CSS next steps
Parts 1 and 2

25 & 26 March 2013
Helen Varley Sargan

Preamble
In this course the code referred to will be using html 1.0. Although html5 is up and coming, there is no standard for it yet, and no full agreement on its final form and content, so it isn’t yet fit for purpose.

External style sheets
An external style sheet is a text file with definitions in the same syntax as it used for internal style sheets, but with the <style> tags and comment marks removed. The style sheet needs to have the filename appended by .css - in the form filename.css. For instance the stylesheet for examples from the first part of the course would look like this:

<link href="mystyle.css" rel="stylesheet" type="text/css" media="all">

This information is referred to by any number of web pages instead of having styles in the <head> part of each document, so similar styles in <style> tags of these documents should be removed before an external style sheet is linked to. Any changes in the stylesheet will then occur on all pages that use it. The rel="stylesheet" attribute in the above line indicates this is a persistent stylesheet - to be used all the time.

If you are using a content management system, there is likely to be a number of stylesheets in use. If you are interacting with a single stylesheet to over-ride other styles, it must be the last in the cascade - you may need admin access to influence this.

Internal style information can be used as well as links to external style sheets, in which case it will be the last style to be applied and will over-rule any matching styles that have gone before.

Going beyond the basics
To move into more sophisticated use of CSS, you need to be aware of a small number of essential terms:

• <div>: to use for creating separate sections of the page to give it a structure

• <display>: to use for changing how elements appear, with values of block, inline or none. Any item styled as {display:block} will behave like a block element, with a newline and space before and after, likewise any element styled as {display:inline} will behave like an inline element with no space and newline before and after. Any item styled {display:none} will not appear at all.

There are also some more specialized selectors that you won’t have come across before (see end section).

Managing style sheets
The arrangement of content in a stylesheet makes no difference to the browser reading it; naturally the syntax is important, and it should be easy for you to follow what the styles are for. Comments can and should be added to annotate the stylesheet to help you remember what styles were set up for: enclose the note between /* and */. Do not add too many returns, especially to add whitespace as it will increase the size of the stylesheet.
Avoiding ‘divitis’ and ‘classitis’

Each div you create adds another layer of complexity to your web page, and every extra class is another ‘special case’ in your code and another entry to a style sheet. Once you have created a div or, say, a table on a page and given it a class or id (class if you want to use it more than once on a page, id if it is to be unique), you can refer to all the elements within that div or table by their nesting within it, rather than give them classes of their own - this is sometimes known as contextual styling, and you are defining ‘descendent combinators’ (see later for other combinators).

As an example, for the heading on the Computing Service templates, the coding is like this:

```html
<div id="dept-title">
  <h1>Computing Service</h1>
</div>
```

and the stylesheet info looks like this:

```css
/*For text to be printed but not appear on screen*/
.noshow, .noshowsmall, div.docbox {display: none;}
```

This is a very small example, but the principal can be applied to any parts of the page, and reduce the addition of classes into the html.

When using contextual styling some classes might have to be added as well. You can also group selectors, by separating them with commas, so that more than one set of mark-up will use the same style, for instance (showing a comment to note what the style is for):

```css
/*For text to be printed but not appear on screen*/
.noshow, .noshowsmall, div.docbox {display: none;}
```
Part 1: Boxes in boxes

An html web page works as a sequence of boxes, threading together and sometimes sitting inside one another, from the top to the bottom of the page of code. The position of the box in the code doesn’t necessarily reflect its position on the visible page, since it can be styled to appear elsewhere. To visualise this, you can use the Firefox web developers toolbar to outline all the block elements on the page (block elements are blocks of text with selectors such as <body>, <h1>, <p>, <ul>, <li>, and any <div>).

The box around any of these elements is the boundary of the area that any style can modify. In addition, any section of text that is enclosed by a span (which is inline rather than block) has a box round it that can be modified. You can see that in many parts of a page you have boxes within boxes, any of which can be modified by styles and many of which will inherit styles from enclosing boxes. Read through the tutorial at http://css.maxdesign.com.au/selectutorial/document_tree.htm to understand how style is inherited through a web page.
For example, the following page illustrates how font family, background, font colour and size are inherited:

The styles are like this:

```html
body {
  font-family: "Trebuchet MS";
  font-size: 120%;
  color: #333;
}

#box1 {
  float:right;
  width: 35%;
  clear:both;
  background: #C99;
  margin: 0 0 1em 1em;
  padding: 0.5em;
}

#insidebox1 {
  font-size: 70%;
  font-style:italic;
  line-height: 90%;
}

#box2 {
  float:left;
  width: 20%;
  clear:both;
  font-family: "Copperplate";
  color: #c30;
  line-height: 120%;
  border: 1px dotted #999;
  margin: 1em 1em 0 0;
  padding: 0.5em;
}

#insidebox2 {
  color: #009;
}
```

The page appearance is as below. If a characteristic has not been set it will be inherited from the browser’s default settings; if it hasn’t been reset by a subsequent style, then it will be inherited from its parent.

---

**Introduction**


---

**BOX1**

Lorem ipsum dolor sit amet, consequetuer adipiscing elit, nunc aliquet facilisis turpis. Mauris non nibh non ligula elementum pretium.

---

**INSIDEBOX1**

Maecenas cursus. Nunc guia lorem vitae risus dapibus dapibus.

