See inheritance, cascading and specificity)
- Multiple declarations are separated by semicolons
- If property value has a space, use quotes:

```
h1 {
  color: teal;
  font-family: Georgia, serif;
  font-size: 200%;
}

p {
  color: blue;
}

ul {
  padding-left: 100px;
  list-style-type: circle;
}

li {
  color: #123456; /* hexadecimal */
}
```


```
font-family: "Lucida Handwriting", Papyrus, serif;
```

8.2 CSS selectors

- Single elements
  - h1 { color: blue; }
- Multiple items – the items are given in a comma-separated list
  - h1, h2 { color: blue; }
- Label specific html elements and then refer specifically to those elements
  - Use of ids and classes
- Use contextual selection
  - See later lecture
- As the number of CSS rules grows, we need to know which rule applies to which element – cascading, inheritance and specificity

8.3 id and

- HTML elements can be labelled with a unique id, which is used only once in the HTML document. Example:
  - `<p id="special">...</p>`
- In the CSS file, the #operator is used to refer to these:

`#special {`
`  color: blue;`
`}`
	Matches an element with id “special”

`h1, h3, #special {`
`  font-style: italic;`
`}`
	Matches all h1 elements and all h3 elements, and an element with id “special”
	Additive process – an element with id “special” is now blue and italic.

8.4 class and .

- If we want to style multiple things at the same time, we can label each HTML element using a class:
  - `<p class="italic">...</p>`
  - `<h1 class="italic">...</h1>`
- Multiple classes can be added to the same element, separated by spaces:
  - `<h1 class="italic purple">...</h1>`
- In the CSS file, the . operator is used to refer to these

`p.italic { font-style: italic; }`	Matches paragraphs using this class
`*.italic { font-style: italic; }`	Matches all elements using this class
`.italic { font-style: italic; }`	Matches all elements using this class
`.purple { color: purple; }`	Matches all elements using this class

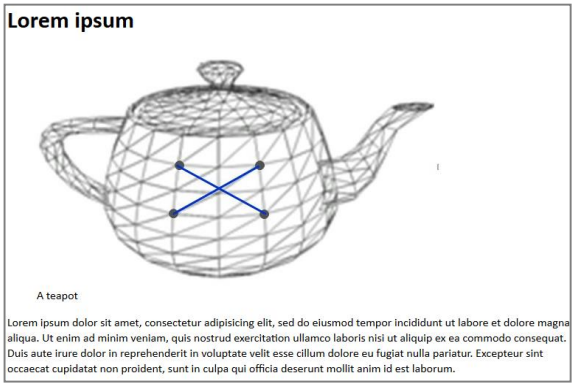
9. Flow

```
<body>
  <h1>Team X</h1>
  <p>We are Team X.</p>
  <ul>
    <li>Art</li>
    <li><a href="http://www.bbc.co.uk/">Bart</a></li>
    <li>Cara</li>
  </ul>
</body>
```

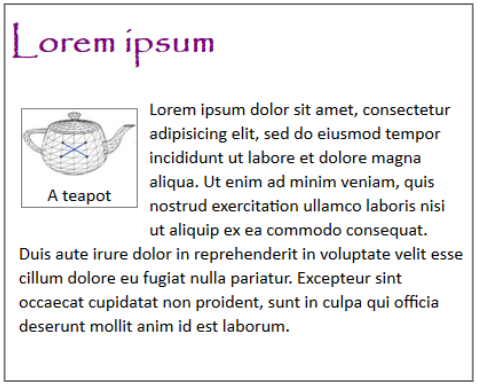
- By default, elements 'flow' onto the page based on their order in the HTML document
  - **Block** elements - begin on new line
  - *Examples*: p, div, ul, li, table, h1, h2, h3, h4, h5, h6, hr
  - **Inline** elements - displayed within blocks
  - *Examples*: a, img, span, em, strong, code, big, small, br, cite
- There are a range of ways to alter the flow:
  - Change the *position* property of an element
  - 'float' an element to one side of its container
  - Change the *display* property, e.g. none

