Programming the Internet—Lecture Notes

Week 1: Introduction to HTML5 and Cascading Style Sheets (CSS)

The Internet has actually existed for decades, but the World Wide Web (WWW) was developed in the early 1990s by Tim Berners-Lee from CERN (Conseil Européen pour la Recherche Nucléaire or, in English, the European Organization for Nuclear Research). The World Wide Web was designed to be governed by a group of protocols or rules for use on the Internet. These protocols determine how content on Web sites is accessed and transmitted. The Internet is a network of networks that hosts Web sites and forms the communication medium that enables Web browsers to download content from distant locations.

Web Protocols

Computers will not communicate with each other without a great deal of encouragement. On the Web there are a series of protocols or rules that set out agreed ways in which computers can communicate. On other occasions, you may well have an opportunity to study these communication issues in greater depth. However, at this stage you only need to understand the meaning of a few basic terms.

Web browser	A piece of software that enables users to connect to Web servers. It is described as client (as opposed to server) software. Examples would be Internet Explorer, Firefox, Opera, Google Chrome and Safari.
Web server	Technically, a piece of software that resides on a computer making a Web site available to Web browsers, but the term is often applied to the computer on which the server software has been installed. Examples of server software would be Apache and Internet Information Services (IIS).
URL	Uniform Resource Locator. A Web address such as www.google.com. The best known form of Uniform Resource Identifier (URI).
TCP	Transmission Control Protocol. Enables data to be transported across networks. It dictates how packets of data should be structured.
IP	Internet Protocol. IP primarily allows you to allocate addresses to computers on a network in the form of IP numbers (e.g. 216.27.61.137) so that messages can reach the correct destination.
HTML/XHTML	HyperText Markup Language/eXtensible HyperText Markup Language. A text and image formatting language for producing pages on the World Wide Web.
CSS	Cascading Style Sheets, often just referred to as style sheets. They provide a means of defining the appearance of a Web page, leaving HTML files to define the page's structure and contain its content.

Last update: 28 February 2013

HTTP	HyperText Transfer Protocol. Governs communications between a Web server and a Web browser. http:// at the beginning of a URL indicates that it is a Web address.
HTTPS	Secure HTTP. Sensitive communications are encrypted (turned into unreadable code) using SSL. Check the address of a site shown in a Web browser to see if it is preceded by https:// instead of http://.
SSL	Secure Sockets Layer. Provides a means of automatically encrypting data sent over the Internet.
FTP	File Transfer Protocol. Used for uploading files to and downloading files from an FTP server. Where the Web server and FTP server are allocated on the same directory on a computer, an FTP client (a piece of software on a local computer) can effectively upload files to a Web server. We shall do this later in this module. Web browsers can also communicate with FTP servers by using addresses beginning with ftp://.
DNS	Domain Name System. Computers use IP addresses to identify each other, but it is easier to remember names such as www.google.com. The DNS allows computers to have both names and IP numbers. The system looks up tables held on DNS servers and translates the names typed into IP numbers.

Table 1: Internet Protocols

There are a number of sources that will provide further information on these topics (Mitchell, n.d.; Orgera, n.d.; Cisco, 2012; W3 Schools, n.d.a).

Internet Standards Organisations

Several organisations work to develop standards and best practices for the World Wide Web. By a process of persuasion they hope to get manufacturers, the developers of Web browsers and influential users to adopt these standards.

Some of these bodies, such as the Internet Engineering Task Force (n.d.) and the IEEE Standards Association (n.d.), are more concerned with the physical infrastructure of the Internet, over which Web browsers communicate with Web servers.

Others are more concerned with software or good practice. The World Wide Web Consortium (W3C) is one of the most important bodies that is trying to develop standards for everything from HTML5 to Web accessibility for people with disabilities (W3C, n.d.a). The Internet Society (n.d.) works closely with W3C, and students should be aware of its activities.

The European Computer Manufacturers Association (Ecma) was influential in harmonising JavaScript standards (Ecma International, n.d.).

A Standards-Based Approach

Although many bodies are developing standards for the World Wide Web, not everyone complies with those standards or implements all of the standards right away. Sometimes businesses favour proprietary solutions, those which are developed by a single company according to its own requirements. Such solutions often create problems of interoperability. They may, for instance, not work on all browsers or interact properly with other hardware or software. When some browser vendors do not incorporate these standards into their browsers, it creates challenges for Web programmers.

In other cases, companies develop what they regard as improvements on the existing standards. In the short term, these solutions will run on only certain browsers, but manufacturers hope that they will be widely accepted in the future.

A number of organisations promote the use of Web standards as a way of making information more widely accessible and reducing the cost and complexity of development. The Web Standards Project (Holzschlag & Kaiser, 2002; n.d.) is a prime example of such an organisation. Also worth noting is OASIS (n.d.), the Organization for the Advancement of Structured Information Standards. W3C (2012) has a useful overview of the emergence of Web standards.

This module adopts a standards-based approach centred on HTML5, CSS, JavaScript and PHP. These programming technologies will be described over the coming weeks.

