COM1008: Web and Internet Technology

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

Exercise sheet 3

1. Aims

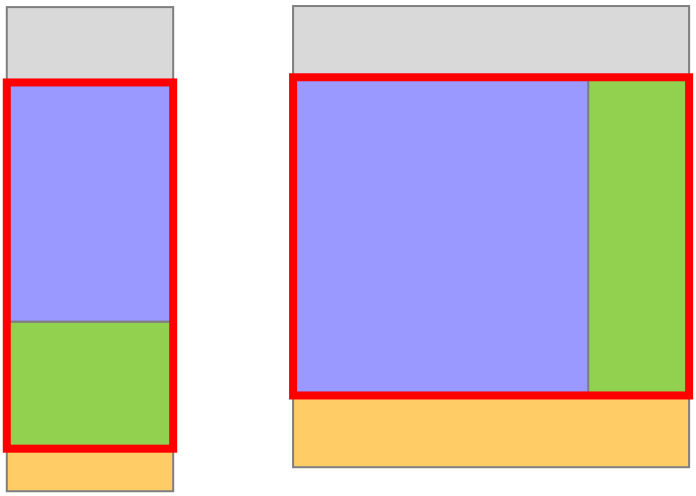
- To practise using Responsive Web Design
- To build up your expertise in using the features of HTML5 and CSS
- To show you how to put your files onto the Department's Web server

As per last week, you may need to refer to https://developer.mozilla.org/en-US/Learn/HTML to work out how to do some of the following.

2. Responsive web design

There is no right or wrong answer to the following exercises. The aim is to practise using some of the responsive web design ideas.

1. Create a Web page that has four areas: a header, a navigation area, a main content area and a footer. The four areas should be wrapped in a element. Random text can be added to each of the areas. For the navigation area, add three random pieces of text to represent three links, but do not add anchor tags to these. For the main content area, add a header and three paragraphs of random text. Do not write the CSS yet.
2. Attach a CSS reset or normalize to your Web page. You choose which. This will change the display of your HTML file.
3. Now create your own CSS file to style the page. Set the width of the wrapper to 320px, i.e. in a CSS file: `#wrapper { width: 320px; margin: 0 auto; }`
   This will make the display of your page as narrow as that of a smart phone and also centre this on the display. Add a background-color to the body tag and to the wrapper so the extent of the wrapper on the screen is clear to see. Then add some rules for each of the areas, e.g. margins and padding, fonts and colours, until you are happy with the layout.
   In doing this, you should use a flexible grid for the layout, i.e. use % to describe widths of areas relative to the parent width, e.g. widths of margins and padding.
   As you resize the browser window the margins of the wrapper will change so that the wrapper stays centred on the screen, but the width of the wrapper itself will not change. Do not include images and do not use media queries yet.
4. Change the width of the wrapper area so that it is expressed as a %, e.g. width: 100% instead of a fixed width of 320px. Your page will grow to the size of the browser window and the text display should now be 'liquid' as the browser window is resized, by which I mean it reflows to fit the available space. However, undoubtedly the display looks better for the 320px width it was originally designed for rather than the large desktop display. Some media queries will now be used to address this. But first, add a min-width to the wrapper:
   `#wrapper { min-width: 320px; width: 100%; margin: 0 auto; }`
   This will stop the wrapper becoming less than 320px in width.
5. Add a media query so that aspects of the design change when the width of the browser window is greater than or equal to 750px. For example, add some extra CSS rules to this media query so that the size of the header and its text is bigger. Perhaps change the width of the wrapper to 80%.
6. Add a media query so that aspects of the design change when the width of the browser window is greater than or equal to 970px. Introduce an extra column to the right hand side of the main



(a) **(b)**

Figure 1. (a) width<970px; different areas are stacked vertically; (b) width>=970px; main content area is split into two separate side-by-side areas of content nested inside the main element

content area. You can choose to do this in one of two ways: (i) use float; (ii) use flexbox. Then make sure this extra content is at the bottom of the main content area for browser window widths of less than 970px. Figure 1 gives the general layout for the two designs. (There is a nice article on RWD patterns at https://developers.google.com/web/fundamentals/layouts/rwd-patterns/?hl=en. Also, Firefox describe use of the flexbox at https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Using_CSS_flexible_boxes)