9.1 Example: Floating an image

- Including an image on the web page…



- And changing the appearance of the page with a stylesheet…



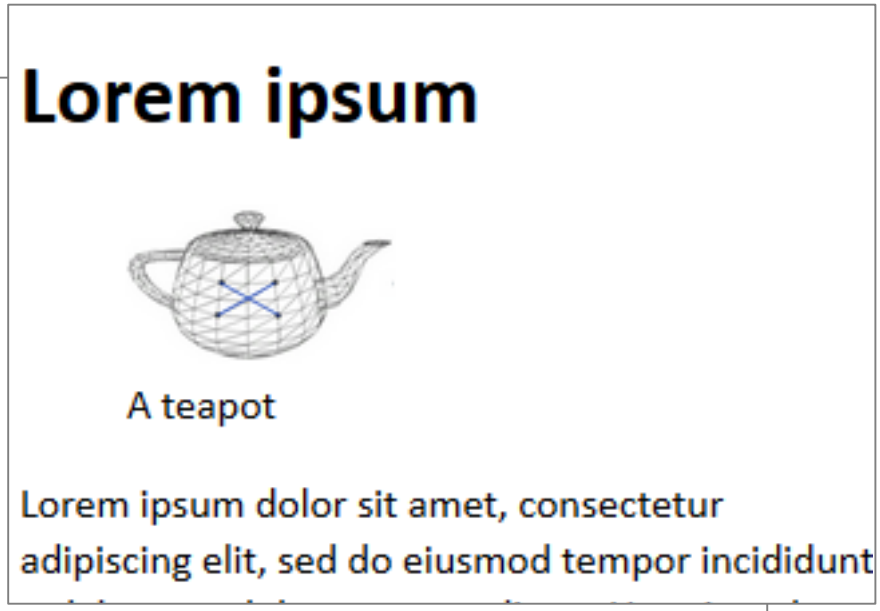
HTML for structure

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8" />
  <title>Lorem ipsum</title>
</head>

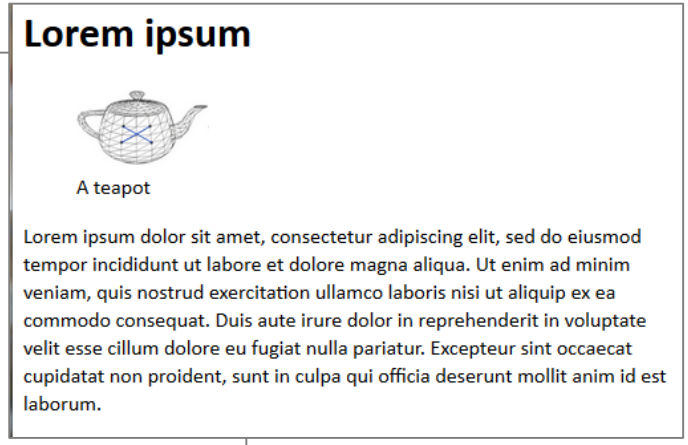
<body>
  <h1>Lorem ipsum</h1>
  <figure>
    <img src="tpot1.jpg" alt="a teapot" width="100" />
    <figcaption>A teapot</figcaption>
  </figure>
  <p>Lorem ipsum…rest of text
  </p>
</body>

</html>
```

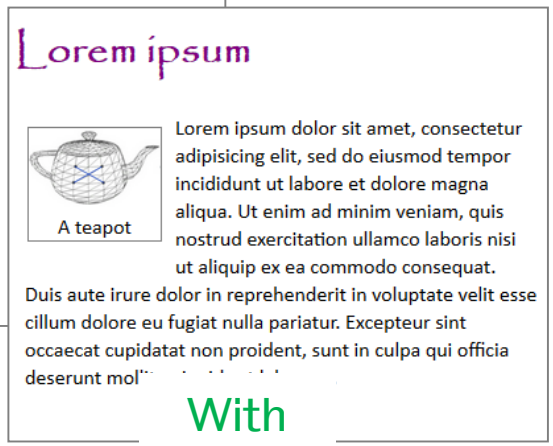


CSS for appearance

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Lorem ipsum</title>
  <link rel="stylesheet" href="x.css" />
</head>
<body>
<h1>Lorem ipsum</h1>
<figure>
  <img src="tpot1.jpg" alt="a teapot" width="100" />
  <figcaption>A teapot</figcaption>
</figure>
<p>
Lorem ipsum…
</p>
</body>
</html>
```