The Road to HTML5

Most of us will have heard of HTML (HyperText Markup Language), but there are different versions, each with its own rules. HTML4 is very tolerant of small errors and the inconsistent use of programming conventions. It tolerates sloppy code. Use of HTML4 would also be characterised by a tendency to combine presentation, structure and content in the same file. The different varieties of XHTML (eXtensible HyperText Markup Language) were designed to impose stricter rules about the production of code. This has certain advantages:

- Although browsers can work hard to interpret sloppy HTML code, they may do this in different ways. Poor code can display very differently in different browsers. Validated XHTML is likely to display more consistently across browsers.

- Code produced according to strict standards (standards-based) is easier to maintain. It can more easily be amended and updated by programmers who have inherited code they did not write.

- Users are increasingly employing mobile devices to access the World Wide Web, and these devices are often less able to cope with bad markup code than PCs. Code produced to XHTML standards helps significantly.

- There are programs, such as those used by search engines, which read Web pages. It is easier for a program to do this if pages are produced according to a set of strict standards.

- Learning XHTML encourages good habits and is a good introduction to learning XML (eXtensible Markup Language). It is easier to start in a more disciplined mode and become more relaxed than to begin with few rules and then become more disciplined.

- XHTML helps to separate presentation (appearance) from structure and content (which you will learn about shortly) by deprecating or ruling out certain tags such as <font> that determine appearance within the HTML code.

This worked quite well under the different versions of XHTML1, but the idea of strictness was pushed too far when XHTML2 was introduced. It was not backwardly compatible, meaning that code produced under older standards would not display properly. It also encouraged the idea that XHTML files would be rather like the even stricter XML. It meant that any errors in a file with a .xhtml extension would typically prevent the file from displaying at all in a browser. This was a step too far.

Some browser vendors and others formed the Web Hypertext Application Technology (WHAT) Working Group and began working on a new version of HTML (WHATWG, 2012). Eventually, this became a new Web standard known as HTML5 (although it is still in draft form). HTML5 is a little more neutral on the question of how strict we should be producing code; it leaves more of that up to the developer. In this module, we shall use most of the rules employed in XHTML1 for the very good reasons outlined above. At the same time, HTML5 has introduced new features that allow programmers to add more functionality to their Web pages. Unfortunately, support by Web browsers for these additional features has been rather patchy. We shall look at some examples of the new features later in the module.

Our First HTML5 code

HTML is contained in a text file, and we shall be using the file extension .html. If you use a .xhtml extension by mistake, you may find that the slightest error will prevent it from displaying in a Web browser. An HTML file normally resides on a Web server and is accessed from remote locations. While programmers are developing such files, however, they can open and display them in a Web browser simply by using commands such as File > Open (Ctrl+O in Google Chrome and Safari for Windows). You may also wish to

open files within a text editor such as Notepad++ or Aptana Studio (both mentioned at the end of the Lecture Notes).

That is enough by way of introduction. Let's take a look at our first example of HTML5 code.

```
<!DOCTYPE html>
<html>
	<head>
		<title>First HTML5 Code</title>
		<meta charset="utf-8" />
	</head>

	<body>
		<!--This is an HTML comment, useful for explaining what
	our code does. The browser will ignore it. -->
		<h1>Our First HTML5 Code</h1>
		<p>
			The tags around this text mean it will be <br />
	shown as a separate paragraph.
		</p>
		<p>
			This is the text in a second paragraph.
		</p>
	</body>
</html>
```

Figure 1: FirstHTML5.html

For those who are unfamiliar with HTML, let's spend a little time explaining what is going on. Experienced students may wish to skip this section. All the code examples in the Lecture Notes are available in zip files within the Learning Resources each week, so you do not need to retype them. Copying and pasting from these Lecture Notes is not recommended as it can introduce non-standard text characters which your browser will not understand.

The `<!DOCTYPE html>` is a Document Type Declaration. It tells you that this file complies with the HTML5 standard. This is much shorter than in XHTML, where the following would be more typical:

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

There are then a number of *tags* enclosed in angled brackets (e.g. `<p>` and `</p>`). These provide information to the browser about how information should be displayed. XHTML requires opening and closing tags and we shall follow that very good habit in HTML5. The text between the opening and closing tags is displayed in the browser. Let's look at the tags first.

The `<html>` and `</html>` tags mark the beginning and end of the area of the code that the browser will actually execute, usually by trying to display something. If you write code outside these tags, it will not be seen. The `<head>` tags can contain a range of items, but one of the most important is the

<title>. This is the information that will be displayed in the browser's title bar at the top of a window. It is a very useful indicator to the user about what is contained in the document. It is almost always worth using the <title> facility.

We can also see that in the <head> section the charset or character set that is being used has been defined. This file uses the type of characters defined in the Unicode standard UTF-8. The main alternative is ISO-8859-1, which is the default setting in some Web servers, but UTF-8 covers a wider range of non-English characters and is the safer option (Oracle, 2010).

As the name implies, the <body> tags define the material that will be displayed in the body of the browser. It is here that most of the HTML code will be written. In our example, you see a few of the most commonly used tags to help structure content.

The first example is a comment enclosed in the tags <!-- and -->. They are a little different from the usual tags used, but they are very useful. In the assessments for this module we shall constantly ask you to insert comments in your code to explain what is going on. This helps reassure your assessors that you understand the code you are producing. Make good use of these comment tags.

The tag <h1> is a level 1 heading. The levels range from 1 to 6, where 1 is the largest. In this case, the actual displayed size will depend on the particular browser, but the heading will normally be shown in bold. We shall soon see how, with the use of Cascading Style Sheets (CSS), the programmer can specify more precisely how items will be displayed.