---

**BOX2**

Lorem ipsum dolor sit amet, consequetuer adipiscing elit, nunc aliquet facilisis turpis. Mauris non nibh non ligula elementum pretium.

---

**INSIDEBOX2**

Maecenas cursus. Nunc guia lorem vitae risus dapibus dapibus.
Types of box

- Block level elements and block boxes (such as `<body>`, `<h1>`, `<p>`) - elements produce at least one box. Some (like lists) produce more than one, as `<ul>` and `<li>` elements each have their own.
- Inline level elements and inline boxes - content is distributed in lines within a block. Examples include links `<a ..>`, `<span>`, `<em>`.
- Run-in boxes - joins box to following box. This is not standard behavior for any element.

The `display` property - specifies a box's type when you want to change it from normal behaviour.

How boxes work

The box model from the CSS2 standard looks like this:

![Box Model Diagram]

If a content box is set to `p {width:200px; border: 20px; padding: 20px}`, the correct method of calculating the overall box width is: content (200px) + padding (20px+20px) + borders (20px+20px) = 280px.

The box model as described above is currently how elements work.

Examples that follow later in the course will show how the box model is used with styling.

Positioning information with CSS

When approaching how to position information on a web page, there is never one right solution for how to do it. There are many different ways, with pros and cons.

The first thing to consider is how to organize your page - create a grid as a guide for where content is to go - this is a very simple example:
Use this to create <div>s to section up your html into the positional blocks you want to use, naming them appropriately.

The second thing is to think about the type of layout that you wish to achieve, which can be either fixed or not:

- **Not fixed**
  - **Liquid or fluid.** Widths of some or all parts of the page are specified in percentages or not specified at all, so that the width of the page will change with the width (and height) of the window. Design can be more difficult and those with large screens can have uncomfortably long lines. It is possible to use a fluid layout with maximum and minimum widths, as on the current University templates or the template available at http://css-tricks.com/examples/PerfectFluidWidthLayout/ (which controls the width in a slightly different way).

- **Elastic.** An elastic layout is of fixed size but units used are relative (using ems), so that if a user enlarges or reduces the size of the page the layout will expand or contract. With a very large font size the page may become unmanageable.

- **Responsive.** Responsive design is the recent aim for many as it accommodates devices of all sizes by incorporating rules that define the layout relative to the width of the device window (known as the viewport). One of the best examples is http://www.bostonglobe.com/ but the new University templates are also responsive (for instance see http://www.cam.ac.uk if you change the window width, there are ‘break points’ where the layout changes. There is no need for device detection and other versions of the site.

- **Fixed.** Is of a specified width, often centered within a window so it looks OK with any width exceeding itself. Appeals to designers as layout is firmly defined, but can waste available screen space, or cut-off content from those with small window. Enlarging type may cause severe disruption to the page design, depending on browser. PDAs and mobiles sometimes deal with fixed widths by linearizing content, or such sites might have device detection to serve a mobile/tablet version. See http://www.telegraph.co.uk/ as an example.

To get a better understanding of flexible web design and where responsive design fits in, a good start is a book like ‘Bullet-proof Web Design (3rd edition)’ – see http://www.peachpit.com/store/product.aspx?isbn=0321808355

**Approaches to positioning content**

1. **Using floats**

   Floats are a simple way of placing content right or left and are not to be confused with true css positioning (see next section) - they are often all you need. You can use float creatively to float sections of the page right or left, or you can float a number of objects right or left, and they will butt up against each other (see http://css.maxdesign.com.au/floatutorial/ for examples) - float is very versatile. When a single object is floated, the surrounding text will flow around the floated item - when you want this flowing to stop, you need to stop the float by adding { clear:both; } and then the following content will start underneath.

   Float is used extensively in many web page templates.
1. For a simple example, floating a picture to the side of the page:

```
Welcome to the University Offices

Unified Administrative Service
The purpose of the UAS:
To support and enable the University’s mission to contribute to society through the pursuit of education, learning and research at the highest international levels of excellence.

The values of the UAS:
- We deliver an effective and high-quality service.
- We collaborate and work in partnership.
- We are open, responsive and innovative.
- We respect others and value diversity.
- We support, recognise and develop our staff.

<p><img alt="The old schools" src="images/old-schools-200.jpg"></p>
```

This is achieved by the following code. Remember that `p img` means ‘any image within a paragraph’- the box around the image is being adjusted by the styling, and the nature of the image is being adjusted to ‘inline’ so that the text flows around it.

**CSS CODE**

```css
p img {
    float: right;
    border: 5px solid #dbd7cc;
    margin-left: 20px;
    margin-bottom: 1.125em;
    display: inline;
    clear: right;
}
```

**HTML CODE**

```html
<h1>Welcome to the University Offices</h1>
<p><img alt="The old schools" src="images/old-schools-200.jpg"></p>
<h2>Unified Administrative Service</h2>
<h3>The purpose of the UAS: </h3>
<p>To support and enable the University’s mission to contribute to society through the pursuit of education, learning and research at the highest international levels of excellence.</p>
```

2. For an example of using floats with a styled list, to create ‘buttons’:

```
There are likely to be changes in water availability and sea level for the United Kingdom including the impact of extreme events such as storm surges. The reliability of the predictions, the extent to which they are consistent with changes to date, and the likely range of uncertainty in the future is another important issue, as is progress in quantifying uncertainty for use in risk assessment and decision making.

Prof John Mitchell is Chief Scientist at the Met Office, Exeter
```