Without



With

HTML for structure

CSS for appearance

index.**html**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Lorem ipsum</title>
  <link rel="stylesheet" href="x.css" />
</head>
<body>
<h1>Lorem ipsum</h1>
<figure>
  <img src="tpot1.jpg" alt="a teapot"
       width="100" />
  <figcaption>A teapot</figcaption>
</figure>
<p>
Lorem ipsum…
</p>
</body>
</html>
```

x.**css**

```
h1 {
  color: purple;
  font-family: Papyrus, serif;
  font-size: 200%;
}

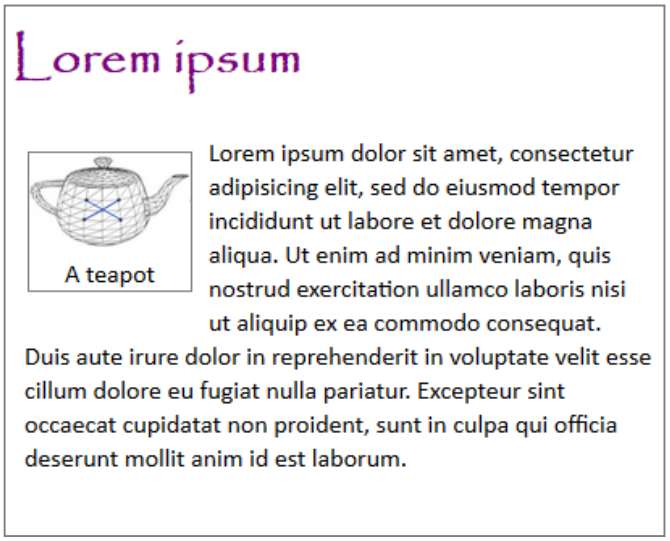
p { margin: 0.5em; }

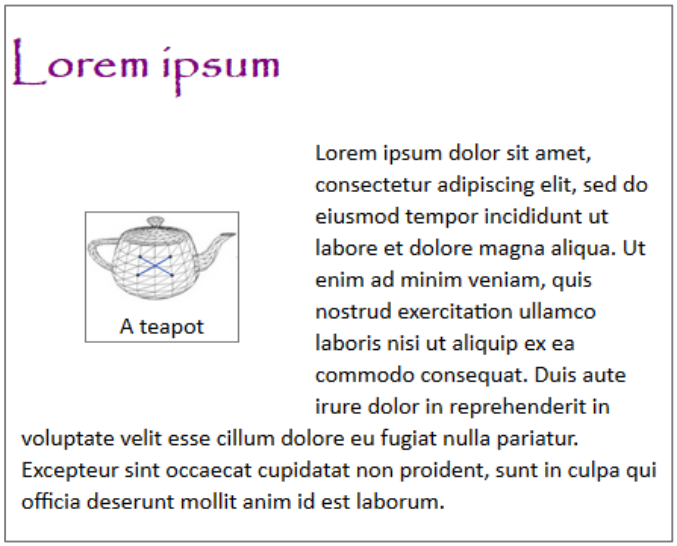
figure {
  float: left;
  border: solid 1px gray;
  margin: 10px;
  padding: 0;
}