The tag denotes a paragraph. Almost all browsers will interpret this as meaning that when the closing tag is reached, the next section of text should be displayed on a new line but only after a blank line has been inserted. It will also usually ensure that there is a blank line just above the text enclosed in tags.

The tag
 denotes a line break. When this tag is encountered, the following text will be displayed on a new line but without inserting a blank line. Note the effect of the
 tag when we display our example code in a browser. This knowledge will immediately help us to produce better formatted Discussion posts in the online classroom. When we compose a message, the online classroom automatically converts our text into HTML. As with all automated code generators, it does not always read our intentions correctly, and it can produce some eccentric spacing between sentences.

There is, however, a facility in the online classroom to inspect and edit the HTML code used to display Discussion posts if the displayed results are not to our liking. This is done by clicking on the <> icon in the online classroom message editor (click it again to go back to the displayed message). Most of the problems are created by the automatic insertion of too many tags or the use of when we really wanted a
 tag. Now that we are all going

to become confident HTML programmers, we should have no hesitation in editing the code produced by the online classroom so that our messages are spaced in the way we intended. Note that a browser will ignore carriage returns and white space in the HTML we write. If we want the output in a browser to take a new line or insert blank lines, we need to specify this by the use of HTML tags.

Now the perceptive members of the class will have noticed that `<br />` tags do not come in pairs. It is one of a small number of exceptions where the tag is not used for displaying text between opening and closing tags in a particular way. Another example would be the `<hr />` tag, which displays a horizontal line. In these cases, XHTML (and, hence, our strict approach to HTML5) insists that the tag be terminated properly, and that means that the forward slash should always be included, but in these cases it will be after the element name. The space between the 'br' and the forward slash is not strictly necessary, but it is considered good practice because some very old browsers require it to display material correctly.

We have referred indirectly to the XHTML-style good habits that we are going to follow. They are not very onerous. Here they are:

- HTML code should be written in lower case letters.
- All elements must be closed (we have already seen this means using a closing tag for `<p></p>` and a terminating forward slash with `<br />`).
- HTML documents should be well-formed, including at least the following tags: `<html>`, `<head>`, `<title>`, `<body>`.
- There must be a DOCTYPE declaration.
- Where elements have an attribute, then their value must be enclosed in double quotes. In our example, charset="utf-8" is such an example.
- All text to be displayed must be contained within tags (such as `<p>` and `</p>`).
- All elements must be properly nested. This refers to tags inside tags. In our example the `<title>` tags are inside the `<head>` tags. Proper nesting means that they do not overlap. This would be wrong:

  ```
  <title>
  <head>
  </title>
  </head>
  ```

  The inner tag must be terminated or closed before the outer tag. This is the correct order:

```
<title>
<head>
</head>
</title>
```

To check whether code complies with the relevant HTML standard, the World Wide Web Consortium (W3C) has provided an online markup validation service (W3C n.d.b) at http://validator.w3.org/. There are three ways of using it:

1. Type in the Web address (URL or URI) of a page that can be accessed over the Internet.

2. Upload a file held on a local machine to the validation service.

3. Copy and paste code (validation by direct input) into the appropriate validation service page.

If there are errors (the code does not comply with all the rules), the validation service should provide some indication of what is going wrong. It will also help you debug your HTML. Make sure you also apply our XHTML-style good habits, which will not be checked when HTML5 is validated.

HTML5 has a reputation for being more tolerant than XHTML, but that does not mean that anything goes. This standard has declared a number of practices obsolete (they should not be used), and these include frames (W3C, n.d.c). Most of these obsolete practices can be avoided by ensuring that content and presentation are kept separate. This is discussed in the next section.

CSS: Separating Content and Presentation

Historically, HTML tried to include both the information to be viewed and instructions about how it should appear in the same file. There are, however, benefits in separating out the content and the presentation aspects into separate files. This enables the appearance (presentation) to be changed easily where the content remains the same or vice versa. The code is more easily maintained, meaning that it can be corrected or updated with less difficulty. It is an ideal situation towards which we are working, but CSS helps us to achieve this separation to a great extent. They are often just referred to as *style sheets*.

XHTML 1.0 Strict encouraged us to adopt this approach by deprecating (removing or making unacceptable) certain tags used by HTML. HTML5 continues this tradition by declaring some tags obsolete. The tags to avoid include `<font>` for defining the font face (Times Roman, Arial, etc.), size and colour; `<center>` for centring text on the Web page; and `<u>` for underlining text. In addition, the use of the `<b>` (bold) and `<i>` (italic) tags is discouraged. To maintain the separation of content and presentation as much as possible, we can instead employ `<strong>` tags for bold text and `<em>` (emphasis) tags

for text in italics, and then define their precise style in CSS (e.g. with font-weight: bold; or font-style: italic).

Now it is possible to use the style sheets approach within the HTML file itself (embedded style sheets), but that rather defeats the purpose of separating content and presentation as much as possible. Similarly, we can have in-line styles where CSS is applied within an HTML tag, for instance, to a single paragraph. However, the same effect can be achieved by other means where there is a separate, external CSS file. Hence, in our examples we shall point our HTML code at an external .css file.