The list is being restyled so that is has no fresh lines or preceding characters, and the list items are being floated one to the right and one to the left. The links in the list items are being styled so that the link itself forms a visible box of a standard size and clicking anywhere in the box activates the link. The styling creates a different colour background when the mouse is hovered over the link box.
3. A simple example of getting a floated left-hand navigation. The list for the left hand navigation is enclosed in a section <div id="nav">, which is given a set width and floated left. The list itself has the styling removed, some padding added and the text made slightly smaller. The area to the right of this is in <div id="content">, which is given a margin to the left that is wide enough for the navigation to slot into it, a little spacing around the text is added, along with a different background colour.

```html
<html>
<head>
  <title>simple navigation</title>
  <style>
    html, body {color: #000;font-family: "Trebuchet MS", Arial, sans-serif; font-weight: normal;font-size: 0.9em;background:#ffc; margin:0;}
    h1 {color: #600; font-size: 140%;text-align: center; }
    h2 {color: #600; font-size: 120%; font-variant: small-caps; }
    #branding {background:#fc0; width: 100%;padding top:0.5em; border-bottom:6px solid #f90;}
    #nav {float:left; width: 9em; padding: 0.5em 0 0 0.5em; margin:0;}
    #nav ul {margin:0 ; padding: 0 1em 1em 0; text-align:left;list-style: none; font-size:90%;}
    #nav ul li {padding: 2px 0 2px 0;}
    #nav h2 {padding: 0; font-size:90%; color: #006;}
    #content {margin: 0 0 0 9em; padding: 0.5em 0 0 0.5em;border-left:1px dotted #006; background: #fff;}
    #footer {clear: both; background:#ff0; width: 100%;text-align: center;border-top:6px solid #f90;padding-bottom: 0.5em;font-size:0.9em;}
  </style>
</head>
<body>
  <div id="branding">
    <h1>Branding</h1>
  </div>
  <div id="nav">
    <h2>Navigation</h2>
    <ul>
      <li>ffgf</li>
      <li>vvbb</li>
      <li>hgmmhb</li>
      <li>ythhh</li>
      <li>yyuuj</li>
      <li>k,k,,k,)</li>
    </ul>
  </div>
  <div id="content">
    <h2>Content</h2>
    <p>Lorem ipsum eos omnes dolorum liberavisse te, no sed putant euripidis, et vide aliquid vel. Pro no mutat ridens. Eum ad nostrud tractatos cotidieque, veritus mentitum euripidis sea eu. Idque fabulas eam at, cu quot homero incorrupte mea. Vix sale volumus elaboraret no.</p>
    <p>Est mucius habemus incorrupte no, te pro utinam inciderint. Civibus omissam</p>
  </div>
  <div id="footer">
    <p>next steps</p>
  </div>
</body>
</html>

clear:both; in the style for the <div id="footer"> clears the float and returns the alignment to normal.

4. A slightly different take on the same floated left-hand navigation:

In this example, the background colour is set for the whole of the page and it is centred with an equal margin on either side. This means that you can see the background colour for the page around the page content, but the background colour for the navigation is that set for the body. The rest of the page is similar to the coding used above.

**CSS Code**

```css
html, body {color: #000;font-family: "Trebuchet MS", Arial, sans-serif; font-weight: normal;font-size: 0.9em; background:#ffe; margin:0;}
h1 {color: #600; font-size: 140%;text-align: center; }
h2 {color: #600; font-size: 120%; font-variant: small-caps; }
#page {background:#ccc; width: 90%;margin: auto;}
#branding {background:#fc0; width:100%;padding-top:0.5em; border-bottom:6px solid #f90;}
#nav {float:left; width: 9em; padding:0.5em 0 0.5em 0; margin:0;}
#nav ul {margin:0 ; padding:0 1em 1em 0; text-align:left;list-style: none;font-size:90%;}
#nav ul li {padding: 2px 0 2px 0;}
#nav h2 {padding: 0; font-size:90%;color: #006;}
#content {margin: 0 0 0 9em; padding:0.5em 0 0.5em 0;border-left:1px dotted #006; background: #fff;}

#footer {clear: both; background:#ffe0; width: 100%;text-align: center;border-top:6px solid #f90;padding-bottom: 0.5em;font-size:0.9em;}
```

**HTML Code**

