COMP112

Web Development & Digital Media

Semester 1 2017

Laboratory Manual

UNIVERSITY of OTAGO
SAPERE AUDE
Te Whare Wānanga o Otāgo

Table of Contents

Lecture and Laboratory Schedule

Week Beginning	Lecture Title	Lecture / Lab Number	Laboratory Title
27 February	Introduction	1	Introduction to OSX
	HTML I Basics	2	HTML I Introduction (1%)
6 March	HTML II Links	3	HTML II Links (1%)
	CSS I	4	Publishing your Web Page (1%)
13 March	Web Technologies	5	CSS I (1%)
	HTML III Tables	6	HTML III Tables (1%)
20 March	CSS II and CSS III	7	CSS II(1%)
	How Google Works	8	Catch Up
27 March	Usability I	9	CSS III (1%)
	Usability II	10	CSS IV (1%)
3 April	Security	11	Make a Web Site I (1%)
	Ethics	12	Make a Web Site II (2%)
10 April	Sound	13	Project Plan Marking (3%)
	Video	14	Catch Up
	Easter Break : 14th April - 23rd April		
24 April	No Lecture	15	HTML IV Forms I (1%)
	HTML IV Forms	16	HTML V Forms II(1%)
1 May	JavaScript	17	JavaScript (1%)
	Accessibility	18	Accessibility Challenge (2%)
8 May	How Colour is Represented	19	Graphics I(1%)
	Graphic Formats	20	Graphics II(1%)
15 May	Cameras and Scanners	21	Graphics III(1%)
	Flash	22	Project Submission
22 May	HTML5/CSS3	23	Site Update (1%)
	Server-side	24	Responsive Web Pages and Frameworks (1%)
29 May	Review	25	PHP (1%)
		26	Catch Up

Course Information

Check list

Ensure that you know:

- ✔ Where and when your lectures are.
- ✔ Where and when your lab streams are.

Course delivery

There are two one-hour lectures and two two-hour laboratories each week, see the schedule on page 5 for full details. The labs are carefully designed so that assuming you have attended the relevant lecture and looked through the lab beforehand you should be able to complete them in the time allocated. In reality you will find that some labs may take you a little longer than two hours and some may take you less. To help you catch up should you fall behind there are 2 'Catch-up' labs as well as the Easter lab and the Project Submission lab during the semester where no lab activities are set. You should also expect to need an additional 3 or 4 hours per week to complete course readings and review lab and lecture material.

Text book

There is no set text for this course. This lab book will make use of online resources especially those at www.w3schools.com and www.w3.org. *Dive into HTML5* is also an excellent resource; it is available for free at www.diveintohtml5.info

Assessment weighting

40% of your total grade comes from internal assessment:

- 18% assignment.
- 22% lab completion

60% of your grade comes from your final exam.

Course Feedback

We welcome both positive and negative feedback on the course, we want to make COMP112 the best course it can be. If you have any comments or suggestions concerning the course please feel free to contact the teaching staff, your Class Representative, the Course Coordinator or the Head of Department. All communication will be dealt with confidentially.

Lab Work

Please bring this lab book with you to each session in the laboratory, you will need it to successfully

complete the laboratory tasks. Please also remember to bring your student id card, we need to scan it to record lab completion.

Checkpoints

Checkpoints are clearly indicated in the lab book As you reach one have a demonstrator check your work. Checkpoints allow demonstrators to give you feedback at appropriate stages throughout your lab work.

Review activities

Review activities are designed to be completed without demonstrator assistance so that you can apply the material covered in the lab and reinforce your understanding of it. You *will* encounter problems, things *won't* behave as you expect first time. The most important skill to develop in web development is the ability to locate and fix errors unaided.
When you ask to have a review activity checked a demonstrator will go over your work with you and may make suggestions on fixes or improvements to make before the work is signed off as complete. There is no limit to the number of times you can have your work checked, but you will gain personal satisfaction from knowing you got it right first (or second) time.

Lab Completion

Each lab completed **on time** over the course of the semester earns you marks (either 1% or 2%) toward your final grade to a total of 22%.
In order for your lab work to be considered complete and on time, it must be signed off as complete **before** your the next-but-one lab starts. For instance Lab 3 will be marked as Late once Lab 5 starts. The current Lab number is prominently displayed on the whiteboard at the front of the lab.

Late Submissions

You may have a total of two labs marked *without any penalty* even if they are late; subsequent late labs will be marked however they will incur a 50% penalty. Labs more than 3 weeks late will receive no marks. The two lab late allowance applies only to lab completion, not to other assessed work.

If there is a legitimate need for an extension covering more than two labs you can apply to the lab administrator or the course coordinator for one.

The two lab allowance is designed for you to manage legitimate short-term impairment or non-attendance and should not be used for trivial reasons. Don't use them unless you need to!

Assignment

There is one submitted assignment worth 18% (see page 11 for details). Late assignment submissions are penalised at 10% per working day or part thereof.

Assignment and Lab extensions

If you have been significantly affected by illness or other circumstances before your assignment or lab is due you may apply to the course coordinator for an extension.

Extensions are only granted to students who have supporting evidence for their request (medical certificate or otherwise).

It is most important that you make contact with us as soon as circumstances affecting your ability to submit your assignment or lab on time arise. If you are significantly incapacitated, you are welcome to have someone contact us on your behalf. You can make contact by phone, email, letter or in person. Please state your name and ID number in any correspondence.

Copying and Plagiarism

It is unfortunate that some students are willing to cheat to succeed at university. Don't be one of these people and don't enable someone else to cheat. The penalties for cheating at University can be severe.

"The University has clear policy with regard to copying the work of others:

Dishonest practice is seeking to gain for yourself, or assisting another person to gain, an academic advantage by deception or other unfair means. The most common form of dishonest practice is plagiarism.

Dishonest practice in relation to work submitted for assessment (including all course work, tests and examinations) is taken very seriously at the University of Otago.

All students have a responsibility to understand the requirements that apply to particular assessments and also to be aware of acceptable academic practice regarding the use of material prepared by others. Therefore it is important to be familiar with the rules surrounding dishonest practice at the University of Otago; they may be different from the rules in your previous place of study."*

- ✔ Don't give your markup to another student.
- ✔ Don't allow anybody else to work on a computer under your username.
- ✔ Don't tell your username and password to anyone.
- ✔ Don't share the workload of activities or projects with other people – we are measuring your individual performance and expect that your work is produced by you alone.
- ✔ Don't copy any part of the work of others, classmates or an internet site for instance.
- ✔ Don't have a friend tell you how to complete your work.
- ✔ All markup you produce as part of this course should be written by you. It is not appropriate to use an application like DreamWeaver or FrontPage to generate markup for you. Generated markup is not marked.
- ✔ All **HTML and** all **CSS of** all **sites you produce for COMP112 must be original work created by you or material supplied by us.**

*See the University web site **www.otago.ac.nz/study/plagiarism/index.html** for more information.

CHECK LIST

Assignment (18%)

Overview

To produce a fully functional web site that is easy to use, error free and uses valid HTML5 to mark up the content and valid CSS to control the layout. The content will be supplied. Your task is to design, build, test and deploy the web site. The assignment is split into two parts, the Design phase and the Build phase.

Due Dates

Part A(3%) "The Plan" is marked during your Lab 13.
Part B(15%), "The Completed Site" must be submitted before midnight on the Friday 19th May.