7. Set a max-width of 1170px for the wrapper. This will mean that for very high resolution monitors, all that will happen when the browser window gets bigger than 1170px is that the margin either side of the wrapper will grow larger. The arrangement of the contents of the wrapper will remain the same and be centred in the browser window. This is currently an accepted practice for Web site design, rather than produce extra layout designs for larger and larger displays.
8. Add an image to the page. At a width of 320-749px, the image should be as wide as the wrapper. For larger page sizes it should occupy only a small proportion of the width of the main element (or, more likely, a proportion of the width of the left hand column for the two column display shown in Figure 1). You choose values that look right.
9. Test the web page using Google's mobile-friendly test: https://www.google.com/webmasters/tools/mobile-friendly/
10. Progressively enhance the Web site with some CSS3 features. For example, add a text-shadow effect and a transition or transformation effect to one or more of the elements.
11. Now duplicate the page three times and add links to each of the pages in the navigation area.
12. Add extra items to the navigation area so that there are now seven links. You don't have to create extra pages – just link to the same pages you have already created; that will suffice for the purposes of this exercise. The layout of the navigation area may now 'break' (look wrong) when the browser window width becomes small or the items in the navigation area will wrap, increasing the size of the navigation area so that it dominates the small screen size. This is a common problem when laying out navigation areas and different solutions have been proposed. The article at http://responsivenavigation.net/index.html presents a range of these approaches. Read this article and decide which approach you favour. Then try to implement that approach.

Further reading

https://developers.google.com/webmasters/mobile-sites/?utm_source=wmc-blog&utm_medium=referral&utm_campaign=mobile-friendly

http://webdesignerwall.com/tutorials/responsive-design-with-css3-media-queries/comment-page-1

http://www.smashingmagazine.com/2013/03/building-a-better-responsive-website/

http://www.html5rocks.com/en/mobile/responsivedesign/

3. Uploading your files to the Department of Computer Science Web Server

(I'll use 'COM' to refer to 'the Department of Computer Science'.)

In a typical Web design setup, files are developed away from the final Web server where they will reside. Often ftp (file transfer protocol) is used to upload the final files to the Web server, where they are then visible to the world. We simulate this process by providing you a way to link to the COM Web server and to copy the files into the relevant folder.

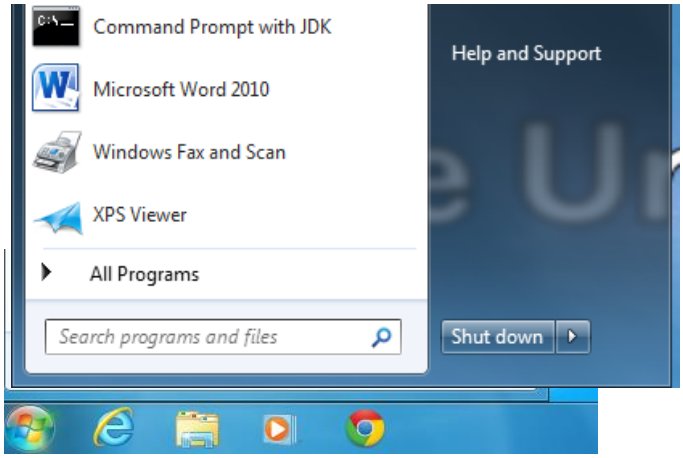


Figure 2. The search area box

From the *Windows start menu* on a CiCS managed desktop machine (i.e. the ones in the Diamond or in the IC), type the following into the 'search programs and files' area (as shown in Figure 2):

\\stustore.dcs.shef.ac.uk\mypublic_html

In the pop-up login box (see Figure 3), you need to type the following using your COM password[1] (**not** your CICS password):

User name: **windows*your_username***

Password: ***your_COM_password***

where *your_username* is replaced by your username and *your_COM_password* is replaced by your COM password (not your CiCS password). (Note: The '**windows**' part is important, as it changes the domain to the COM windows network server, as can be seen in the lower picture in Figure 3.)

