

Today's Lecture

- ▶ Assignment 1 due Friday (28th)
- ▶ Labs next week
- ▶ The W3C Recommendations
- ▶ The Anatomy of CSS
- ▶ Applying CSS to XHTML Pages
- ▶ Cascading

Cascading Style Sheets (CSS)

HTML is a markup language, which means that its primary purpose is to **mark up** the structural parts of the document (such as headings and lists). It was not designed to control the **appearance** of the document (that is, the style).

Over the years, many appearance elements have been incorporated into HTML. However this means that there is little separation between the document content and layout instructions. This makes pages difficult to maintain and leads to accessibility issues (more on this with Tom Worthington).

The solution? To specify the style of a web page in a separate file called a **style sheet**.

The W3C Recommendations

The W3C has three recommendations for Cascading Style Sheets:

- ▶ **CSS1** released in 1996. Contains properties for fonts, margins, colours etc.
- ▶ **CSS2** released in 1998. Includes everything in CSS1 and expands on it by supporting features that allow positioning of content. This is the specification in use.
- ▶ **CSS3** under development. Includes everything in CSS2 and extends it with fancy borders and backgrounds, vertical text, user interaction, and speech to name a few.

See <http://www.w3c.org/Style/CSS/> for more details.

The Anatomy of CSS

A style sheet is a collection of **rules** written in a particular format. A rule consists of two parts:

- ▶ a **selector** which defines the XHTML element(s) to which the style will be applied, and
- ▶ a **declaration** which defines the style to be applied to the selector.

Example 1:

```
h1 {text-align : center;}
```

Example 1 illustrates a simple CSS rule. In this example, `h1` is the selector and `text-align : center;` is the declaration.

A declaration also has two parts:

- ▶ the **property** which is the property of the selector that is to be set, and
- ▶ a **value** which the property will take on (for instance, a position or a colour).

In Example 1 `text-align` is the property to be set and `center` is the value.

Example 2:

```
p {color : blue;}
```

Question: What is the selector, declaration, property and value in Example 2? What effect will this rule have?

Grouping Selectors and Properties

Selectors can be **grouped** if they have the same property declarations, for example:

```
h1, h2, p {color : orange;}
```

Declarations can be **grouped** if they all apply to the same element, for example:

```
h1 {text-align : center;  
    color : blue;  
    font-family : arial;  
}
```

Note that **commas** (,) separate grouped selectors and **semicolons** (;) separate grouped property declarations.

Contextual Selectors

All the styles we have defined so far apply to the elements whenever they are used in a document. However it is possible to define the exact sequence of elements to which a rule will apply by using **contextual selectors**.

Example 3:

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

Example 3 defines a rule which causes emphasised text to appear red when used in an `h1` element. Everywhere else it will appear with its “default” colour.

Question: What does `ol li {list-style-type : lower-roman;}` do?

Applying CSS to XHTML Pages

How do we apply the styles in a CSS to a specific web page? There are three ways of doing this in XHTML:

- ▶ **linking** the web page to an external style sheet,
- ▶ **embedding** the CSS style sheet in the web page,
- ▶ applying CSS-style rules **inline** to a section of the page.

These methods can be used in combination. When combined, **in-line** styles take precedence over **embedded** style sheets which take precedence over **linked** style sheets.

In XHTML Basic, we have only one valid choice: linking to an external CSS.

Linking to an External CSS

An **external style sheet** is a text file containing one or more CSS rules. It must have the extension `.css`, for example `assignment_template.css`.

It is linked to an XHTML document by using the `<link>` tag in the `<head>` element of the page.

Example 4:

```
<link rel="stylesheet" type="text/css" href="lecture_12_style1.css" />
```

The **attributes** of the `link` tag:

- ▶ **rel** – used to define the relationship between the linked file and the XHTML document (more on this in a moment).
- ▶ **type** – specifies a media type, in this case a CSS text file.
- ▶ **href** – the URL of the style sheet file.
- ▶ **title** – names a style sheet.

Types of External Style Sheets

HTML and XHTML allows authors to associate **any number** of external style sheets with a document. External style sheets fall into one of three categories: **persistent, preferred, alternate**.

- ▶ **Persistent** style sheets must *always* be used by the browser in addition to any other style sheets specified.

To make a style sheet persistent, set the `rel` attribute to `stylesheet` and don't set the `title` attribute.

The linked style sheet in Example 4 is a **persistent** style sheet.

- ▶ **Preferred** style sheets are used by the browser by default.

