Overview

Unit 3 is worth 15% of your portfolio grade

Podcast - Introduction to Unit 3

Cascading style sheets (CSS) are one of the cornerstones of the Web, allowing you to make your page look and feel however you want it to look and feel, to work with multiple devices and outputs, and (very importantly from our perspective) to provide foundations that can be used by JavaScript to manipulate page elements in sophisticated ways—moving things, showing things, hiding things and changing how they look with relative ease.

In this unit you will apply styles to the pages you developed in Unit 2 to make them more attractive and more usable, bearing in mind the needs that you established in Unit 1.

Pass your mouse over this bit of text to get a hint of what CSS can do.

Things You Need to Know about First

You will need to know about writing well-structured XHTML, including tags, attributes, ids, divs, spans, and anchors.

Important: do not begin this task until you have completed Unit 1 and Unit 2, including your reflective learning diary for each unit.

Learning Outcomes

When you have completed this unit, you should be able to write well-structured, easily maintained, standards-compliant CSS code to present HTML pages in different ways.

Problem

Make code you created in Unit 2 beautiful and, if possible, more usable, **without** changing the HTML tags used in the <body> of the documents apart from, where necessary, adding or elements and, occasionally, extra attributes (particularly to specify IDs and classes) when required.

Note that you may (and should) change the <head> section and add or change attributes of existing tags, or use and elements to organize your text. The only notable exception to this rule is where feedback from the tutor or classmates on Unit 2 suggests necessary improvements. If you are not sure whether your change is suitable, a simple rule of thumb is that the content should not change, and you should only add tags that, without the aid CSS, make no observable change to the appearance of the document.

We strongly recommend that you keep your styles completely separate from your HTML, in a different document (technically, an *external style sheet*). Use of *inline* or *embedded* styles (you will be learning about this as you attempt tutorials) are fine for quick and dirty fixes, and may sometimes be useful to override external style sheets but, unless there is a good reason for you to do so that you can explain in your reflective learning diary, you are likely to lose marks for using them where an external file would suffice.

Process Guide

It would be helpful to **explore** some CSS-enabled sites beforehand, noting features that you like and don't like about them. Most modern sites use styles, but some make a point of it. Where possible, collect snippets of code that you might re-use, making **sure** that you record where they came from so that they can be properly ascribed: remember that we encourage and reward re-use but strongly discourage and punish plagiarism! You may want to collect things that change colours, styles, positions, and so on.

You will certainly need to **work through tutorials and examples**. We provide a few to get you started, but do feel free to go beyond these, and if you find something useful, please share it back with the community through the Landing bookmarks for this course.

Things you need to know about by the time you have finished this include

- HTML id and class attributes.
- CSS selectors, properties, and values.

- importing a style sheet.
- CSS classes, pseudo-classes, IDs.
- CSS positioning (absolute, relative, static, fixed).

There are many other things beside these that you may find useful or necessary as you progress through this topic, but these are the basics you will definitely need to know about.

CSS is a big subject, and there are many things to discover. We do not expect you to learn everything there is to know about it. While it is very helpful to remember how to write some of the more common types of CSS that affect things like fonts, sizes, colours, and positions, you don't need to remember everything. There are very few people in the world who could write any arbitrary bit of CSS without the aid of a reference. What you do need to know is how to find *what* you need *when* you need it. The more you use it, the easier it will become. As with the whole of the content of this course, *practice makes perfect*.

We suggest that you work on a local copy of your site before uploading the work to the SCIS server. You are strongly advised to keep your CSS code in at least one separate style sheet rather than inline or embedded (this will make sense to you soon!)

As always in this course, in your learning diary, you should explain why your changes are improvements by relating them to the personas and scenarios identified in Unit 1. This is not just about prettifying your site, but about doing it in a way that is informed by the design decisions you have already made. If, as a result of this, you need to make changes to Unit 1 documentation, you should explain what changes were made and why you made them.

Submitting Your Work

Create a separate CSS folder on your SCIS site into which you place your CSS file or files—we suggest that you call this folder “css,” “styles,” or “stylesheets.” Make sure that you update your HTML so that you link each page to your style sheet(s).

Important: create a zipped-up version of all of the site code (no need for the images) and save it as an attachment to your diary entry. Remember that you must do this at the end of each unit as part of your reflective learning diary process. This will help to provide us with a clearer image of how your site has developed over the course, what you have learned, and how you have developed your skills.

It should be fairly clear from looking at your site whether you have been successful, and you will receive no formal feedback at this point. However, please feel free to ask your tutor whether you are doing the right kind of thing. It would be sensible to ask specific questions about things that concern you as your tutor has only a very limited allocation of time for this and will likely spend no more than 5–10 minutes on the process. If you ask for general comments, do not expect deep feedback at this point! Also, feel free to explore (and comment on) the

work of others on the course, if available. You may get ideas from this, and it will help you to gauge roughly where your work lies in relation to others.