Here is our first HTML5 example with an extra line of code in the `<head>` section pointing to a file called style1.css:

```
<!DOCTYPE html>
<html>
	<head>
		<title>First HTML5 Code</title>
		<meta charset="utf-8" />
		<link href="style1.css" type="text/css" rel="stylesheet"
		/>
	</head>

	<body>
		<!--This is an HTML comment, useful for explaining what
	our code does. The browser will ignore it. -->
		<h1>Our First HTML5 Code</h1>
		<p>
			The tags around this text mean it will be shown as
	a separate paragraph.
		</p>
		<p>
			This is the text in a second paragraph.
		</p>
	</body>
</html>
```

Figure 2: FirstHTML5withCSS.html

In the above example, we assume that the file style1.css is in the same directory as our HTML file, and hence we do not need to supply a full URL. We can then show what we might put in this CSS file to influence the appearance of our Web page.

```
h1{
font-size: 32pt;
}
p{
color: blue;
font-size: 14pt;
font-family: "Times New Roman", serif;
}
```

Figure 3: style1.css

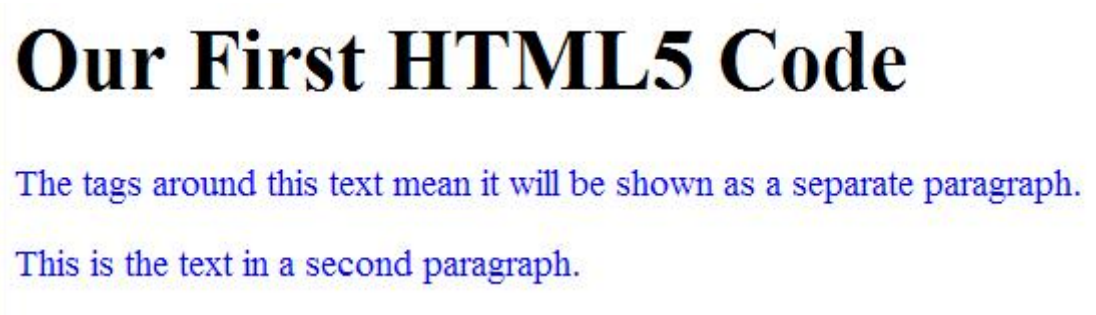

Figure 4: Output from FirstHTML.html and style1.css

In this example, we have also defined the size of the `<h1>` headline as being 32 point. Do not forget those braces (curly brackets) in the CSS file to indicate the beginning and end of our instructions about how information within all `<h1>` or `<p>` tags should be set out. Note also the important semicolons at the end of each line.

The CSS code determines that any text within `<p>` and `</p>` tags will be displayed in blue. The size will be 14pt and our preference will be for the font to be Times New Roman. If this is not available in the browser, another serif typeface will be used. The quotation marks are needed around "Times New Roman" because it consists of more than one word. A single word such as Arial does not require quotation marks. Note that both HTML and CSS tend to use US English spellings such as 'color'.

Hypertext Links

One of the main attractions of using Web pages is that they can be linked to many other pages. By convention, this is done by clicking on underlined text. Having underlined text that is not a link can cause confusion. Many Web pages remove the underlining from hypertext links and instead change the colour of the link text when the mouse hovers over it.

Let us add to our example HTML file some links to useful pages at W3 Schools, a fine source of introductory material about Web technologies.

```
<!DOCTYPE html>
<html>
  <head>
      <title> First HTML5 Code With Links </title>
      <meta charset="utf-8" />
      <link href="style1.css" type="text/css" rel="stylesheet" />
      <link href="style2.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
      <h1>Our First HTML5 Code With Hypertext Links</h1>
            <p>
                  The tags around this text mean it will be shown as
a separate paragraph.
            </p>
            <p>
                  This is the text in a second paragraph.
```

```
            </p>
            <p><a href =
"http://www.w3schools.com/css/css_font.asp">Serif and Sans-Serif
Fonts Explained </a> <br />
            <a href =
"http://www.w3schools.com/html5/default.asp">HTML5 Tutorial </a> <br
/>
            <a href = "http://www.w3schools.com/css/default.asp">CSS
Tutorial </a></p>
  </body>
</html>
```

Figure 5: FirstHTMLWithLinks.html

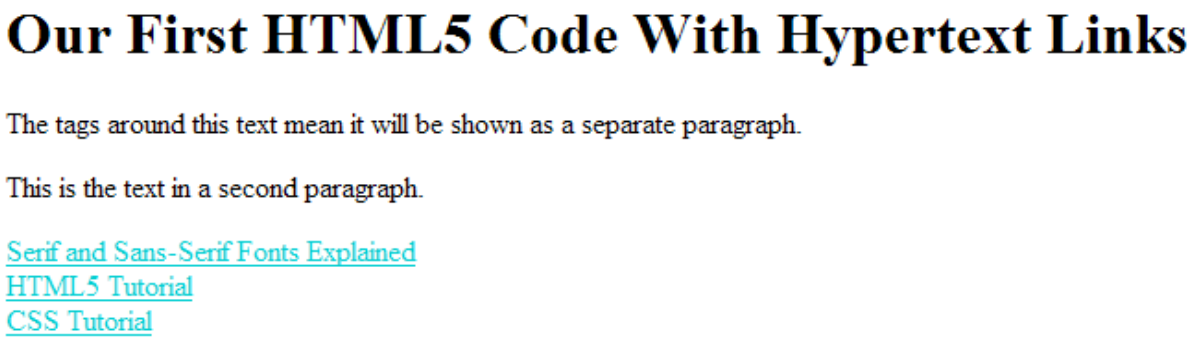

Figure 6: Output from FirstHTMLWithLinks.html and style2.css

A hypertext link is placed between `<a>` and `</a>` (anchor) tags. The letters *href* stand for hypertext reference, and it is an attribute of the `<a>` tag. The text in double quotes is the attribute's value, and that will typically contain the name of a file to which we want to link. In this case, we have full URLs (Web addresses), but we could link to another file in the same directory on the same server if we wanted to without using a URL. We can even link to different parts of the same Web page.