```html
<div id="page">
  <div id="branding">
    <h1>Branding</h1>
  </div>
  <div id="nav">
    <h2>Navigation</h2>
    <ul>
      <li>ffgf</li>
      <li>vvbb</li>
      <li>hgmmhb</li>
      <li>ythhh</li>
      <li>yuuju</li>
      <li>k,k,,k, </li>
    </ul>
  </div>
  <div id="content">
    <h2>Content</h2>
    <p>Lorem ipsum eos omnes dolorum liberavisse te, no sed putant euripidis, et vide aliquid vel. Pro no mutat ridens. Eum ad nostrud tractatos cotidieque, veritus mentitum euripidis sea eu. Idque fabulas eam at, cu quot homero incorrupte mea. Vix sale volumus elaboraret no. </p>
    <p>Est mucius habemus incorrupte no, te pro utinam inciderint. Civibus omittam eam at. Ut ius puto graecis. Ius id elit virtute iuvaret. </p>
  </div>
  <div id="footer">
    <p>We are big and we are clever</p>
  </div>
</div>
```
If, in the above example, the background colour for the navigation was set on only that section, the colour would only extend down the column until the bottom of the navigation list rather than to the bottom of the column, like this:

When using floats you must be careful about clearing them after use so that text assumes its normal position again - see http://www.brainjar.com/css/positioning/default3.asp for a good description and demonstration.

2. **CSS Positioning**

CSS positioning (or CSS-P) was the CSS2 successor to using tables for layout purposes. At its inception it had the problem of not being 'backwards compatible' with old browsers. It allows you to position areas of your windows relative to each other or with absolute definition (or both), but has the multiple drawbacks of being complex to understand and tricky to use and as a result it hasn’t been taken up as readily as using floats. If you find that floats don’t do what you need, to begin to understand how positioning works, it might be useful to look at:

- Section on positioning from the HTMLdog website - http://www.htmldog.com/guides/cssadvanced/layout/ and an article at http://www.htmldog.com/articles/elasticdesign/
- Three articles at http://www.vision.to/articles.php

In addition these rules and methods don’t work in some browsers, even the newest (see 'position' in http://www.quirksmode.org/css/contents.html)

**The answer for how to do positioning**

There is no answer. There are a number of different ways to use positioning, but often problems are thrown up by the simplest of needs. Simple positioning is achieved by using fixed widths, usually set into a background frame, but it is usually better to explore using floats first rather than getting into anything more complicated.

Since IE7, IE8, and IE9 are now in general use and IE10 upcoming, you need to take note whether websites giving advice have been updated recently. Many that have been hanging around for a while may now be giving spurious advice about what IE can do. See:

Day 2

Part 2: Doing more things with CSS

A perspective

For a typical measurement, in March 2013 (March 2012) on http://www.cam.ac.uk the breakdown is:

- Chrome (versions too difficult to untangle) 29.33% (23.21%)
- Safari (versions too difficult to untangle) 21.22% (17.78%)
- Firefox (versions too difficult to untangle) 19.60% (22.69%)
- Internet Explorer 9 11.06% (10.18%)
- Internet Explorer 8 10.92% (17.33%)
- Internet Explorer 7 2.19% (4.45%)

You can see that the popularity rating has changed quite a lot over a year but perhaps not quite as you would expect. About 11.7% (5.30%) of visits are from Safari users on iPhones, iPads or iPods. Over 85% of the users of Chrome are Windows users.

By now IE6 is not something you need to worry about, unless there is an app-related tie to it in your community. This is a great relief, as in a more modern climate for CSS and html it is really difficult to support. http://msdn.microsoft.com/en-us/library/hh781508(v=vs.85).aspx shows CSS supported in version of IE from 6 to 10.

Things you can do with lists

If a number of items on the page looks like a list, it ought to be marked up as a list so that users of speaking browsers are clear what sort of content it is. Lists are also the best way to structure navigation. Although the techniques shown here are all in the context of lists, they can equally be applied on other elements or in other circumstances.

See http://css.maxdesign.com.au/ for a seriously large number of things you can do with lists, only some of which are covered here, plus tutorials on floats and selectors. One of the reasons that lists are so malleable with CSS is that by default a list is a set of boxes within another box - there are a number of different places to control functionality and appearance, and also the <display> attribute can really come into its own.

You can use stylesheet formatting to alter a standard list to use a graphic instead of a plain bullet, formatted into buttons or run in a paragraph. In all these examples the formatting of the list is exactly the same, with a named div enclosing the list to allow use in the stylesheet of contextual markup. In some cases list items have been allocated a class to add some extra formatting.

Taking a simple list, as follows:

```
<p>Main ingredients:</p>
<ul>
<li>onions</li>
<li>carrots</li>
<li>venison</li>
<li>beetroot but only with the tops still attached</li>
</ul>
```

1. Replacing bullets with graphics

Adding a different graphic instead of a bullet can be done two different ways, either by using the list styling to do it or by using a background image to do it. The problem with the first method is that browsers vary in how they position the image and may raise it above the text by an unacceptable amount, so the second is more visually reliable. (In the examples below are both inheriting some list styling, but they are both inheriting to the same extent.)
Adding a graphic directly the code is

\[
\text{lists #first2 ul li \{}
\text{list-style: url(http://www.cam.ac.uk/icons/burst.gif) disc;}
\text{background-image: none;}
\text{padding-left: 0.5em;}}
\]

‘disc’ is added to the list-style as a fallback should the graphic not be available.

and adding the graphic as a background image the code is

\[
\text{lists #first3 ul li \{}
\text{background: transparent url(http://www.cam.ac.uk/icons/burst.gif) no-repeat 0px 5px;}}
\]

In both of these techniques you will have to adjust the padding and positioning to get the appropriate space between the graphic and the start of the text.

2. Run-in list

A list can be run in with the preceding text and punctuated – the preceding paragraph and the list have to be enclosed in a <div> with an id (in this case <div id="inlinelist">) so that the styling can be targeted at them. If it is to be offset from the surrounding text that div can be styled (as in the example). Below, the last list item has a class (called last) attached and the attributes ‘li:after’ and ‘li.last:after’ are used to add the punctuation (they are what’s called pseudo elements, and will be explained later).

The style for this example is

\[
\text{#inlinelist \{border: 1px solid #000;width: 50%;padding: 1em; font-family:}
\text{"Trebuchet MS", sans serif;}}
\]
\[
\text{#inlinelist p \{display: inline;}}
\]
\[
\text{#inlinelist ul, #inlinelist li \{display: inline; margin: 0; padding: 0; color:
\text{#333; font-weight: bold;}}
\]
\[
\text{#inlinelist ul li:after \{content: ", \}}
\]
\[
\text{#inlinelist ul li.last:after \{content: ". \}}
\]

3. Simple nav bar list

Cutting down the final entry for ease of use, we can produce a coloured navigation bar from it. Using what are called adjacent sibling selectors (which work in IE from version 7 upwards) and a simple list with an id. The black bars between the coloured boxes comes from the background colour of the list. (Some of the styling is being inherited and isn’t shown here.)

The style to produce this is:

\[
\text{ul#navlistbest \{margin-left: 0;margin-bottom: 25px;padding: 3px 0; display:}
\text{inline; background:#000;}}
\]
\[
\text{ul#navlistbest li \{margin: 0;padding: 3px 15px; list-style: none;}
\text{width: 40px;display: inline;}}
\]
\[
\text{ul#navlistbest li \{background:#ffff99;}}
\]
For a version that will work in all browsers, the whole list is contained in a `<div>` (in this case it gives a black background) and each list item has a class specifying the colour. (Some of the styling is being inherited and isn’t shown here.)

The div formatting does this:

```
#navlistbetter {padding: 5px;margin-bottom: 25px;background:#000;}
#navlistbetter ul {margin-left: 0;padding-left: 0;display: inline;}
#navlistbetter ul li {margin-left: 0;padding: 3px 15px;list-style: none;display: inline;width: 40px;}
#navlistbetter ul li.first {background:#ffff99;}
#navlistbetter ul li.beet {background:#CC0033;}
#navlistbetter ul li.carr {background:#FF9933;}
#navlistbetter ul li.ven {background:#CC6600;}
```

This approach can be adapted in a number of ways (and there are several other ways of doing this), for instance by styling the link, you can have a ‘live’ link colour (and a hover colour) and all the other tabs the same colour, so the navigation flags where the user is in the content. Examples are

- [http://www.study.cam.ac.uk/undergraduate/](http://www.study.cam.ac.uk/undergraduate/) (in this case a live link colour and matching colour for sub-navigation)
- [http://www.ucs.cam.ac.uk/az](http://www.ucs.cam.ac.uk/az)

4. Rollover nav bar list

- The list is contained in a div that sets the width, the initial background colour and text colour, the font and details of padding, margin and right border.
- Within the list items, the borders that create the bottom rules between the buttons are set.
- The buttons themselves are set on the links - all links are defined as block with decoration turned off (so no underlining), and the colour details of the background and on the right and left of the buttons are set.
- The hover form of a link changes the colour detail when the link is used.

by adding a div and the following:

```
#buttonlist {float:left;width: 9em;border-right: 1px solid #000;padding: 0 0 0.5em 0;margin-bottom: 0.5em;font-family: 'Trebuchet MS', 'Lucida Grande', Verdana, Lucida, Geneva, Helvetica, Arial, sans-serif;background-color: #66CCFF;color: #333;}
#buttonlist ul {list-style: none;margin: 0;padding: 0;border: none;}
#buttonlist li {border-bottom: 1px solid #66CCFF;margin: 0;}
#buttonlist li a {display: block;padding: 5px 0px 5px 0.5em;border-left: 10px solid #000000;border-right: 10px solid #508fc4;background-color: #2175bc;color: #fff;text-decoration: none;}
#buttonlist li a:hover {border-left: 10px solid #1c64d1;border-right: 10px solid #5ba3e0;background-color: #2586d7;color: #fff;}
```
Lists can also be styled into horizontal navigation with drop-down menus and/or flyouts. Examples can be seen at:

- http://webdesignerwall.com/tutorials/css3-dropdown-menu (more explanation of this later)
- http://css3menu.com/index.html (a downloadable pack that builds styles for you)

Most methods require widths to be set and some require additional use of javascript.

Selectors and how to use CSS

The standard for CSS2 was published in 1998 with version 2.1 published in June 2011 - 2.1 removes poorly-supported or not fully interoperable features and adds already-implemented browser extensions to the specification. Work has been progressing on CSS3, which is a modular specification, since 1999. By March 2013 there were several recommendations, with more as working drafts (see http://www.w3.org/Style/CSS/current-work). CSS3 has introduced several advances that are already well used, in particular media queries, as a way of producing responsive pages; backgrounds and borders, which allow CSS-only rounded corners and gradient backgrounds; and the selectors that CSS3 works with.

Selectors are the structures that you can use for defining styles - they can be as straightforward as h1 or div.name, or much more conceptual, some of which are mentioned in the following text. There are a variety of ways you can use selectors and a number of special selectors that you can use, however, it is likely that many are not yet supported by enough browsers to make them usable. A good comparison table may be found at http://www.quirksmode.org/css/contents.html and a more complex one at http://en.wikipedia.org/wiki/Comparison_of_layout_engines_(Cascading_Style_Sheets)

As an introduction, there is a good tutorial on selectors at http://css.maxdesign.com.au/selectutorial/index.htm

Examples so far

So far you have seen how to style selectors such as paragraphs, headings and divs that you add. You have seen how to use the “display” attribute with selectors to change the default way they work, so that you can change links into buttons, and lists into continuous text.

Adjacent sibling selectors and Child combinator

This is potentially an extremely useful method of adding a style specific to the order of elements, for instance in a previous example of styling a list as navigation, or if an h3 heading is preceded by an h2 heading, you might want to reduce the space above. Adjacent sibling selectors are supported by IE7 and above and other modern browsers. Style information is as shown below:

h2+h3 {margin-top: 0.5em;}

A child combinator can be useful to define a style for a child of a particular element, for instance h2 > p defines a paragraph only if it has been preceded by an h2 heading, not any other paragraph following that. Child combinators can be mixed with descendant combinators to give, say h2 > p > ul, which will refer to only a ul that has been preceded by p that followed an h2.

Using Pseudo elements

In CSS2.1 the pseudo elements are the selectors :first-line and :first-letter and :after and :before. They all work in newer browsers (including IE8 and above) but only the first two will work in IE7 and below. :first-letter can be used quite successfully as a drop cap - see http://www.javascriptkit.com/howto/pseudoletter.shtml for example.

The example above for running a list together with a paragraph shows how to use :after.

CSS3 pseudo selectors

In CSS3 there are additional pseudo selectors, such as

- Nth-child (not below IE9)

This pseudo-class will allow you to easily add alternate row tint to tables, by using the style:
tr:nth-child(odd) td { background-color: #86B486; }

'odd' and 'even' are the common keywords used here. ‘2n’ is equivalent to ‘odd’. You can also use ‘3n’ for every third row, and so on.

- last-child (not below IE9)

The ‘last-child’ selector can be used, say, for the last item in a list, to make it format differently without having to add a class. This would have been useful for the list example using :after.

CSS support for printing (CSS2 and CSS3) markup is incomplete, although there is more support in newer browsers. (More info on combinators and pseudoclass selectors - http://www.vanseodesign.com/css/combinators-pseudo-classes/)

**Other CSS3 selectors**

Some CSS selectors are decorative (like adding curves, colour gradients, text shadowing, which will mean fewer graphics are needed on web pages), others much more functional like media queries. At present IE9, Firefox, Opera, Safari and Chrome are likely to display CSS3 effects but the table at http://www.quirksmode.org/css/contents.html will give you a better steer.

**border-radius**

This explanation http://www.css3.info/preview/rounded-border/ is one of the best.

**border-shadow**

This explanation http://www.css3.info/preview/box-shadow/ is a good one.

The example of dropdown menus at http://www.webdesignerwall.com/demo/css3-dropdown-menu/ uses border-radius to give the rounded corners and box-shadow to give the shadow effect, as well as :first-child and :last-child to style the lists.

Overviews of some CSS3 effects may be seen at:


**Producing slides**

There are several tools you can use to produce slides, using xhtml, css and javascript - s5 (http://meyerweb.com/eric/tools/s5/) and slidy (http://www.w3.org/Talks/Tools/Slidy2/) are the most well known. Both of these tools work with a single xhtml file and need a knowledge of xhtml and css to make the presentation work properly and look professional. s5 is said to have better documentation.

There are a couple of alternatives that may be worth looking at if you are familiar with ruby (s9 - http://slideshow.rubyforge.org/) or ajax (Ajax-S - http://www.robertnyman.com/ajax-s/).
Part 3: The mechanics of style sheets

Media-dependent style sheets

The ‘media’ attribute can be used if you want a different stylesheets to be used for different presentation, for instance printing, in which case in the <head> part of the document you might have:

```html
<link href="mystyle.css" rel="stylesheet" type="text/css" media="all"/>
<link href="mystyleprint.css" rel="stylesheet" type="text/css" media="print"/>
```

The available media types are print, screen, braille, aural, TV, handheld, projection. Screen and print are mainstream; handheld, projection, braille and aural have some support but cannot be relied upon. See http://www.w3.org/TR/REC-CSS2/media.html for full information. Don't confuse this with media queries in CSS3.

You can use a media stylesheet for printing to change the look of or remove information from the screen that the user won’t need, and add information that they will. If minimal changes are needed then you can use an @media rule in the style information in the head of a document (see later). If used carefully, you may be able to bypass having to create pdf versions of pages. Since (x)html is much more accessible this would be very advantageous. A good outline for use of media-dependent stylesheets, with links to information about designing for each media type may be read at http://css-discuss.incutio.com/?page=MediaStylesheets, and another good (and relatively up to date) source of information is http://www.howtocreate.co.uk/tutorials/css/mediatypes

Handheld devices (smartphones and tablets)

Mobile phones and smartphones use mobile-specific browsers, which may manage rendering and use of stylesheets in a different way from standard browsers. If you look in your website logs of what operating systems connections are being made from, you will see iPhone, iPad, Android, Blackberry, Symbian OS, and possibly other less-used systems too.

The iPad and iPhone users will be connecting with a modified version of Safari, which with a standard website gives a miniature but enlarger version of the page. If you sign up as an iPad/iPhone developer (https://developer.apple.com/ipad/sdk/), you can download a kit that includes an iPhone emulator for desk-top use. iPhone does not support the handheld.css stylesheet. The number of Android devices is growing more rapidly than any other device at the moment - by default android devices use a Webkit-based browser (logged as ‘Android Browser’) but may also use various other mobile-specific browsers if the user wants to install them. There is a developer kit for Android (http://developer.android.com/sdk/index.html) but it is much more arduous to install an emulator than it is for Apple products.

Opera mini has been used by Symbian OS devices but these are little used now and its use has declined, which is a shame as it is a very good browser.

Responsive design

There are some good articles about handheld devices at http://alistapart.com/topic/mobile-multidevice. CSS3 makes it possible to add CSS media queries, so that, for instance a style rule can be passed to a handheld device on the basis of screen width. Techniques needed to do this effectively have advanced a lot in the last year but some are still a work in progress but are being developed rapidly (see http://www.smashingmagazine.com/responsive-web-design-guidelines-tutorials/ for an overview from last year, but some of this will have been superseded).

The new University templates are created as a responsive design - see http://www.ucs.cam.ac.uk and change the window width to see what happens.

Print stylesheets

How to introduce a print stylesheet

- Look carefully at your screen page and decide which parts of it you want to remove and what you need to change for it to be more useful printed.
- the paper version should be branded, dated, and have on it contact information;
• remove navigation aids - this may leave white space but if you are using stylesheets effectively you ought to be able to avoid this;

• add extra information - if there are many links then it’s probably best to underline link text, since it may help readers, and the url of the link needs to be given explicitly (if it isn’t the link text).

• Decide on a strategy. Depending on how your page is marked up you may either need to introduce additional styles (usually by adding <div>s) or select existing styles that may be defined differently.

• Ensure your general-purpose stylesheet is labeled as media="all". It may also need editing, depending on how you add extra information (to ensure information that is only for print doesn’t appear on the screen). Create a new print stylesheet to hide or change items you wish to appear differently on the printed version.

• Edit your page to add any extra styles and the link to the print stylesheet.

• You can review the results of the stylesheet by going to Print Preview in your browser

• If you add new styles to your general stylesheets remember you may need to add them to the print stylesheet as well.

Print stylesheet in action
To add or remove areas on either the printed or screen version, use styles called something like .noshow and .noprint, as they are very explicit and easy to remember. These are both defined in the respective stylesheets as {display: none;}; the first in the all media stylesheet and the second in the print stylesheet. Different parts of the page, either tables or sections marked as <div> have been given ids to identify them, so, for instance the printing of the global navigation bar can be ‘turned off’ by styling:

```plaintext
div#navigation {display: none;}
```

You should remove the global navigation and the breadcrumb trail navigation - each of the sections is a <div> that has an id, which is referenced in the stylesheet as above. If they were not defined in <div> but a table, you would add an id to a table that could be referenced by a style. We also removed the images from the list to make neat printing in black and white.

In the print stylesheet a different typeface can be defined, to allow easier reading, and add some extra information, giving the url of the document and for further information, as well as printing urls of links if they are not already used as link text. This information is added with enclosing spans or divs styled to allow them to be hidden on-screen.

To print URLs of links in the document: For cross-browser compatibility, put the URL in a span that only shows up when the document is printed. This requires forethought and isn’t really recommended as a sustainable solution. In the longer run you can use the :after pseudo class (once almost all browsers in use support it).

What you can’t do with a print stylesheet
• Determine where ‘pages’ start and end. There are CSS2/CSS3 print controls (a new working draft at http://www.w3.org/TR/2013/WD-css3-page-20130314/) but as the controls are not cross-browser compatible or well supported, they are useless.

• Therefore no reliable headers and footers or page numbering is possible.

There are open-source and third-party applications (for instance PDFreactor [http://www.realobjects.com/index.php?id=808&type=0] or Prince [http://www.princexml.com/]) that will produce pdfs either on the fly or statically, using a style sheet, but these can often get very complicated (requiring xslt as well). Consider whether page layout and numbering is as important to users as you think it is.

@import and @media
A stylesheet can start:

```plaintext
@import "reset.css";
@import "content.css";
@import "typography.css"
```

CSS next steps
University of Cambridge Computing Service, March 2013
@import "forms.css";
@import "lists.css";

@import allows import of rules from other style sheets. It can be very useful if you want to use several
stylesheets, keeping separate particular functionality, but call them all from a single stylesheet (but bear
in mind it uses less bandwidth to load one longer stylesheet than to load several shorter ones).

@import must always come first either in rule sets or in a stylesheet. If a separate @import stylesheet is
added to a page it may be used in the header, like this:

```html
<style type="text/css" media="all">@import "/style/import.css";</style>
<link rel="stylesheet" type="text/css" href="/style/camstyle.css" media="all" />
<link rel="stylesheet" href="/style/print.css" type="text/css" media="print" />
```

@import can include media types as well (which saves using an @media rule as well (see below).
Media="all" will be assumed as the default. Another way of including a media type in an @import rule
looks like this:

```html
<style type="text/css">@import url("fineprint.css") print; </style>
```

@media rules can be used if you just want to add a few simple css rules for different media, as follows:

```html
@import "fineprint.css"; @import "list-style.css";
```

Add media stylesheets in a link in preference to using either this method or @import, as it is more reliably
supported by browsers.

Streamlining style sheets

There are always several ways of writing stylesheet information. The arrangement on the page makes no
difference to the browser reading it - only the syntax is important, and how clear it is for you to follow
what the styles are for. Comments can and should be added to annotate the stylesheet to help you
remember what styles were set up for: enclose the note between /* and */.

Look at the examples at http://webhost.salford.ac.uk/common/general.css and
http://www.danwebb.net/stylesheets/base.css to see different ways of laying out a stylesheet.

Compacting your expressions

When using syntax such as `margin: 1em 0 2em 0;` to define separately the margin widths on four side of
an element, the order follows the quarters of a clock, clockwise, from 12, so the first is the top, the
second, right; the third, bottom; fourth, left.