Submission:

- ✔ **Publish your site on the Computer Science web server**. We should be able to access it using the following URL (where *username* is your COMP112 username):
  `http://csnet.otago.ac.nz/username/assignment/index.html`
- ✔ A compressed (.zip) version of your complete site saved as `<username>.zip` at:
  `http://csnet.otago.ac.nz/username/assignment/zip/`
- ✔ Include unchanged copies of your marked Plan in
  `http://csnet.otago.ac.nz/username/assignment/docs/`

Do not alter your published version after the submission deadline. The last modified date of your project directory is considered to be the date your assignment was submitted.

Project Tasks

Part A

The task for Part A is to develop a detailed plan for how you will present the supplied content via a browser. That is, how will your website *look* and *behave* when it is completed? Your plan should answer such questions as:

- What content will be on which pages?
- How will the pages relate to one another in the site?
- How will the pages look? Colours? Typefaces? Layout? Size?
- How will the page react to the browser window being resized?
- How will the navigation work? How will I know where I am in the site?
- How will I recognise links? How will I know if I have already been there?

To answer these and other questions you will need to produce a series of documents including at the very least:

- A site Map - to show the hierarchal structure of the site and the links between pages.
- Wireframes – documents to show how you intend to break up the visual space.
- Detailed design documents – to show how a finished page will actually look.

See Lab 11 for more details on the documents required.

Part B

The task for Part B is to use the supplied content to make a complete website to your original design and conforming to the following specifications:

- ✔ **Any images that we do not supply must be appropriately licensed and referenced on the web page it is used on and in a file in your docs directory – see above.**
- ✔ Your entire site should consist of no more than 10 and no fewer than 4 html files.
- ✔ Your entire site should be no more than 3MB in total with no one page being larger than 500KB.
- ✔ You must use the design that you had marked off in the Project Planning Lab 13
- ✔ The file that is the entry point for your site should be called index.html.
- ✔ Your COMP112 username and the file name should appear in comment tags in the `<head>` section of each of your HTML pages and at the top of your CSS file.
- ✔ Your site should behave as expected in Firefox on OS X.
- ✔ Where possible you should use CSS to control your layout.
- ✔ Your site should include a form that gathers contact and other useful information and emails it to your student email account. You should use the .cgi script you used in labs. (The form and styling should not be a direct copy of the form given in labs, there are marks for originality.)
- ✔ You site should use JavaScript. The JavaScript should be in a linked file and degrade gracefully. That is, if JavaScript is turned off (or is not supported) then the web page should still work and look ok.

Required standards:

- ✔ Your entire site must validate as HTML 5
- ✔ Your CSS must validate as CSS 3, with no errors or warnings excepting those triggered by the use of CSS 3 attribute overloading.
- ✔ You site must pass WCAG Priority A.

Additional marking considerations:

- ✔ Relevance and suitability of features used.
- ✔ Markup and code layout.
- ✔ Sensible use of comments.
- ✔ Efficient and sensible use of markup and code.
- ✔ Navigation implementation.

Hints and Tips:

- **Back up your work regularly**.
- Skills for this project have been covered throughout the course so checking back through previous labs and lectures will be helpful.
- Ensure your work displays an understanding of both the practical skills taught in labs and the concepts discussed in lectures especially concerning standards, semantic markup, accessibility and usability.
- You should not copy anything directly from the lab book or any other source. Adapt material so that it is appropriate for your project use and shows an understanding of the underlying concepts.

Suggested Production Workflow:

(Part A)

Plan / Design the Site

It's no use starting to make something if you don't know what you are trying to make. Making it up as you go along is all very well for avant-garde Lego creations but it doesn't work well for web development. Don't be afraid to look at other web sites for inspiration. Obviously don't copy anyone else's design directly, but you can see what features and looks you like. You will also see things you don't like, so you can actively avoid including them. In the end, the more detail you can include in your plan the easier you site will be to build and the better it will end up being.

(Part B)

Markup the Content

This should be familiar ground by now. First make a template page that includes all the common markup and then using that as a base insert and markup the content you have previously created. Your markup will of course meet the appropriate standards and use the principles of semantic markup. You should also make sure your files are properly commented and indented.

Apply the Style

Make it look pretty and functionally obvious with valid, commented, well organised and properly indented CSS .

Add the Functionality

Add behaviour with JavaScript, but remember to make sure your site still works with JavaScript switched off.

Test

Make sure you leave yourself sufficient time to go through the *test* → *fix* → *test* → *evolve* → *test* cycle many more times that you thought you would need to.

Common Problems

Here is an email I sent last year outlining the places where students commonly lost marks:

Not proof reading HTML or CSS.

The first thing you write shouldn't be the thing you submit. You should read and re-read your HTML and CSS several (many many) times. On each reading you will be able to make it a bit better.

Insufficient/Poor HTML

Some students used very little HTML at all making it hard to award high marks for this. We need to see more than an `<h1>`, `<h2>`, a few `<p>`s and a list. The use of HTML to markup the content is like an essay, it can either be very shallow and only touch on the main features or it can go a bit deeper.

Too many divs. Divs around `<h1>`s and `<p>`s and `<ul>`s for no good reason. Re-reading should eliminate these.

Insufficient/Poor CSS

The marks available for your CSS are for layout, not just changing the colour of text and backgrounds. So you need to do a bit of laying out. Like the HTML, this is your opportunity to show us your skill with CSS. No need to go wild but don't rely totally on the standard flow, use float, and inline or relative positioning, appropriately.

Unused or ineffective lines of CSS. Unhelpful id and class names. Many ids where one class would

do. Again, a proof reading issue.

Validation

Many people lost marks through their pages not validating properly. CSS Warnings in particular were a problem.

Images

Many people resized images with the HTML attributes width and height. Don't do it. Make the image the correct size in the first place. The width and size attributes should reflect the image's actual size.

Form

Most people just copy and pasted the form from the lab, this was not 'original material' and attracted few marks. This was another chance to show what you could do with HTML. Make an original form that uses inputs other than those we showed you in the lab.

All forms need comments to say what they do; almost no student's form did.

CHECK LIST

Lab 1 Introduction to OS X

Reference

Please make sure you have read the Course Information section of this lab book.

Check list

Ensure that you are confident doing the following:

- ✔ Logging on and off
- ✔ Using DemoCall
- ✔ Changing your password
- ✔ Creating new directories
- ✔ Renaming files and directories
- ✔ Using keyboard short-cuts
- ✔ Using the Dock
- ✔ Saving your work
- ✔ Printing a document
- ✔ Finding files and directories
- ✔ Archiving files and directories
- ✔ Customising your user-code

Introduction

Each lab session you will work through the relevant section of this book. Demonstrators are there to assist you. You should attract their attention by using **DemoCall**, an application that queues requests for help (we will show you how to do this in this lab). In this lab you will become familiar with the use of the Macintosh OS X desktop and the basic features of the operating system. Note: OS is short for Operating System and the X is the Roman numeral for 10 so OS X means Operating System 10.

Please make sure you have read the Course Information section of this lab book.

Activity 1.1 Logging On

Every time you come to a lab you will need to log on to a computer. When you do this you will be connected from the machine you are using to our server where all the student directories are stored. It is important that you make sure you are logged on correctly to ensure the safety of your files.