The text between the `<a>` and `</a>` tags is what is shown in the browser, usually as underlined text. This is what the user clicks on. You may use the URL itself as the text to be clicked (you would have to type it a second time), but it in many cases it is more helpful to provide some descriptive text for the hypertext link.

Below, we shall see how we could link to another part of the same page. This will work best when the page is long and links would help us find our way around.

```
<!DOCTYPE html>
<html>
  <head>
      <title> First HTML5 Code With Links </title>
      <meta charset="utf-8" />
      <link href="style1.css" type="text/css" rel="stylesheet" />
      <link href="style2.css" type="text/css" rel="stylesheet" />
  </head>
```

```
  <body>
      <h1>Our First HTML5 Code With Hypertext Links</h1>
            <a href = "#PageBottom"> Take me to the bottom of the Web
page </a>
            <p>
                  The tags around this text mean it will be shown as
a separate paragraph.
            </p>
            <p>
                  This is the text in a second paragraph.
            </p>
            <p><a href =
"http://www.w3schools.com/css/css_font.asp">Serif and Sans-Serif
Fonts Explained </a> <br />
            <a href =
"http://www.w3schools.com/html5/default.asp">HTML5 Tutorial </a> <br
/>
            <a href = "http://www.w3schools.com/css/default.asp">CSS
Tutorial </a></p>
            <a name = "PageBottom"> The bottom of the Web page </a>
  </body>
</html>
```

Figure 7: FirstHTMLWithInternalLinks.html

First, we have to give a part of the Web page a name so that we can find it. At the end of the above HTML file we have defined a location called *PageBottom* using `<a name = "PageBottom">`. Near the beginning of the file we have defined a link using `<a href = "#PageBottom">`. If the user clicks on this link, he or she will be taken to that part of the page defined as *PageBottom* in our example. This example is a little artificial because the page is so small, but on a larger page these internal links can help the user find his or her way around.

Here are some examples of the CSS we can apply to links:

```
a:link { /* unvisited link */
color: #00CED1;
}

a:visited { /* visited link */
color: #FF1493;
}

a:hover { /* when the mouse hovers over the link */
color: red;
text-decoration: none; /* the link is not underlined */
}
```

Figure 8: style2.css

The reader will almost certainly be able to improve on this shocking choice of colours. A list of CSS colour names supported by most browsers can be found at W3 Schools (n.d.b). However, the only way of ensuring maximum browser support for some of the more obscure colours is to use the corresponding hexadecimal value preceded by a hash (#) sign. Hexadecimal is a 16-base

number system that uses the numerals 0 to 9 and then the letters A to F to represent the numbers 10 to 15. Thus, #00CED1 is DarkTurquoise and #FF1493 is DeepPink. The numbers denote different combinations of red, green and blue.

There are, however, 16 standard colour names defined by the W3C which are pretty safe to use. They are blue, black, red, green, white, yellow, aqua, fuchsia, gray, lime, maroon, navy, olive, purple, silver and teal. For more information about colours, see W3 Schools (n.d.c).

Comments can be inserted into CSS between the symbols `/*` and `*/`. As usual, comments will be ignored by browsers. They are useful in making the meaning of code clearer to programmers who come after us, and they will be essential in the assignments you complete. Note that the symbols used for comments in CSS are different from those used in HTML files.

You will see that where the programmer wants to remove the underlining from links, the CSS to use is `text-decoration: none; .`

Now that our CSS code is becoming a little more sophisticated, we can check that it is correct by going to the W3C (n.d.d) site http://jigsaw.w3.org/css-validator/. The examples in these Lecture Notes have been validated at CSS3, the latest standard. However, in these introductory examples they would tend to use facilities that were available in CSS2.x to provide maximum browser support. Some facilities introduced in CSS3 are not well supported by all major browsers.