(Almost) Everything You Need to Know about CSS in a Few Short Paragraphs

Cascading style sheets are defined by a set of standards managed by the W3 Consortium that allow an author to fully specify the appearance and, to a limited extent, the behaviour of HTML (and some other XML-related language) pages. It allows you to create different style sheets for different media and ways of displaying content, such as web, mobile, voice, braille, or printed page.

CSS defines rules for the display of content. Because one CSS rule might often conflict with another (for instance, when using multiple style sheets or using your own styles on your local machine) there are strict specifications defining the priority of one rule over another—this is the origin of the term *cascading*—rules build on rules, build on rules in a cascade of potentially overlapping styles, each of which adds to or replaces the last.

CSS syntax and commands are completely different from those of HTML, but, like HTML, CSS is written in plain text.

The elements of CSS are

- a selector – typically a tag, ID, or class (other types of selector exist) which matches something in your HTML code. For example, **p** specifies styles applied to all paragraph tags, while **.aClassThatIInvented** refers to tags labelled with an attribute of **class="aClassThatIInvented"** (e.g., <p class="aClassThatIInvented"> or <h1 class="aClassThatIInvented">)

- one or more declarations, enclosed in curly brackets and separated by semi-colons - these are the 'language' elements of CSS that do the work. They consist of property:value pairs, a little like the properties and values of attributes in HTML. For example, **color:pink** is a declaration stating that things to which the selector applies should be pink.

For example,

TOP

```
p {color:blue; text-align:center;}
```

will make every paragraph in your document blue and centred, unless overridden by other styles. In this example there is one selector (p) and two declarations (color:blue and text-align:center).

Pass your mouse over this paragraph to see the kind of effect you would get with the style above.

It doesn't matter much to the browser whether you spread these over multiple lines or not, but they are generally easier to read if you use a separate line for each declaration (**note**: there are other slightly different ways to represent and manipulate CSS, notably when using inline CSS or JavaScript, but we will ignore them for the purposes of this brief tutorial).

CSS may be written and applied within your document in three different ways:

- inline (embedded in the page as attributes of HTML tags)
- embedded (provided as a block of commands inline with the HTML, typically in the section of the document)
- external (created in separate external text files that are linked to from the HTML documents that use it. **This is the far preferred method to use in most instances**.

To attach a style sheet to an HTML document, in the <head> section of the document create a tag like this:

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

(This assumes that you have created a style sheet in the same directory as your HTML file called "mystylesheet.css"; however, it is better to create a separate directory for your style sheets and to give them more meaningful names, so that you or someone else can more easily maintain your site later.)

For information: you will come across the notion of *type* again when we start to look at JavaScript. The type refers to the MIME-type of the document. MIME is a set of standards that was originally designed for email so that email clients would know what kind of a document was in an attachment—for example, a picture, a Word document, a program, or a

compressed file. Without such information, your email client might try to open an image in a text editor or load an executable file into a word processor.

Web servers and browser also use MIME types extensively. A web server can tell a web browser what type of file it is sending so that the browser can deal with it appropriately, and the browser can tell the server what it can and cannot accept—handy if you are using a text-only or voice browser that cannot display images, for example, because it saves effort at both ends if the server knows there is no point in sending it. You can recognize MIME types because they have two parts: a general classifier such as 'text', 'image' or 'application' and a sub-type, such as 'jpeg', 'html', 'css' or 'javascript'. So, for instance, text/css' means it is a style sheet in text format. 'image/jpeg' means it is an image in jpeg format (the sort usually created by digital cameras).

We strongly advise that you use only external style sheets wherever possible; otherwise, maintenance becomes a nightmare. External style sheets can be easily used by many documents across the whole site, so you need to edit them only once. Inline and embedded styles are very much harder to manage once your site starts to grow and should only be used in very special cases. These are usually dictated by the order of parsing: external CSS is the first to be processed, then embedded, then inline, so inline styles override the rest.

CSS can define how tags are presented; for instance, you can specify how all <h1> tags are displayed:

```
h1 {font-family:arial;}
```

Alternatively (and more flexibly) CSS can be used in conjunction with HTML attributes that supply an ID (a distinct identifier for a given tag or tags), using a hash symbol to tell the CSS interpreter what you mean. For instance, if you want a particular heading to appear differently from the rest, that is easily done: In the HTML, you could create this line:

```
<h1 id="mySpecialHeading"> Hello world </h1>
```

and then create some CSS to define how 'mySpecialHeading' should appear, e.g.,

```
#mySpecialHeading{margin-left:20px;}
```

This would result in the following:

Hello world

Alternatively (and maybe more flexible still), you can define CSS classes of things that share some aspect of appearance or behaviour. For example, you might define a class that you call 'important' which turns the text red, then or <h1 class="important"> would each appear in red text, e,g.,

```
.important {color:red;}
```