NOTE: Your ITS username will not work here.

The machines in the labs are normally not turned off, but they will start to display a screen saver if they have not been used for a while.

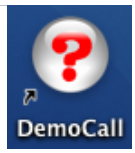
1 **Click the mouse button and something should appear on the screen. If nothing happens then it may have been turned off. You will find the startup button on the back of the machine; look for the ⏻ symbol. Ask a demonstrator for help if nothing happens.**

2 **Once your machine is running you should see the login screen:**

3 **Enter your Username.**
It should be your first initial and your whole last name (e.g. if your name is Jane Doe your username is "jdoe").
Remember, your ITS username will not work in this lab.

4 **Enter your Password.**
Initially your password is your student ID number.
Your machine will now log you on and you will see the Mac OS X Desktop.

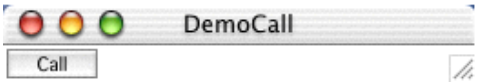
Activity 1.2 Using DemoCall to get help



You will notice that on the Desktop there is a red question mark. This icon starts the DemoCall application which is used to queue student requests for help.
Always use DemoCall to get a demonstrator's help. It is the fairest way for us to allocate the available demonstrator resources and ensures that every student has fair and equal access to demonstrator time.

To run DemoCall

1 **Double-click on the DemoCall icon to start the application. A window similar to the one below will appear.**



2 **To get help, click on the Call button.**
Your request will be placed in the queue and a demonstrator will be with you as soon as possible. Your place in the queue is indicated by your computer number, which will appear in red in the DemoCall window.

3 **When a demonstrator arrives, or if you change your mind about needing to see a demonstrator, click on the Cancel button.**

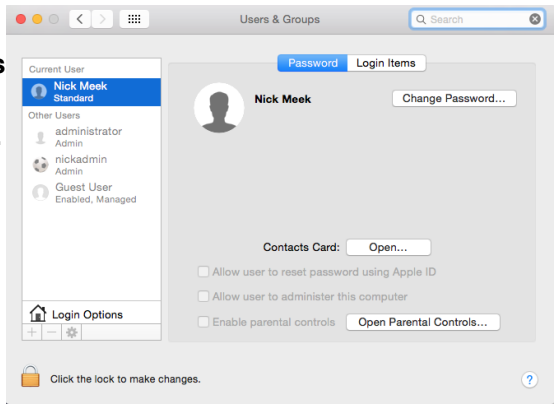
Activity 1.3 Changing your password

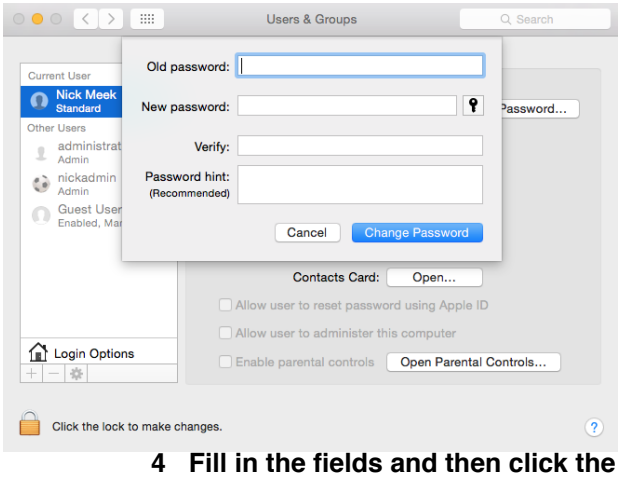
You **must** change your password from your ID number. With an insecure password anybody can access your personal directory and delete or copy your work.

1 **Open** System Preferences.

2 **Click the** Users & Groups **icon, it is in the System section.**

3 **Click the** Change Password **button.**





4 **Fill in the fields and then click the Change Password button.**

5

To be accepted your new password must fulfil the following conditions:

- Use at least 8 characters for your password.
- Use a combination which - within the first eight positions - has a mixture of upper and/or lower case letters, digits, and at least one of the following special characters: ! / $ % & * ( ) _ - = + : ; , .
- See www.cs.otago.ac.nz/resreg/goodpw.php for more details

6 **When you next log on you will need to use your new password.**

Do not write your password in your lab book.

Change your password regularly to reduce the risk of others knowing it.

You should never let anyone else work under your user-code.

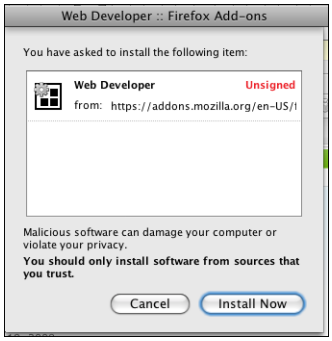
Activity 1.4 Firefox and Chrome Extensions for Web Developers

Firefox and Chrome can have plug-in extensions added to it that supply additional functionality. A particularly useful extension for web developers is the Web Developers' Toolbar. You will find it *very* useful during this course, install it now.

1 **Open Firefox.**

2 **Navigate to: http://chrispederick.com/work/web-developer/ and click the Download For Firefox button.**

3 **Click on the Install Now button.**



4 **You will be prompted to restart Firefox, do so.**

5 **When Firefox restarts you will see a new toolbar.**

6 **Navigate to the COMP112 homepage (www.cs.otago.ac.nz/comp112) and try a few of the new tools, the Outline or Images tools for instance.**

7 **The tools here can be very helpful in developing pages and diagnosing problems, you should use them more and more frequently as the course progresses.**

8 **Open Chrome, navigate to http://chrispederick.com/work/web-developer/ and install the extension for Chrome.**
This is nearly the same but lacks one or two features. It is however still very useful.

Activity 1.5 Safari Tools for Web Developers

The Safari Web Inspector

Safari has quite a sophisticated set of web developer tools. To enable them do the following:

1 **Safari > Preferences > Advanced and ensure the "Show Develop menu in menu bar" checkbox is checked.**

Note the new menu that has appeared.
This is quite an advanced tool so for now just know that it is there. When you feel comfortable with HTML and CSS come back and experiment with it.

Note about instructions in this book

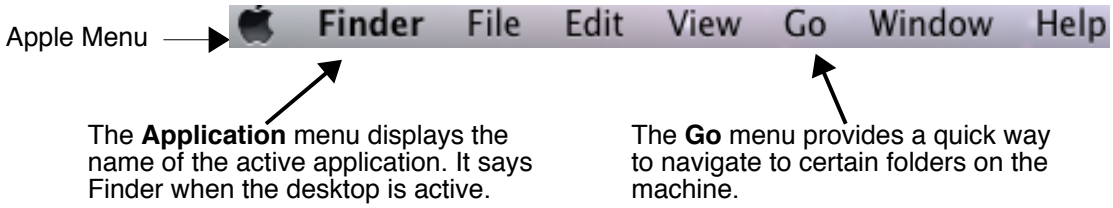
- ✔ From now on, the > symbol will be used as shorthand in instructions. For example: **File> New Folder** means that you must select the File menu, then select New Folder.
- ✔ In addition, the **>** symbol is used as shorthand for describing file locations. For example open the file:
  coursefiles>COMP112>coursefiles112>Lab01> practiceFile.txt
  means open the ***coursefiles*** directory (by double-clicking on the Desktop link), then open the ***COMP112*** directory, then open the ***coursefiles112*** directory, then the Lab01 directory then open ***practiceFile.txt***.