figcaption {
  text-align: center;
}
```

10. Spacing

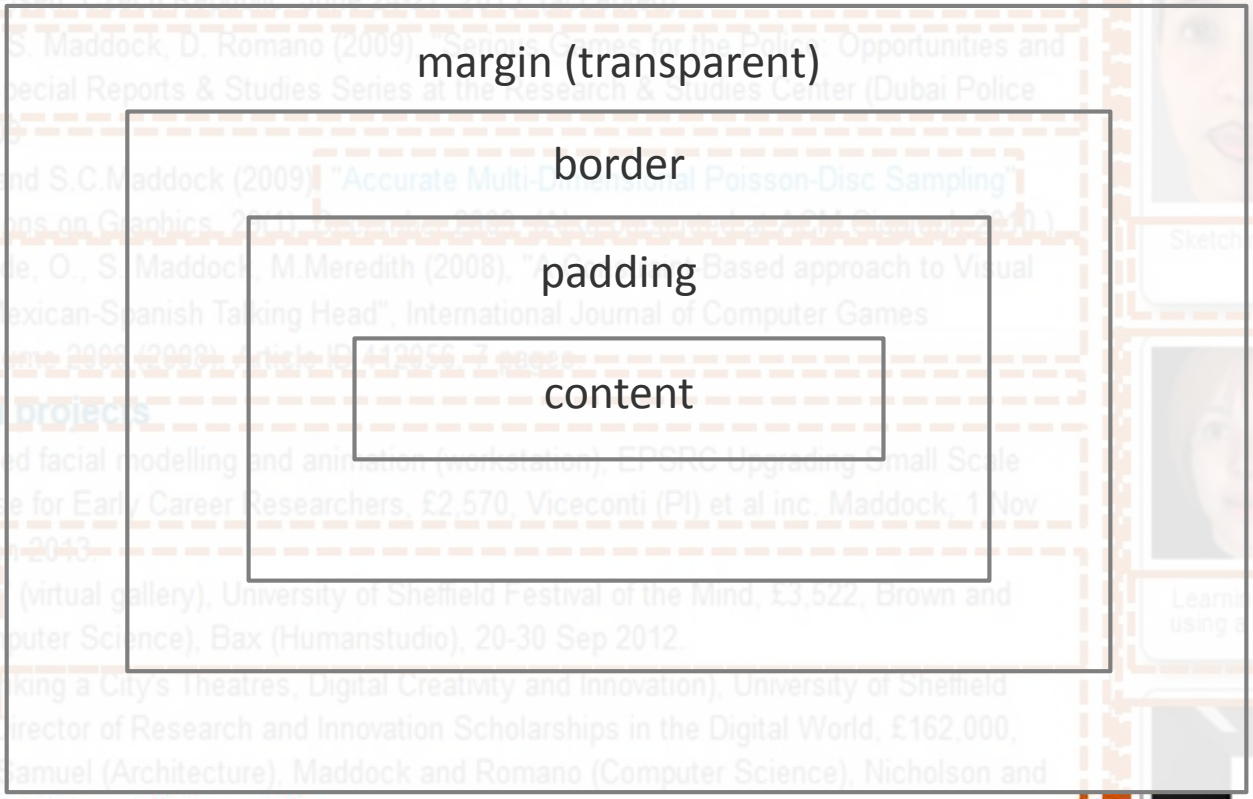
- How do we change the spacing around elements on the page?





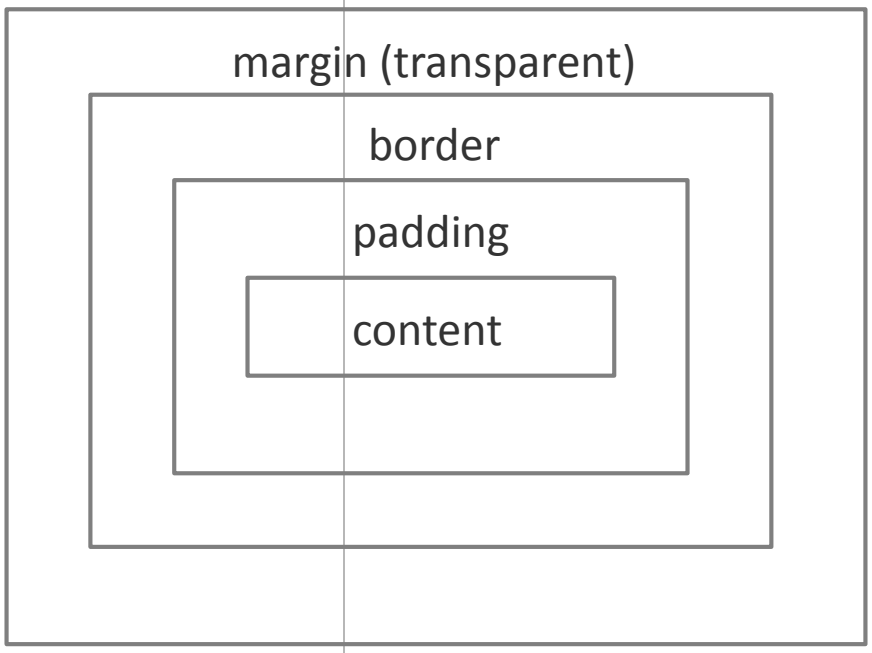
10.1 The Box Model

- Every element (content) is placed inside a box
- Each box region may have a thickness of zero