Unordered Lists

A simple way to structure a collection of links, for instance, is to use an unordered list. As its name implies, it is a list that does not use numerical or alphabetical characters to imply that items appear in a particular order. Unordered lists use bullet points in the form of discs, circles or squares. As usual, we try to separate the content from the presentation of the data by employing CSS. First, the HTML:

```
<!DOCTYPE html>
<html>
  <head>
      <title> An Unordered List </title>
      <meta charset="utf-8" />
      <link href="style1.css" type="text/css" rel="stylesheet" />
      <link href="style2.css" type="text/css" rel="stylesheet" />
      <link href="style3.css" type="text/css" rel="stylesheet" />
  </head>
  <body>
      <h1>An Unordered List</h1>

            <p>
                  The tags around this text mean it will be shown as
a separate paragraph.
            </p>
```

```
            <p>
                  This is the text in a second paragraph.
            </p>
            <ul>
            <li><a href =
"http://www.w3schools.com/css/css_font.asp">Serif and Sans-Serif
Fonts Explained </a></li>
            <li><a href =
"http://www.w3schools.com/html5/default.asp">HTML5 Tutorial </a></li>
            <li><a href =
"http://www.w3schools.com/css/default.asp">CSS Tutorial </a></li>
            </ul>
  </body>
</html>
```

Figure 9: ExampleUL.html

The unordered list appears between `<ul>` and `</ul>` tags. Each item in the list is enclosed in `<li>` and `</li>` tags. This will automatically place each item on a new line, so in this example, we do not need to use tags such as `<br />`. If we leave it like this, the browser would use its default value for the bullet points. In many cases, that may be adequate. However, we are taking control with CSS, and hence we might want to add an external style sheet as follows:

```
ul {
list-style-type: square;
}
```

Figure 10: style3.css

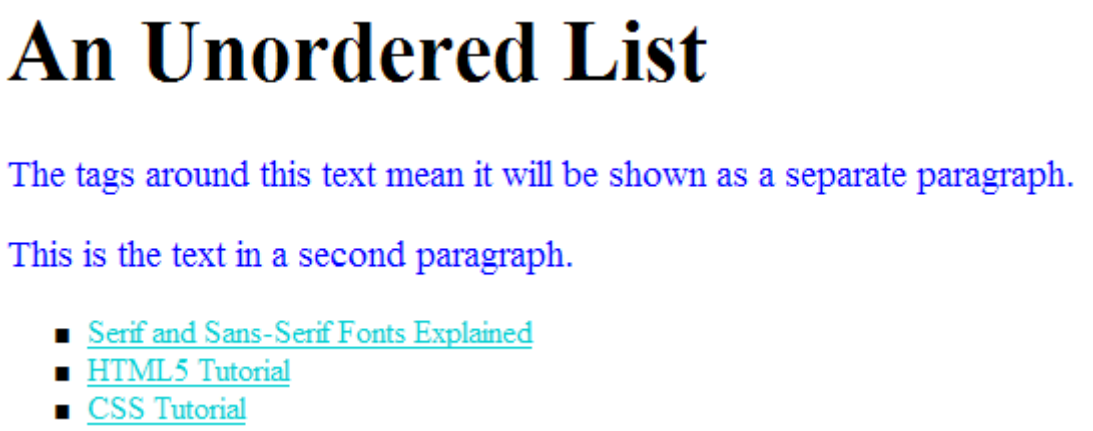

Figure 11: Output from ExampleUL.html and style3.css

The CSS will ensure that we have square bullet points. Alternatives to square would be circle and disc. There are many other ways to style content other than those in these Lecture Notes. Once you have become confident in using style sheets, you can explore these other options in your own time.

Ordered Lists

The HTML code involves just changing `<ul>` to `<ol>`.

```
<!DOCTYPE html>
<html>
  <head>
      <title> An Ordered List </title>
            <meta charset="utf-8" />
            <link href="style1.css" type="text/css" rel="stylesheet"
/>
            <link href="style2.css" type="text/css" rel="stylesheet"
/>
            <link href="style4.css" type="text/css" rel="stylesheet"
/>
  </head>
  <body>
      <h1>An Ordered List</h1>

            <p>
                  The tags around this text mean it will be shown as
a separate paragraph.
            </p>
            <p>
                  This is the text in a second paragraph.
            </p>
            <ol>
            <li><a href =
"http://www.w3schools.com/css/css_font.asp">Serif and Sans-Serif
Fonts Explained </a></li>
            <li><a href =
"http://www.w3schools.com/html5/default.asp">HTML5 Tutorial </a></li>
            <li><a href =
"http://www.w3schools.com/css/default.asp">CSS Tutorial </a></li>
            </ol>
  </body>
</html>
```

Figure 12: ExampleOL.html

A style sheet entry might look like this:

```
ol {
list-style-type: lower-roman;
}
```

Figure 13: style4.css

This will display lower case Roman numerals at the beginning of each line, (e.g. i, ii, iii, iv, v and so on). Alternatives might be upper-roman, decimal (1, 2, 3, etc.), lower-alpha (a, b, c, etc.), and upper-alpha.