If you have used OS X before and feel that you are familiar with this operating system you can skip to the final checkpoint.

Activity 1.6 The Desktop Environment

The Macintosh operating system presents a different environment to a Windows machine. Even though many operations are the same or similar in the two OSs; clicking, selecting, double-clicking and right-clicking with the mouse for instance as well as the concepts of files and directories. Some of the Desktop functions such as the menu bar and the Dock and the way application windows behave are however different to their Windows counterparts and can take some getting used to.

1 Examine the desktop menus:



There are 8 menus on the upper left-hand side of the screen. Click and hold your mouse on each one to see what is in it.

- All Macintosh computers running OS X present a similar desktop. Earlier Macintosh operating systems appear and function a little differently.
- All **menus** reside in the grey strip at the top of the screen. Whenever a menu is referred to this is where you will find it.
- The **Apple** menu lets you log out, restart or shut down the computer. It also provides access to recently used items and the System Preferences.
- The **Application** menu lets you hide and show applications, quit from the active application and access preferences for the active application. Finder is the name that appears in this menu when the desktop is active. The Finder is the program that allows you to create and control files and directories, and starts up automatically when your machine is started.
- The **Go** menu allows you to quickly navigate to certain directories on the computer, including your Home directory.

Examine the Dock

The dock is a multi-part toolbar similar to the Taskbar in Microsoft Windows.

- The left side of the Dock contains shortcuts to applications. The default set are not particularly useful in COMP112, we will show you how to change them later in this lab.
- A small black dot appears beneath applications which are currently running, Finder and TextWrangler in the picture above.
- Any running application will display an icon in the Dock, whether or not it has a shortcut there.
- The right-hand side of the dock contains the Trash and any window or documents you collapse as well as shortcuts to directories.
- Web site bookmarks can also be placed here also, clicking on them will open the bookmarked page in a browser.

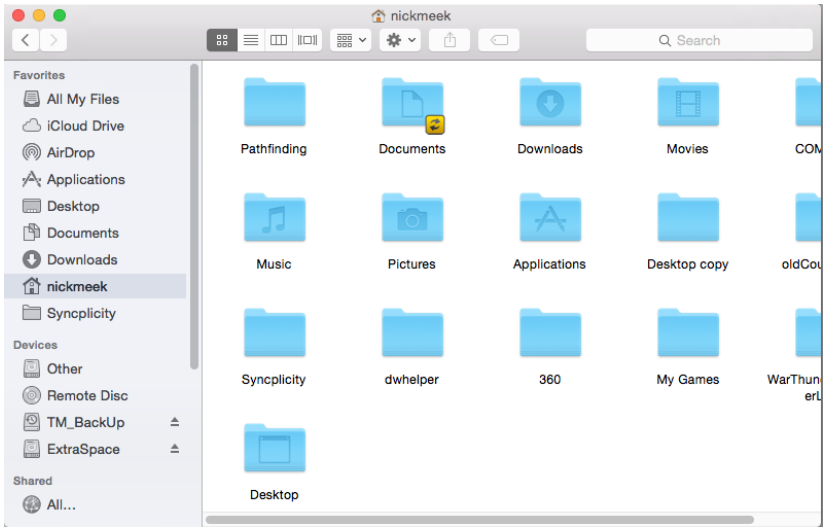
Clicking once on any item in the Dock will do one of a number of things:

- If the item is an application and the application is not already open it will open.
- If the item is an application that is already running it will become the active application. Note that although an application is active it may not display a window in which case you will need to select File>New or File>Open from the Desktop menu.
- If the application was hidden it will become visible.
- If the application had been collapsed it will be restored.

Clicking and Holding your mouse on any icon in the Dock will display a mini-menu.
The menu contents will depend on the item that you have selected. The Show in Finder option will open the containing directory.

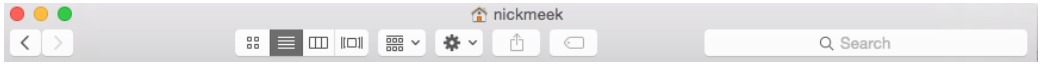
The Finder

1 **Open Finder, its shortcut is at the left end of the Dock.**
The Finder window allows you to explore you file system. You should see something similar to the window below except your will have far fewer items. On the left hand side there are shortcuts to various directories and in the right panel the contents of the selected directory.



Finder window features

The top toolbar of Finder windows looks like this:



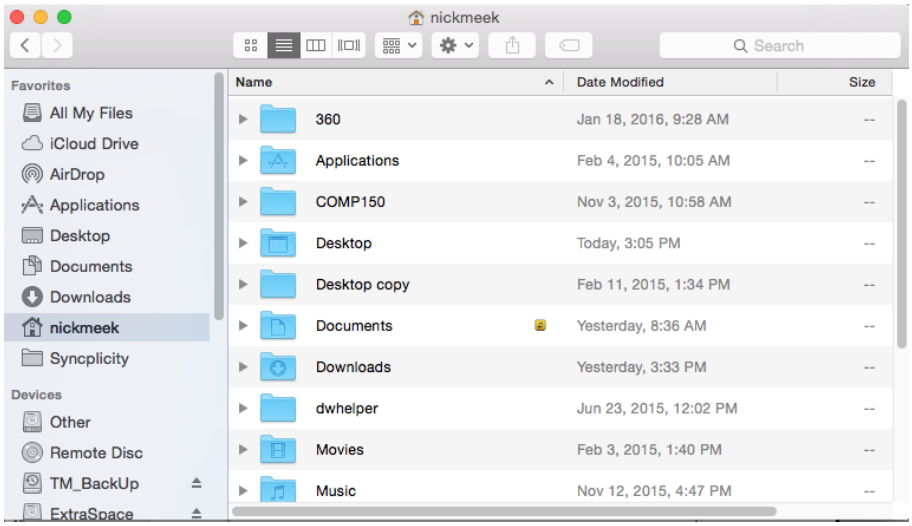
- The three coloured buttons in the top-left close, collapse or enlarge/reduce the window.
    - Click on the red button to close the window.
    - Click on the orange button to collapse the window into the Dock.
    - The green button will either enlarge or reduce the window size.
- The **arrow** buttons on the left take you back or forward one level in the file structure. These operate much like back and forward buttons in a web browser.
- The four **View** buttons change the way files and directories are displayed in a window. The first of these buttons displays the items in the window in icon form. The second button displays the contents in list form. The third button displays the contents in column view and the fourth in a carousel type of view.
- The **Quick View** button displays information about the current directory or item.
- The **Action** button is a menu of items that relate to files and directories. The same menu appears when you 'control click' (press the control key when you click the mouse) on an item.
- The field on the right allows you to search for files, directories and applications.

Other window features

Every window in the Mac OS X environment contains the following features:

- To resize a window click and drag any of the corners.
- Each window has horizontal and vertical scroll bars. These bars won't be visible if you can already see everything in the window. If there is more to be seen you can use the scroll bars to view the rest.