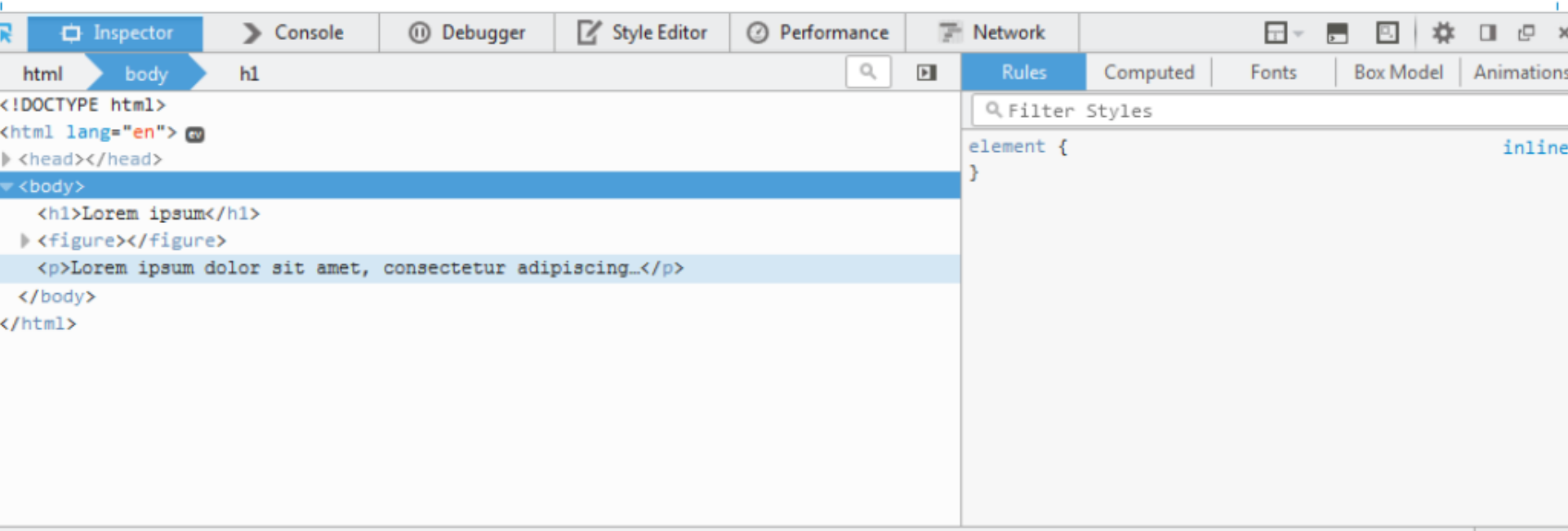
10.2 Example: margins

- Lots of properties for margins, padding and borders
- Example: margins
- margin-top, margin-right, margin-bottom, margin-left
  - *length* | *percentage* | auto
  - `p { margin-top: 10px; }`
- Set all at once with:
  - ```
    body {
      margin: 1em 20px 3% 15px;
    }
    ```