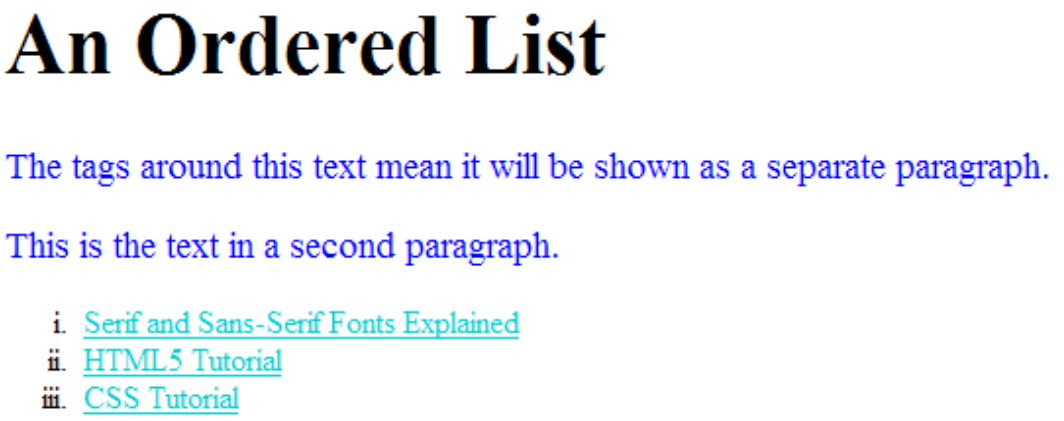

Figure 14: Output from ExampleOL.html and style4.css

Images

To conclude this week's Lecture Notes, we should not neglect the inclusion of images on the Web page. The image used here is from the Education Scotland Web site. This is provided royalty-free for educational purposes as long as we acknowledge the source, which is http://www.educationscotland.gov.uk/rsascottishart/imagedetails/designfortheroyalhighschooledinburgh.asp (Hamilton, 1831). It is a picture described as a 'Design for the Royal High School, Edinburgh'. Always show respect for intellectual property rights.

Below is the code for our HTML page displaying the image. Here, we have downloaded and stored the picture in the same directory as our HTML file.

```
<!DOCTYPE html>
<html>
  <head>
      <title> An Ordered List With an Image </title>
            <meta charset="utf-8" />
            <link href="style1.css" type="text/css" rel="stylesheet"
/>
            <link href="style2.css" type="text/css" rel="stylesheet"
/>
            <link href="style4.css" type="text/css" rel="stylesheet"
/>
            <link href="style5.css" type="text/css" rel="stylesheet"
/>
  </head>
  <body>
      <h1>An Ordered List</h1>

            <p>
                  The tags around this text mean it will be shown as
a separate paragraph.
            </p>
            <p>
                  This is the text in a second paragraph.
            </p>
```

```
            <ol>
            <li><a href =
"http://www.w3schools.com/css/css_font.asp">Serif and Sans-Serif
Fonts Explained </a></li>
            <li><a href =
"http://www.w3schools.com/html5/default.asp">HTML5 Tutorial </a></li>
            <li><a href =
"http://www.w3schools.com/css/default.asp">CSS Tutorial </a></li>
            </ol>

            <p><img src = "HamiltonT1p2sw_tcm4-307759.jpg" alt =
"Picture of a Design for the Royal High School, Edinburgh" /></p>
  </body>
</html>
```

Figure 15: ExampleOLwithImage.html

The letters *img src* stand for the image source (the name of the image file). The browser will here expect the image file to be in the same directory as the HTML file, but you could use a full URL if you wished. The term *alt* is short for alternative (or alternate in US English). This *alt* text will be displayed if images are turned off in a user's browser, and it may be of use to someone who is visually disabled.

Here are a few things we can do with our image in a style sheet:

```
img{
float: left;
width: 200px;
border: 2px;
border-style: dotted;
border-color: #00008B;
padding: 10px;
}
```

Figure 16: style5.css

Figure 17: Output from ExampleOLwithImage.html and style5.css

- The image will be displayed on the left and any subsequent text will be displayed beside it on the right (*float*). The image can be floated to the right if we wish.

- It will be shown as 200 pixels wide. Note that the screen resolutions are defined in pixels (W3 Schools, n.d.b). If in doubt about your own screen resolution, visit http://www.screenresolution.org/ (Sheng, 2013. This figure could just as easily have been defined as a percentage (e.g. 25%), meaning that the image would take up 25% of the screen's width. The picture will maintain the correct proportions as its width is adjusted.

- The image has been given a border which has a thickness of 2 pixels.

- The border of the image will be in the form of a dotted line and coloured DarkBlue (#00008B). Some alternatives to dotted could be solid, double, dashed or none.

- The padding inserts a gap of 10 pixels between the image and the border.

By now, you will have the confidence to change these settings to produce something that looks much better. A little research will help you explore additional options.

Software for Web development

In the coming weeks, we shall be using a number of pieces of software to assist us in our work.

1. ***A Web browser:*** Any modern version of a Web browser, such as Internet Explorer, Firefox, Google Chrome, Opera or Safari (for Windows or Mac), should be satisfactory. Because not all of the new features of HTML5 are implemented in all browsers, it will be useful to install more than one browser on your local computer. It is recommended that you download and utilise either Google Chrome or Safari, as these tend to support the most HTML5 features.

2. ***An HTML editor:*** This will also help you type JavaScript and PHP programs. Some editors automatically generate code after manipulating visual objects on screen. They are known as WYSIWYG (what you see is what you get). They are not allowed in this class, as they do work which the student should be doing for him or herself in order to learn programming. The use of such code generators would be regarded as cheating. Free software and open source editors which provide a small amount of assistance and are suitable for the purposes of this class are Notepad++ (http://notepad-plus-plus.org/) (n.d.) and Aptana Studio (http://www.aptana.com/) (n.d.). Whichever is used is

simply a matter of preference. If students intend to use other editors, please check with the module Instructor that they are permissible.

3. ***All students will be allocated space on Laureate's Web server.*** This will be important in Week 5 and onwards when we shall need to see PHP programs running on a Web server. In order to upload files from your local computer to this server, students will need to install some FTP client software. Any such software should be satisfactory, but you may wish to download and install FileZilla (http://filezilla-project.org/download.php?type=client) (n.d.a). Documentation for FileZilla can be found at http://wiki.filezilla-project.org/Documentation (n.d.b).