2 Go to your Home directory, mine is pictured below.
Notice I have selected to display the directories as a list.

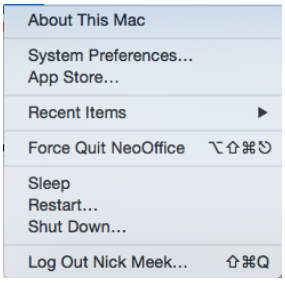


- This is the directory in which you save your work. You can create directories inside your home directory to keep your files tidy and easy to find. Organise your files and directories carefully.
- You can save your files **anywhere** within this directory.
- The **Desktop** directory contains any items you place on your desktop.
- The **Documents** directory is where application-related settings files are saved. These settings directories are created and maintained automatically by an application and you will not need to alter them.
- The **Movies**, **Music** and **Pictures** directories can be used to store relevant files. Some applications store relevant files in these directories by default. For example, if you use the lab scanners, your files will be saved to the **Pictures** directory unless you alter the default path.

The Apple menu

Previously we identified the Apple menu. Now learn about what it contains and how it can help you.

Examine the Apple Menu:



About This Mac
This displays information about your computer such as the version of the operating system it is running, how much memory it has and what processor it contains.

System Preferences
This allows you to customise aspects of your Macintosh environment.

Recent Items
This is a useful function for opening applications or documents that you were recently running. This sub menu acts as a short-cut to such items.

Force Quit...
This function allows you to terminate an application that may not be responding any more. If you do have problems with software ask a demonstrator before taking this action; using Force Quit means you will **lose any unsaved work** from that application.

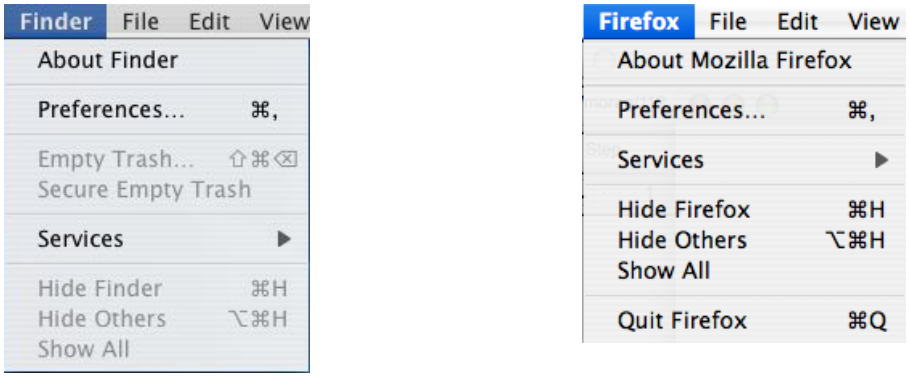
Sleep, Restart, Shutdown
Generally you will only need to log out of a Mac lab computer. We do ask that you **do not shut down** when you have finished.

Log Out...

This does exactly what it says. Note the short-cut key combination beside this option. Many menu options have a keyboard key combination related to them. In the case of log out, you hold down the shift , Command and Q buttons together.

The Application menu

The Application menu contains information relevant to the active application.



Two views of the Application Menu (left: Finder is active, right: Firefox is active).

When the desktop is active your Application menu will read Finder. When an application is active it will display the name of that application. To make an application thw active application select its icon from the Dock or click on any of its windows. Only one application may be active at a time.

The Application menu when the desktop is active (Finder):

About Finder
This produces a simple little window that tells you what version of the operating system is running.

Preferences...
This allows you to set preferences related to your desktop environment.

Empty Trash...
Files build up in the trash as you discard things. Choose this option to empty the trash. Make sure that you really do want to get rid of the contents of the trash before you choose this option. There is no way to recover items once they have been emptied from the trash.

Services
This menu provides a selection of services provided by other applications on the machine.

Hide Finder, Hide Others, Show all
These options are very useful when you have many applications and windows open at a time.

The Application menu when an application is active:

When an application is active, the name of that application appears in the Application menu. The image above shows the Application menu as it appears when the desktop is active and as it appears when the application Firefox is active. The Application menu contains similar options for all applications.

About *Application Name*
(E.g. **About Mozilla Firefox...**)
This gives you details about the application including what version is installed on your machine.

Preferences...
This allows you to set preferences specific to the application.

Services
This is the same as the Services menu when the Finder is active.

Hide Application Name, Hide Others, Show All
If you want to work with another application or on the desktop choosing the Hide option allows you move the application and all its associated windows out of the way without closing it. Hide Others allows you to hide windows related to all other applications and the desktop. Show All brings everything out of hiding.

1 **Open some windows on the desktop.**

2 **Open** Safari **by clicking on the icon in the Dock.**

3 **Use the** Hide **and** Show **options to see what happens.**

Activity 1.7 Keyboard short-cuts

OS X provides many convenient keyboard short-cuts which allow certain simple tasks to be done more efficiently. If you examine the Finder and application menus you will see keyboard short-cuts beside some commands.

Each symbol in the short-cut relates to a key on your keyboard:

⇧ = The shift key

⌘ = The command key

⌥ = The option key

^ = The control Key

⌫ = The delete key

Here are some common shortcuts:

⌘ + x = Cut,	⌘ + c = Copy	⌘ + v = Paste
⌘ + n = New	⌘ + s = Save	⌘ + ⇧ + s = Save As
⌘ + q = Quit	⌘ + p = Print	⌘ + z = Undo

Keyboard short-cuts are often quicker than the mouse actions they reproduce. Get used to looking for the short-cuts in menus and try adopting them for actions you perform regularly.

There are other keyboard functions that can speed up your work but that aren't listed in menus:

- To close all the directories on the screen at once press **Command-option-W**.
- To select multiple items at once you can shift-click on them (hold down **shift** and click on each item in turn). This is time-saving if you want to delete or move many items.
- Holding down the control key whilst you click on an item will produce a contextual menu. This is the same as right-clicking on an item.

OS X also has a feature called Exposé:

- Press F3 (or fn + F3 depending on the keyboard configuration) and all the windows you have open will be displayed on the desktop, so that you can pick the one you want.
- Press F4 to access the Dashboard.

You can alter Exposé settings from within the System Preferences.

Activity 1.8 Files and directories (folders)

Good file and directory control makes all the difference when using a computer. It is important that you can organise your work properly within your Home directory to ensure you can find the right files for further work or study. The terms 'directory' and 'folder' are synonymous but directory is preferred and I will use it when it make sense.

To create a new directory

1. **Open your** Home **directory.**
   Select **Home** from the **Go** menu.
2. **Select** New Folder **from the** File **menu.**
   Or press the shift key, the Command key and the 'N' key at the same time.
   Or you can select **New Folder** from the right-click contextual menu.
3. **While the name of the diretory is highlighted by a blue box you can type a new name, 'comp112' for instance. Press the** return **key to confirm the name.**
4. **Make a directory inside the directory you just made, name it "lab01".**
   Notice the filenames (comp112 and lab02) are all lowercase and have no spaces, this is done on purpose for a number of reasons, what do you think they are? Ask a demonstrator if you can't think of least one good reason for this.

Moving files and directories

Files and directories can be moved around by dragging them. Remember that you only have full permissions within your Home directory. You will receive error messages if you try to delete or move items outside of that.