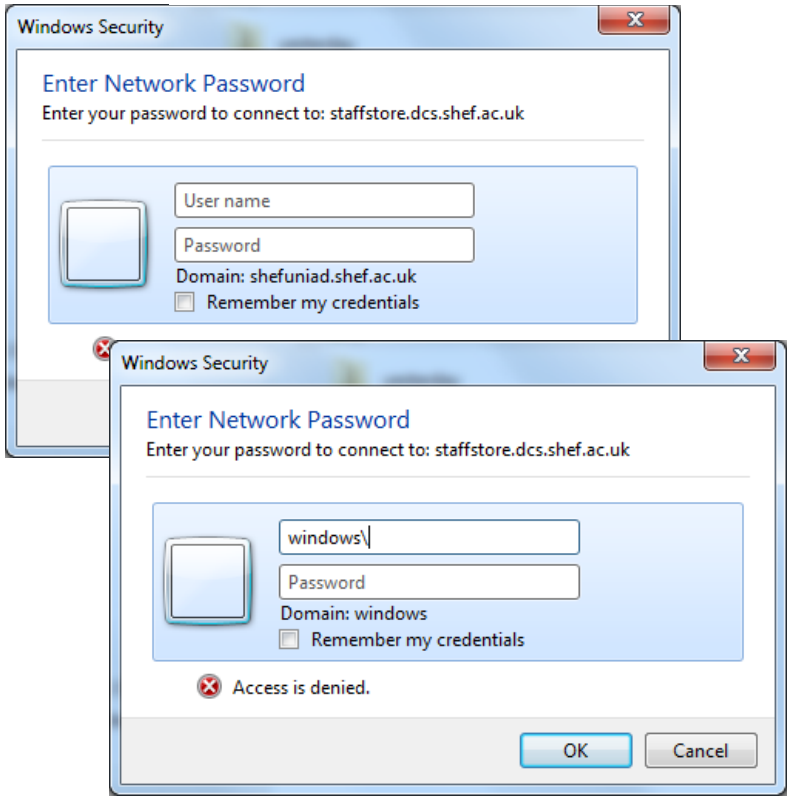


Figure 3. The login window. (a) first appearance; (b) as you begin to type 'windows\', notice that the domain changes. The rest of the input depends on *your_username* and *your_COM_password*

[1] If you have forgotten your COM login details, then go to https://www.dcs.shef.ac.uk/password

After this, a window will open that shows the contents of the folder mypublic_html. You can then copy your files and folders (e.g. html files and subfolders for images) to this folder. They are now on the COM Web server and you can use a full URL to access them:

`http://stuwww.dcs.shef.ac.uk/people/`*`your_e-mail_name`*`/`*`filename`*`.html`

where *your_e-mail_name* is replaced with the part of your CICS e-mail address that appears before the '@' symbol. So, if you are jsmith1@sheffield.ac.uk, then *your_e-mail_name* is jsmith1 and the URL would be:

`http://stuwww.dcs.shef.ac.uk/people/jsmith1/filename.html`

(Note: The mypublic_html folder may contain other subfolders called campus_only and dept_only. If this is the case then any files you store in those folders will only be accessible to users logged in to the University campus network or to the COM network, respectively. If these folders do not exist, you can create them and they will automatically assume those properties.)

Appendix A. Your COM filestore

As you are a student in the Department of Computer Science (COM), you have both of the following:

- a campus account managed by CiCS
- a COM account managed by the COM support team

When you are out and about on the campus, you use the CiCS account to login and you have a certain amount of filestore, with U: as your home drive. On the COM network (which you can log into in Regent Court or in the Edgar Allen Building), you also have a home filestore, but the drive letter is H:.

Further details about accessing your COM filestore from CiCS machines across the campus or from home can be found at:
https://guide.dcs.shef.ac.uk/Teachnet/AccessingHomeDirectoryFromHomeOrFromOnCampus

Within the COM filestore, you have a set of folders that are automatically set up. One of those is h:\public_html. This folder can be 'seen' by the COM Web server. mypublic_html, as described above in Section 2, is known as a *Windows share*. It is an alternative route to your personal `h:\public_html` directory. When you copy files to mypublic_html, you are actually copying them to your personal h:\public_html folder[2], but there is an important difference.

Why is mypublic_html needed?

In order for the world to access files on the Web server, the correct *access permissions* need to be set for the files. The world needs to be able to 'read' the files. You need to be able to read and write them. Writing means you can create files, copy them from somewhere else and edit existing files.

When files are placed in mypublic_html, the access permissions for the files are automatically set so that the files are world readable, i.e. anyone in the world can see them, as long as they have the full URL. This Windows share has been configured by the COM network manager to automatically set the correct access permissions, which are read from the world and read/write for you. Be careful what you put in the mypublic_html folder, as everything in there is world readable.

If you were to copy the html and CSS files directly to h:\public_html, then they would not be world readable. You would have to change the access permissions of the file yourself to make them world readable. You can do this with Windows file explorer. Highlight the file and right click to produce a

[2] This is similar to the idea of 'My Documents' on a personal Windows machine, which actually maps to a specific folder on the hard drive of a Windows-based PC.

menu from which the option ‘properties’ should be selected. This launches a properties window. Select the Security tab in this window. Select the Edit button. Highlight the group ‘Everyone’ and set the access permissions to read and execute. Now everyone in the world can read this file, i.e. they can display it in a browser. If you highlight your own user name, you can change the permissions for yourself to give yourself full editing rights. You can see that this is a faff. It is better to use mypublic_html.

A further issue is that when you open the file from h:\public_html in certain text editors, e.g. jEdit, they make a copy of the existing file and then save the edited file back to the old name when you select the save option. This effectively creates a ‘new’ file, with the default access permissions (i.e. no access, which is very annoying as you then have to manually change them all again). It is obviously better to use mypublic_html, which automatically sets the right access permissions.

As with files, folders in h:\public_html have to have their access permissions set to make them accessible via the Web server. From Windows file explorer, highlight the folder, right click and then change the ‘properties’ as described above. ‘Execute’ must be set for folders for ‘everyone’, whereas often ‘read’ permissions is all that is needed for individual files. Again, it is better to use mypublic_html, which automatically sets all the correct permissions for files and folders.