4. ***It is possible to install a Web server, such as Apache or IIS, on your local machine for development purposes if you wish, but this is not necessary.*** HTML and JavaScript files will run happily in a Web browser, and PHP programs can be uploaded to the Web server space that will be allocated to you and run from there. If you are feeling adventurous, a quick way of installing Apache, MySQL and PHP on your local machine is to download a free package such as XAMPP (http://www.apachefriends.org/en/xampp.html) (...APACHE FRIENDS..., 2011).

References

...APACHE FRIENDS... (2011) *XAMPP* [Online]. Available from http://www.apachefriends.org/en/xampp.html (Accessed: 2 February 2013).

Aptana (n.d.) [Online]. Available from http://www.aptana.com/ (Accessed: 2 February 2013).

Cisco (2012) *Internetworking technology handbook: internetworking basics* [Online]. Available from http://docwiki.cisco.com/wiki/Internetworking_Technology_Handbook#Internetworking_Basics (Accessed: 2 August 2012).

Ecma International (n.d.) *What is Ecma International?* [Online]. Available from http://www.ecma-international.org/memento/index.html (Accessed: 2 February 2013).

FileZilla (n.d.a) *Client download* [Online]. Available from http://filezilla-project.org/download.php?type=client (Accessed: 2 February 2013).

FileZilla (n.d.b) Documentation [Online]. Available from http://wiki.filezilla-project.org/Documentation (Accessed: 2 February 2013).

Hamilton, T. (1831) *Design for the royal high school, Edinburgh* [Online image]. Available from

http://www.educationscotland.gov.uk/rsascottishart/imagedetails/designfortheroyalhighschooledinburgh.asp (Accessed: 2 February 2013).

Holzschlag, M.E. & Kaiser, S.E. (2002) 'Frequently asked questions (FAQ): What are web standards and why should I use them?'. *The Web Standards Project* [Online]. Available from http://www.webstandards.org/learn/faq/ (Accessed: 2 August 2012).

IEEE Standards Association (n.d.) *Develop standards* [Online]. Available from http://standards.ieee.org/stdsdev/index.html (Accessed: 2 August 2012).

Internet Engineering Task Force (n.d.) *About the IETF* [Online]. Available from http://www.ietf.org/about/ (Accessed: 2 August 2012).

Internet Society (n.d.) *FAQ about ISOC and W3C* [Online]. Available from http://www.isoc.org/standards/isoc-w3c-faq.shtml (Accessed: 2 August 2012).

Mitchell, B. (n.d.) *Internet protocol tutorial: domain name system* [Online]. Available from http://compnetworking.about.com/od/dns_domainnamesystem/a/introduction-to-dns_domain-name-system.htm (Accessed: 2 February 2013).

Notepad++ (n.d.) [Online]. Available from http://notepad-plus-plus.org/ (Accessed: 2 February 2013).

OASIS (n.d.) *About us* [Online]. Available from https://www.oasis-open.org/org (Accessed: 2 February 2013).

Oracle (2010) *What is UTF-8 and why is it important?* [Online]. Available from http://developers.sun.com/dev/gadc/technicalpublications/articles/utf8.html (Accessed: 2 August 2012).

Orgera, S. (n.d.) *What is a web browser?: web browsers and how they work* [Online]. Available from http://browsers.about.com/od/howbrowserswork/a/whatisabrowser.htm (Accessed: 2 August 2012).

Sheng, A. *What Is My Screen Resolution/Display Resolution?* [Online]. Available from http://www.screenresolution.org/ (Accessed: 15 February 2013)

The Web Standards Project (n.d.) *About* [Online]. Available from http://www.webstandards.org/about/ (Accessed: 2 August 2012).

W3C (n.d.a) *Help and FAQ* [Online]. Available from http://www.w3.org/Help/ (Accessed: 2 August 2012).

W3C (n.d.b) *Markup validation service* [Online]. Available from http://validator.w3.org/ (Accessed: 2 August 2012).

W3C (n.d.c) *HTML 5.1 nightly: obsolete features* [Online]. Available from http://dev.w3.org/html5/spec/single-page.html#obsolete (Accessed: 2 August 2012).

W3C (n.d.d) *CSS validation service* [Online]. Available from http://jigsaw.w3.org/css-validator/ (Accessed: 2 August 2012).

W3C (2012) *The history of the web: the coming of web standards* [Online]. Available from http://www.w3.org/community/webed/wiki/The_history_of_the_Web#The_coming_of_web_standards (Accessed: 2 August 2012).

W3 Schools (n.d.a) *TCP/IP tutorial* [Online]. Available from http://www.w3schools.com/tcpip/default.asp (Accessed: 2 August 2012).

W3 Schools (n.d.b) *CSS color names* [Online]. Available from http://www.w3schools.com/cssref/css_colornames.asp (Accessed: 2 August 2012).

W3 Schools (n.d.c) *CSS colors* [Online]. Available from http://www.w3schools.com/cssref/css_colors.asp (Accessed: 2 August 2012).

WHATWG (2012) *FAQ* [Online]. Available from http://wiki.whatwg.org/wiki/FAQ (Accessed: 2 August 2012).