1. **To move a file or directory simply click and drag it from one place to another.**
2. **To place a *copy* of a file in another location within your Home directory hold down the option key as you drag it.**

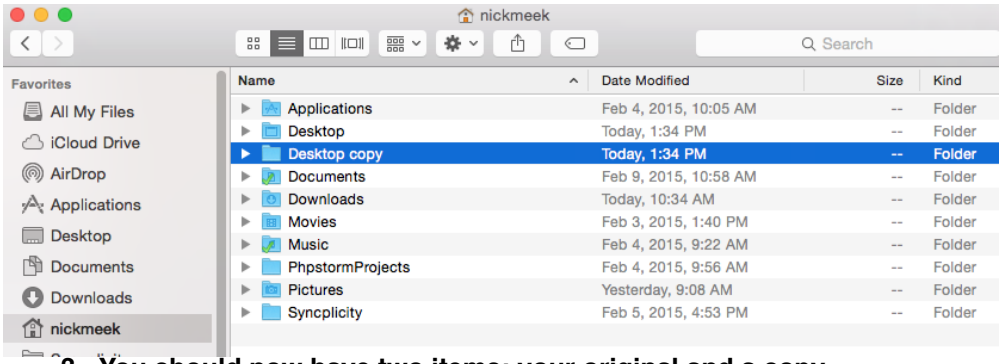
Rename a file or directory

1. **Select (click once) a file or directory.**
2. **Press the** return **key.**
3. **Wait until the name is highlighted by a blue box then press the delete key.**
4. **Now type a new name.**
5. **Press the** return **key again to confirm the new name.**

Duplicate a file

Duplicating a file or directory is very useful when you want to make a backup copy of important work.

1. **Select a file or directory.**
2. **Choose** Duplicate **from the** File **menu.**
   Or use the keyboard short-cut: press command key and the 'd' key together.
   Or you can select **Duplicate** from the right-click contextual menu.



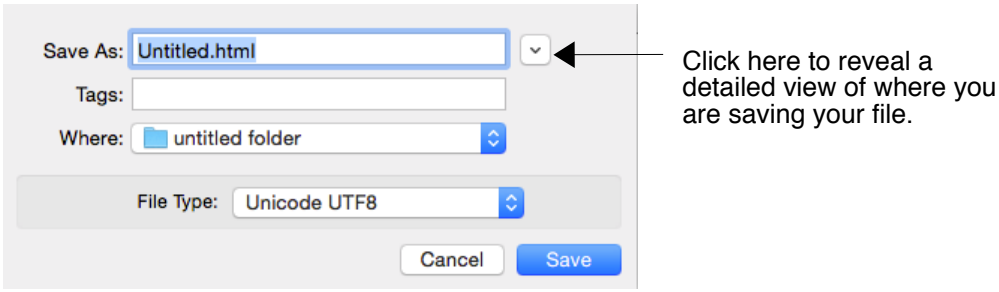
3 You should now have two items: your original and a copy.

Activity 1.9 OS X dialog boxes

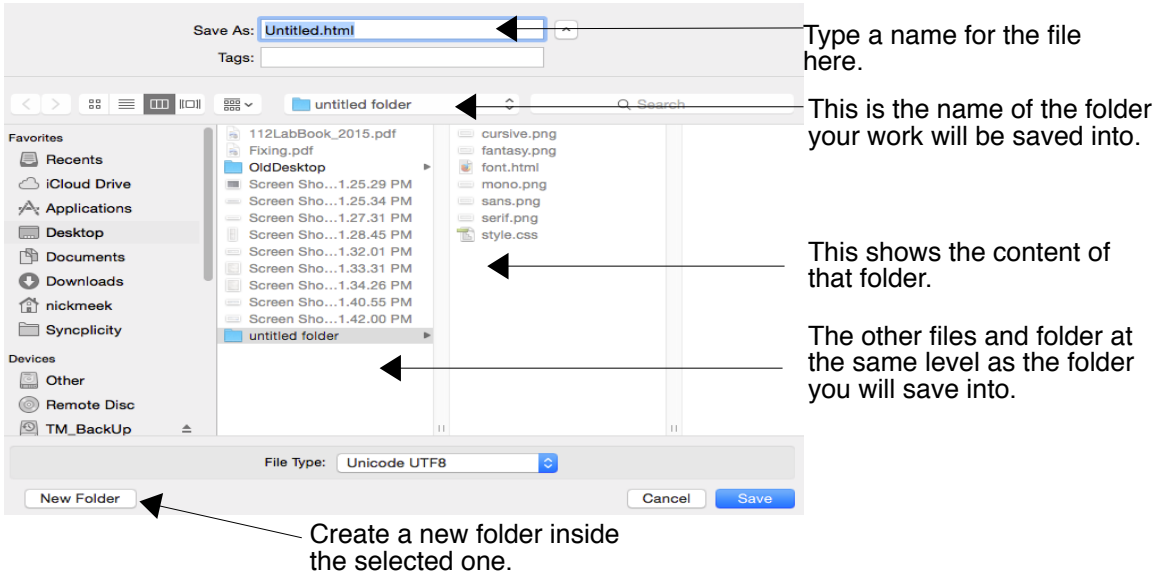
Dialog boxes that require you to negotiate the file structure are all fairly similar.

Saving

If you are saving from an application you will be presented with a dialog box similar to this:



Click on the arrow beside the Save As category to reveal a detailed view of where you will save your file.



4 Select the directory that you want to save this document in. Do this by using the pop-up menu and the list below it.

Remember you can save anywhere in your Home directory but you should try to save in appropriate and well ordered directories.

Activity 1.10 Using memory sticks

Memory sticks

To insert a memory stick, use one of the USB ports on the machine – there are some free on the rear right-hand side of the machine (beside where the keyboard is plugged in) and there is one free on the keyboard itself either at the rear edge or on one end.

To un-mount a memory stick, click the eject symbol (⏏) beside the memory stick's icon in a Finder window. Do not remove the stick from its port until the icon disappears from the Finder window.

Activity 1.11 Archiving items (zipping)

When you want to transport large files or multiple files between machines, it is often easier to create a single archive file (often known as a 'zip file' because it bears the extension .zip) of that material so that it is easier to transport.

1. **Select the directory or file you wish to archive, select the Compress "*folderName*" option from the Finder > File menu.**

If you are trying to archive a large item (or items) you may have to wait a while for the process to complete.

2. **When the process is finished a .zip file will be produced in the same place as the original.**
3. **If you want to compress a number of files or directories move them all into one directory and then compress that directory.**

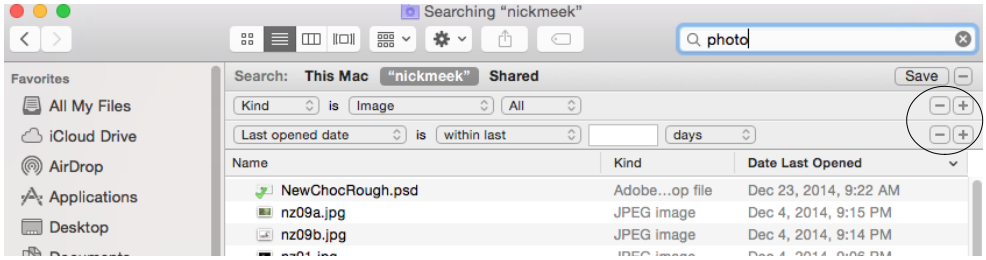
Activity 1.12 Finding items

OS X has two useful features for searching your computer for files or directories.