So, Hello world (unstyled) would look like this:

Hello world

but, with the style applied, it would look like this:

Hello world

For greater flexibility still, you can use HTML and tags with associated IDs or classes. These tags simply tell the browser that you are treating either a block (in the case of) or inline element (in the case of) as a distinct element, without making any assertions about its structural role within the page. Whereas most tags have semantic purpose (e.g. tells the browser it is a paragraph, <h1> that it is a heading, <a> that it is an anchor, that it should be strongly emphasized), and tags have no inherent meaning and can be used as you wish in a more flexible way. So, for example,

```
This sentence contains some <span class="bluetext"> blue text. </span>
```

can make use of the CSS

```
.bluetext{color:blue;}
```

to look like this:

This sentence contains some blue text.

TOP

CSS can also be used to include and format other types of content, including images for backgrounds and even text.

There are plenty of other more advanced CSS selectors, pseudo classes, and neat tricks, as well as useful ways to combine selectors to limit or extend their use, but this is a simple overview, and we do not want to scare you—yet! We do want you to learn how to use them, though, and you will find plenty of tutorials and examples in the resources provided. If they don't suit you, seek others and share them with the rest of us. If you're interested, we used one of those tricks when creating the bits of text on this page that changed when

you moved your mouse over them: the :hover pseudo-class, to be exact. That turns out to be very useful when you want to create effects like drop-down menus, for instance.

CSS comes in three versions, each adding new and richer functionality to the former version. Most browsers behave in a similar way when presented with CSS 1; there are some moderately complete implementations of CSS 2, but (at the time of writing) few, if any, browsers implement a complete set of CSS 3 selectors and declarations, though almost all implement some of them. This means that, even more than for HTML, you must be very careful to test your layouts and designs on as many browsers as you can and you must constantly remain vigilant and sensitive to the possibility that what looks beautiful and works brilliantly on your own machine may not work the same for everyone else.

OK, we clearly lied when we suggested that this is almost everything you need to know! There's a great deal more to discover, but this should get you started. Remember, you do not need to know every single declaration for every kind of style. As long as you know roughly what you are trying to do and what CSS makes possible, there are plenty of online, computer-based, and paper references to help find the right declarations for what you want to do.

For more complex examples, don't be afraid to copy what others have done, assuming copyright restrictions allow it, but don't forget to include a comment (in CSS, /* this is a comment */—you'll need to learn about that) to say where it came from and any other required copyright information. **Very important: we encourage and reward appropriate, effective and legal re-use but *we will heavily penalize improperly cited or uncited use* (plagiarism), with punishments ranging from failure up to and including exclusion from the course or even from the university. We have many tools and procedures for this, and we use them. The rule of thumb: If in doubt, cite it**.

Creating effective CSS can be a long and complex task once you get beyond the basics, but we hope and expect that you will find it very rewarding and interesting.

Indicative Grading Criteria

Grade	Criteria

A

- an attractive set of pages that effectively separate the content, structure, and presentation of the site
- sophisticated use of a wide range of CSS options to format, style, and position content on the page, using many different selectors, combinations of selectors and declarations in a manner that enhances both communication and presentation as well as demonstrating mastery of the technology
- close and explicit attention to the needs identified in Unit 1
- no cross-browser compatibility issues and degrades gracefully if browsers do not fully support CSS
- well-presented and maintainable CSS code with good use of comments and structure
- certainly making good use CSS 2, likely using a number of CSS 3 options

A typical style sheet might extend over several pages, and it is quite likely that you may include more than one style sheet.

B

- an attractive set of pages that separate the content, structure and presentation of the site
- good use of a wide range of common CSS features to format, style, and position content on the page, including an assortment of selectors and declarations, all correctly and maintainably used
- explicit attention to the needs identified in Unit 1
- no significant issues running the code in different browsers and degrades gracefully in non-compliant browsers
- clearly written code, mainly CSS 2 with some CSS 3 used here and there

A typical style sheet might be a few pages long.

C

- a set of pages that separate the content, structure and presentation of the site
- use of several basic CSS features to format, style, and position content on the page
- no major mismatches with the needs identified in Unit 1
- few cross-browser compatibility issues—works in the two or three most popular browsers
- some issues with formatting of code but mostly maintainable
- mostly a mix of CSS 1 and CSS 2

A typical style sheet might be a couple of pages of CSS.

D

- a set of pages that mostly separate the content, structure and presentation of the site and that somewhat improve the appearance of the pages
- limited or poor use of CSS to format, style, and position content on the page
- weaknesses in choice of appropriate selectors or declarations
- CSS adds little to improve communication or presentation of the page
- might be some mismatches with the needs identified in Unit 1
- some cross-browser compatibility issues
- poorly formatted CSS code, some issues with maintainability
- mostly CSS 1, with just a few CSS 2 elements

External style sheet probably quite brief—a page or two and may not fully match with the HTML content.