To make a preferred style sheet, set the `rel` attribute to `stylesheet` and set the `title` attribute.

Example: `<link rel="stylesheet" type="text/css" href="lecture_12_style2.css" title="style2" />`

If two or more `link` elements specify a preferred style sheet, the first one takes precedence.

- ▶ **Alternate** style sheets are not used by the browser by default, but may be selected by the user (from a menu in the browser).

To make an alternate style sheet, set the `rel` attribute to `alternate stylesheet` and set the `title` attribute.

Example: `<link rel="alternate stylesheet" type="text/css" href="lecture_12_style3.css" title="style3" />`

Alternate Style Sheets

Browsers should allow users to select from alternate style sheets. In FireFox (v2) this can be done by going to *View* and choosing the *Page Style* menu option.

It is possible to group several alternate style sheets (including any preferred style sheets) under a single style name using the `title` attribute. When a user selects a named style, the browser must apply all style sheets with that name (and no other alternate style sheets).

Example:

```
<link rel="stylesheet" href="mystyle1.css" title="mystyle" />  
<link rel="alternate stylesheet" href="mystyle2.css" title="mystyle" />  
<link rel="alternate stylesheet" href="mystyle3.css" title="mystyle" />
```

Media Types

One of the most important features of style sheets is that they specify how a document is to be presented on **different media**: on the screen, on paper, with a speech synthesizer, with a braille device, etc.

In XHTML Basic this is done using the **media** attribute of the `link` element.

Specifying the `media` attribute allows browsers to load and apply style sheets selectively. Browsers download the style sheets if they support style sheets; otherwise they ignore them.

The **media types** supported are: `screen`, `tty`, `tv`, `projection`, `handheld`, `print`, `braille`, `aural`, and `all`.

(More about this from Tom Worthington.)

Cascading

Cascading style sheet languages like CSS allow style information from several sources to be blended together.

With external style sheets, a **cascade** is defined by specifying a sequence of `link` elements (as in our examples). The style information is cascaded in the order the elements appear in the `head`. That is, if two rules set a value for the same element, the latter rule will take effect.

Other Sources of Style Sheets

In a browser there are three sources of style sheets:

- ▶ **Web Developers** who write style sheets like those we have considered so far (external, embedded and inline).

- ▶ **Browsers** which have style settings built into their software and define how each element is going to be displayed. These define the **default styles**.
- ▶ **Users** who can define their own style preferences by changing options in the browser software.

Precedence is given to these style sheets (by the browser) as follows:

- ▶ the web developer's style sheet overrides the user's style sheet,
- ▶ the user's style sheet overrides the browser's default styles.

Therefore the designer of a web page (and associated style sheet) has final say on how the page should be displayed.

Telling the Browser What's Important

It is possible to override the precedence rules of the browser by using the keyword `important`.

Example:

```
h1 {text-align : center;
    color : blue ! important;
    font-family : arial;
}
```

The browser will not override `important` styles. However, if two rules for the same element mark the same declaration as `important`, the usual precedence rules apply.

Inheritance

There is a hierarchy defined by the way XHTML tags are used.

Example:

```
<body>  
  <p>A paragraph with some <em>emphasised</em> text.</p>  
</body>
```

In this example the body tag is the **parent** of the p tag, and the em tag is the **child** of p.

In CSS, child tags **inherit** the style properties of their parents. Hence the style

```
body {color : blue;}
```

will be inherited by the p tag in the example and then by the em tag. The result is that all text in the body of the document will be blue (unless overridden by another style).

Examples:

- ▶ http://cs.anu.edu.au/student/comp2410/examples/lecture_12_example.html
- ▶ http://cs.anu.edu.au/student/comp2410/examples/lecture_12_style1.css
- ▶ http://cs.anu.edu.au/student/comp2410/examples/lecture_12_style2.css
- ▶ http://cs.anu.edu.au/student/comp2410/examples/lecture_12_style3.css

References:

- ▶ Darlington, Chapters 5 & 6.
- ▶ CSS1 – <http://www.w3.org/TR/REC-CSS1>
especially Section 1 <http://www.w3.org/TR/REC-CSS1#basic-concepts>
- ▶ CSS2 – <http://www.w3.org/TR/REC-CSS2>
- ▶ W3C CSS Home Page – <http://www.w3.org/Style/CSS>

Coming Lectures:

- ▶ More on Cascading Style Sheets.
- ▶ Web Servers, Proxies and Caches.