Finder window search

The search field at the top of Finder windows, provides a quick method of finding items by file name. As soon as you type something in it the search options bar will appear.

1. **Open a Finder window.**
2. **Type a keyword (or words) into the search field. Results of your search are returned in the window.**
3. **The search results list all the occurrences of keyword(s) found:**
4. **When many results are returned it may help to look at the Kind of thing each result is returned as.**
   If for example, you are searching for an application, look only at the results listed under Applications.
5. **Clicking on the "+" at the right of the search options bar will reveal more search options.**



These buttons allow you to add or remove search criteria.

6 **Select the areas you want to search: Servers, Computer, Home etc.**
If you want to search for an application choose "Computer", if you want to search for a file you have made choose "**Home**".

7 **Select appropriate options from the pop-up menus, so that your search is as specific as possible. You can add criteria to make your search very specific.**

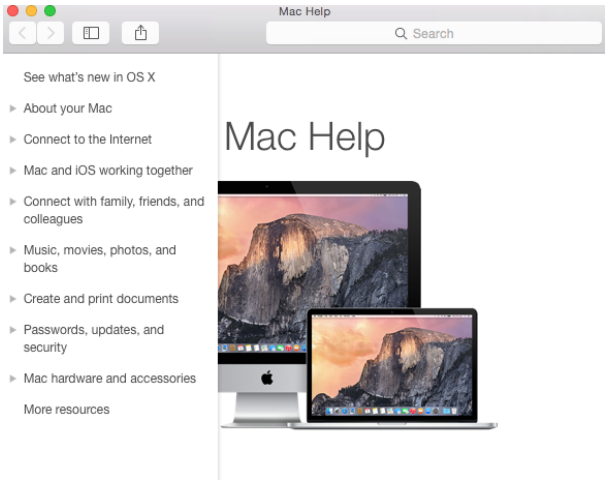
Activity 1.13 Mac Help

Mac OS X has a help system that you can use for some assistance if you need clarification on any part of the operating system.

1 **With Finder active select** Mac Help **from the** Help **menu.**

2 **Use the toolbar buttons to move through previously viewed pages or return to the home page.**

3 **You can search for keywords by typing them into the search field in the top right of the window and then pressing the return key.**



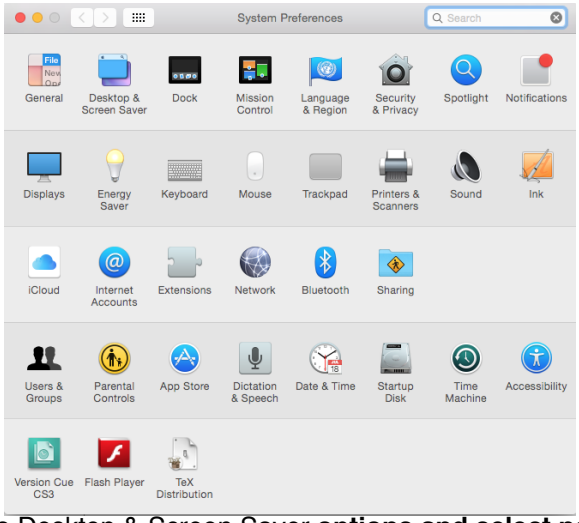
4 **When you have finished with Help, click the red close button at the top left.**

Activity 1.14 Customise your environment

You can set up the Mac desktop environment in the lab to suit you. We recommend that you alter your user-code's environment to make your work in the COMP112 lab easier.

System Preferences

You can change aspects of many of these settings.



Look at the Desktop & Screen Saver **options and select new ones if you like.**
Click the Show All button to return to the System Preferences window when you have finished in a section.

The Dock

You can drag items into and out of the Dock:
If you no longer want some of these here simply drag them off the Dock.
If you want to add an item to the Dock simply drag it from its directory into the Dock.

The default dock items aren't a particularly useful set for COMP112 work. Consider removing some of the items. A dock set-up for COMP112 may include such applications as Safari, Firefox, Opera, Taco 2, YummyFTP, QuickTime Player and Photoshop.

1 **Customise your Dock. Add at least Firefox, Safari, Taco, YummyFTP, Photoshop.**
You will be using them a lot in this course so you may as well have easy access to them.

View options

With Finder active, select View> Show View Options.
This allows you to alter the size icons are displayed, the way they are arranged and the background of a window.

Finder preferences

Select Finder> Preferences.
This allows you to change a number of features of the Finder. For example, if you would prefer a new window to open when you double-click on an icon, check the *always open folders* in a new *window* option.

Spring-loaded folders: if you drag an item to a directory and pause over the directory it will open in a new window. This is very convenient for moving items.

Dashboard

Mac OS X comes with something called the Dashboard which allows you to use and manage "mini-applications" called widgets. The Dashboard comes with some widgets pre-installed but there are many more available for download. There are many web development related widgets you may find useful in this course.

Special items

Coursework

Double-clicking on this icon will take you to a directory that contains directories for most Computer Science papers. Locate the COMP112 directory and double click on it, In this are two more directories, **coursefiles112** and **submit112**. Don't worry about submit112 for now, you will use that to submit your project later in the course. Open the coursefiles112 directory. Inside this are files provided to complete some lab activities. There is a .zip directory for each lab session. When you need the files download the .zip file to your Home directory and decompress it. You can then delete the original .zip file.

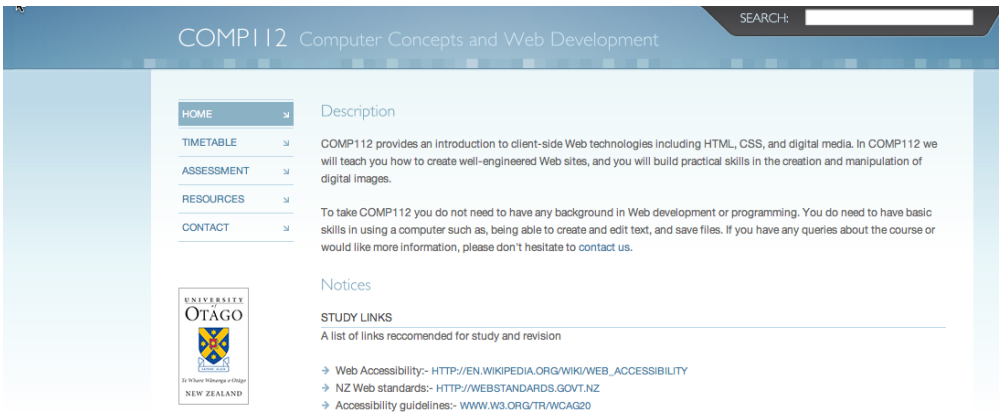
1 **There is a compressed file for this lab, download it now to the Home > comp112 > lab01 directory you created earlier.**
You might find it easier to open two Finder windows so you can drag the file from one place to another or, you could copy lab01.zip (command+c) and then paste it in the right place (command+v).

DemoCall

DemoCall sends a request for help to the appropriate DemoCall server depending on which lab you are working in. You can only receive help for COMP112 work if you are working in the COMP112 lab.

Activity 1.15 The COMP112 web page

The URL for the course web page is: **`http://www.cs.otago.ac.nz/comp112`**
You may like to add this to your Dock.



Please read the notices on the web page regularly. We may not repeat this information elsewhere.