```
margin: 1em 0 2em 0;
```

If they are all the same, one value will suffice. If the values for the top and bottom are the same and the
left and right the same, then two values will do the job. You can specify values for each margin
separately as needed (margin-right, etc), but beware that if you don’t specifying all values when you use
this method it may cause unexpected spacing problems.

```
margin: 1em;
```
The same system can be used for defining padding.

For colour definitions, when a hex number is composed of three sets of pairs, for instance #330066, the number can be compacted to a single number for each pair, for instance #306 (if you use web safe colours they will always be expressed as sets of pairs).

When it comes to syntax, some expressions can be compacted. For definition of borders, there are several individual properties that can be defined separately (border-width, border-style, border-color), and there is the shorthand border property that, when all four borders are to be the same, allows you to define all these three properties at once, making

\{ border-width: 1px; border-style: solid; border-color:#333; \}

into

{border: 1px solid #333;}

- the order of the property values doesn’t matter.

For definition of font, a similar shorthand can be employed, making

h1 {font-weight: bold; font-size: 120%; color: #990000; ;font-family: Arial, Helvetica, sans-serif}

into

h1 {font: bold 120% #900 Arial, Helvetica, sans-serif;}

CSS hacks

Many web designers (particularly for consumer sites) use hacks to accommodate giving style information (particularly) to versions of IE, all of which have serious flaws of various kinds. It is much better (and no more difficult) to seek a standards-based style solution and avoid using hacks, since, once you start, you are pretty much stuck with maintaining it.

- Developer information for IE10 on Windows 8 is at http://msdn2.microsoft.com/en-us/ie/ with links for IE9

Hacks are being used more recently to ensure that CSS3 attributes are used properly across a range of browsers (often supplying jQuery info to get older browsers to emulate the CSS3 functionality), and also to enable use of HTML5, but ensure the code works with older browsers.

Inheritance, cascading and multiple style sheets

As you saw in the first part of the course, elements specified in style sheets, for instance the colour of the base type set in the <body> tag, may be inherited by all elements through the document. Other elements are not inherited but may be computed, for instance the font size when specified proportionally will be calculated from the size of the base font (which will be there as a default setting but may be adjusted by the user). Many problems arising in older browsers come about because inheritance and calculation don’t work properly (or as expected).
Multiple stylesheets can and should be used in a document – there are rules built into the browser about how multiple stylesheets interact (or cascade - see http://www.w3.org/TR/REC-CSS2/cascade.html). As a rule the stylesheets specified by the author will be used in order of listing in the head, so the styles listed last will prevail (this includes style information in <style></style>). So if you need to use <style></style> always add it after any external links to stylesheets then you know they will always be used. In the end, if two different styles are ascribed to the same element, the one specified last in the chain will usually ‘win’. Imported stylesheets will also cascade but often styles imported into another stylesheet will ‘lose’. In the browser, stylesheets can usually be turned off so the user can implement their own needs. As a general rule, don’t try and be too complicated with your use of stylesheets since there is an overhead of linking to many separate stylesheets for each web page and the results may be fragile. Keep stylesheets as clear as you can.

**Troubleshooting style sheets**

Information about styles used on a web page can be viewed easily with the web developer toolbar (by going to the CSS menu and selecting ‘View style information’) or by installing the Firebug extension into Firefox. Firebug is by far the most effective tool (https://addons.mozilla.org/en-US/firefox/addon/firebug/). It calculates the style plus the inherited styles and allows you to modify styles on the fly to see effects before you change the stylesheets. Firebug (or Firebug-like functionality) is also built into Opera and Safari.

Other points to look out for:

- make sure your page validates before doing too much work on styles
- If you are catering for IE7/8 users, be ready to cater for them differently. You may need to provide browser specific stylesheets for the pages to work. Unless there is an overwhelming need, don’t bother trying to cater for IE6 users.
- if possible ensure you have a direct connection, not a cached connection
- make sure your browsers aren’t caching files, or that you are reloading a new version of the page each time you change styles
- during intensive work, keep frequent backups of stylesheets - stepping backwards can be difficult
- when styles are superseded, remove them from stylesheets - have a long stylesheet with lots of redundancy in it will give your browser extra work to do and will increase the download time for the page. There is a Firebug extension called CSS Usage that will help you identify unused styles (https://addons.mozilla.org/en-us/firefox/addon/css-usage/).

**Providing IE specific style sheets with conditional comments**

In the head part of your pages, if you need to, provide stylesheets for IE6 or later, like this:

```html
<!-[if IE 7]><link rel="stylesheet" href="global/style/ie7.css" type="text/css" media="screen" />
<![endif]-->
```

You can also provide IE directed javascript in the same way. IE10 does not support conditional comments (see http://msdn.microsoft.com/en-gb/library/ie/hh801214%28v=vs.85%29.aspx for more information).
Other examples and reference sites

CSS articles, downloads and videos - http://css-tricks.com/

HTML and CSS Tutorials, reference and articles (also book in list below) - http://htmldog.com/

CSS snippets from W3C - http://www.w3.org/Style/Examples/007/


How to develop standards-based web pages - http://dev.opera.com/

Tools and tutorials


- CSS validator (for CSS2 by default but can be switched to check CSS3) - see http://jigsaw.w3.org/css-validator/ - can be used online or downloaded.

- Impressive web - http://www.impressivewebs.com/tutorials/

Articles

- A list apart (http://www.alistapart.com/) - two new articles roughly every fortnight


Books (which date very rapidly)