    ```

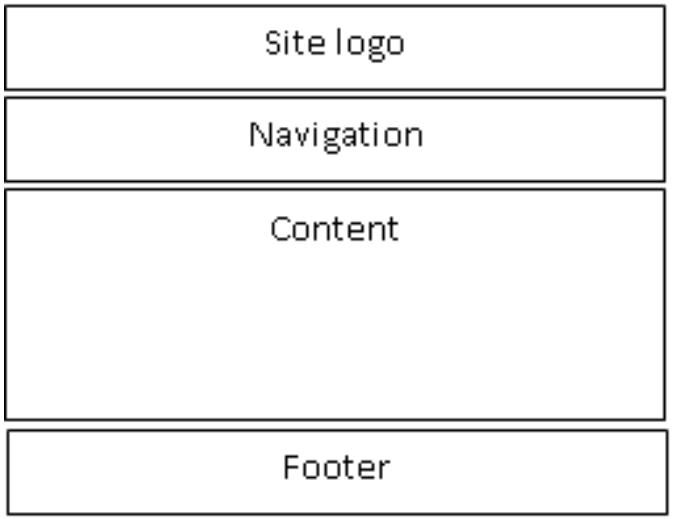
11. Debugging

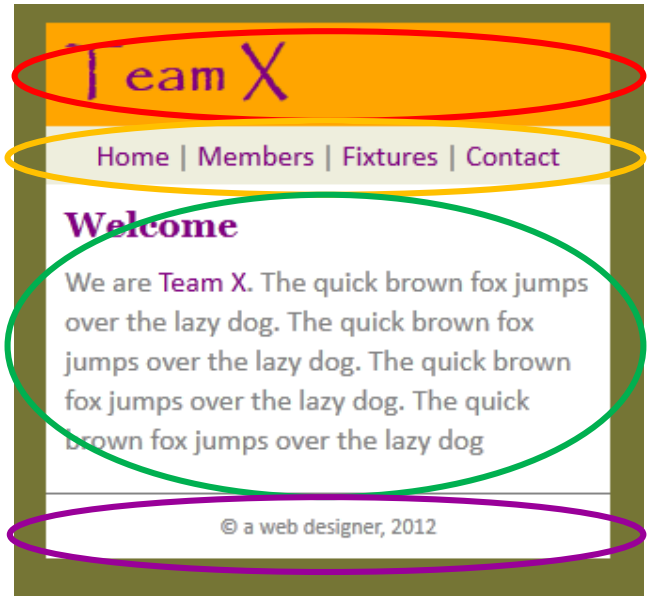
- Example: Firefox Web Developer



12. Grouping elements (Next week)

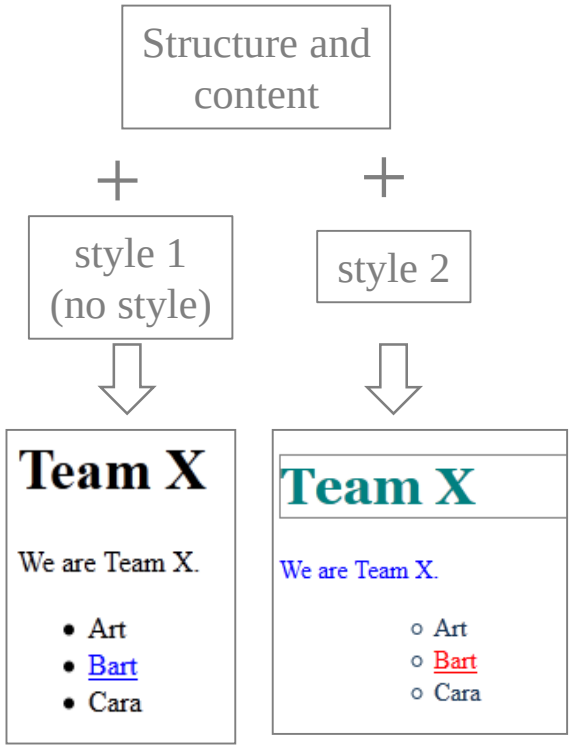
- div
- HTML5 semantic elements: header, nav, main, section, article, footer, aside





13. Summary

- When creating a Web page, separate the structure (HTML) and the appearance (CSS)
- Element 'flow' in documents can be altered
- The box model is used to control spacing around elements
- Grouping HTML elements helps control complexity
- Visualising the document as a tree is useful
- CSS rules can proliferate, so need to organise and work out which apply – cascading, inheritance and specificity



Next week