Applications

There are a variety of applications installed on the lab machines for you to use in your COMP112 work. Although we sometimes specify an application to use, there will be instances when the choice of application is left up to you.

Applications you will find of use for COMP112 are as follows:

Application	*Kind*	*Usage*

Application	*Kind*	*Usage*
Adobe Photoshop CS3	Image manipulation	The premier digital image manipulation tool.
Firefox	Browser	A Mozilla product. Allows tab browsing.
Font Book	Fonts	Preview all the fonts installed on the lab computers.
Opera	Browser	Allows tab browsing.
Preview	Graphics	View images and PDF files.
QuickTime Player	Movies	View movie files.
Safari	Browser	Apple's web browser.
Taco HTML Edit	Editor	Basic HTML and CSS editor. Has spell checking, automatic tag organisation and line numbers.
Yummy FTP	FTP client	Lets you connect to a server via FTP to upload files.

Checkpoint

Have a demonstrator check your work now.

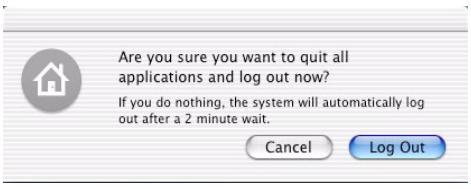
- ✔ Customised Dock?
- ✔ Changed password?
- ✔ Installed Firefox Web Developers Toolbar?
- ✔ Installed Safari Developers Toolbar?

Demonstrator:_____

Log out

You must log out at the end of each session on the computer. If you simply walk away from the machine then another person can access your files. Please **do not Shut Down** the computer – logging out is sufficient.

1 **Choose Log Out from the Apple menu. Click Log Out in the dialog box that appears.**
Please don't turn the computers off, it annoys the sys admins who can't then run automatic updates and so on.



Lab 2 HTML I – Introduction (1%)

Reference

 Lecture 2

 http://en.wikipedia.org/wiki/HTML

 http://en.wikipedia.org/wiki/Document_type_declaration

Check list

Ensure that you are confident with the following:

- ✔ Basic HTML document structure
- ✔ Using comments
- ✔ Tag syntax
- ✔ Standalone tags
- ✔ Element nesting
- ✔ Viewing the source of a web page from a browser
- ✔ Inserting images

Introduction

The World Wide Web comprises billions upon billions of web pages. Almost all of these pages use HyperText Markup Language (HTML) in some form to mark-up text and images for display in (typically) a browser.

Marking-up content requires only a text editor, something that is available in some form at no cost for every computer operating system. In this lab you will learn the basic structure of an HTML document and mark-up a page using basic HTML tags.

Activity 2.1 Lab Preparation

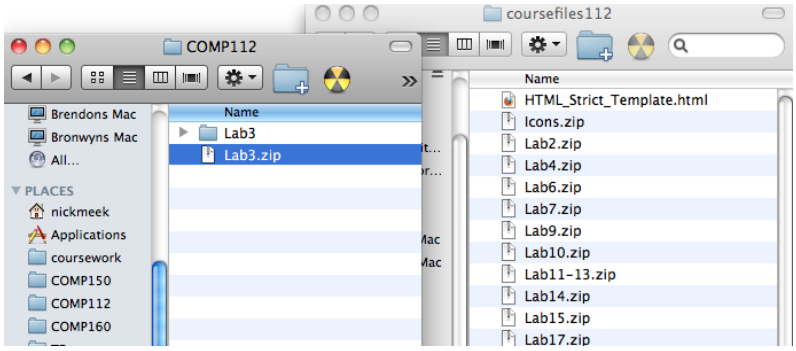
Before coming to each lab you should do the following things:

- ✔ Attend the relevant lecture.
  We spend a lot of time designing labs that support lectures and lectures that support labs. You will find labs a lot easier and more satisfying if you have been to the supporting lecture.
- ✔ Read through the lab before you come to the lab so it isn't all totally new.
  Reading the lab for the first time during your assigned lab times isn't a good use of your time; if you read it earlier you can have a coffee while you read, something you can't do in the lab.

Working this way means you should be able to finish the lab work during your assigned lab time and not have to come back later.

- ✔ Check the whiteboard at the front of the lab. Any changes or extra information about each lab will be posted there.

1 **Copy the file lab02.zip from coursefiles > coursefile112 to the comp112 directory in your Home directory by clicking and dragging the file.**



2 **Double-click on the lab02.zip file you just copied to decompress it.**

3 **Delete the lab02.zip file in your Home > COMP112 directory.**
You now have a lab02 directory in your home directory that contains some of the files needed to complete the lab. Save other files you create in this lab to this lab02 directory.

Activity 2.2 Setting Preferences in Taco

Creating HTML source files is independent of computer operating systems or applications. That is you can create HTML files on any computer you want and they will all work exactly the same. In our lab we are using Mac OSX but we could as easily be using Windows or Unix or some other operating system. There are a number of text editors on our lab machines, and you are welcome to use whichever suits you best but we'd really rather you use the one we use in the lab book.

In this lab book we have used a text editor called "Taco HTML Edit" henceforth Taco. Taco allows the use of colours to easily differentiate between different parts of the HTML document as well as having other useful features. Before you start using Taco however it will be useful to set a few preferences that will make it even easier to use.

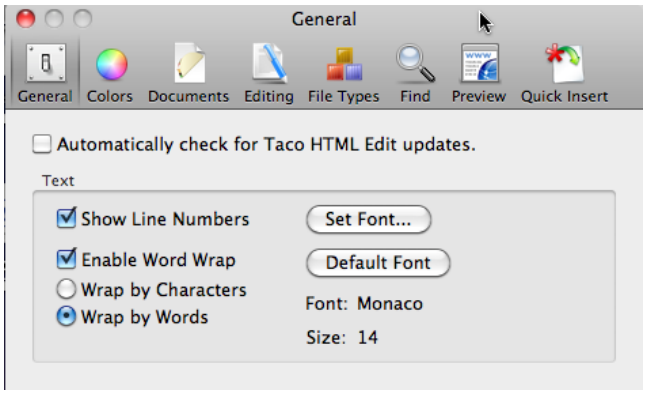
1 **Create a directory in your Home directory called TacoTemplate.**

2 **Copy the file** HTML5_Template.html **from your lab02 directory to the directory you just created.**
This file contains a minimal HTML document and will be used as the basis for all new documents. Having a template saves a bit of typing each time you start a new document so it is worth doing.

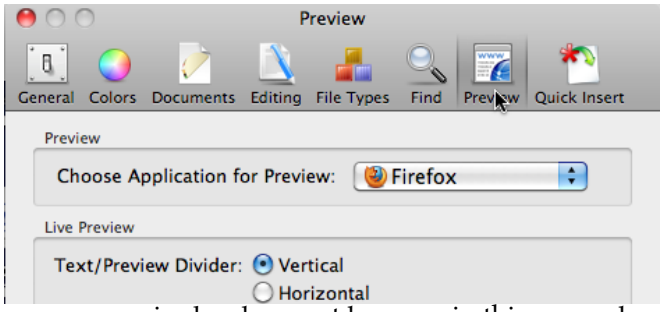
3 **Open Taco.**

4 **Select Taco HTML Edit > Preferences**

5 **From the Application preferences ensure 'Show Line Numbers' is checked.**
This will make errors easier to find.
Also make sure the Auto updates box is unchecked.



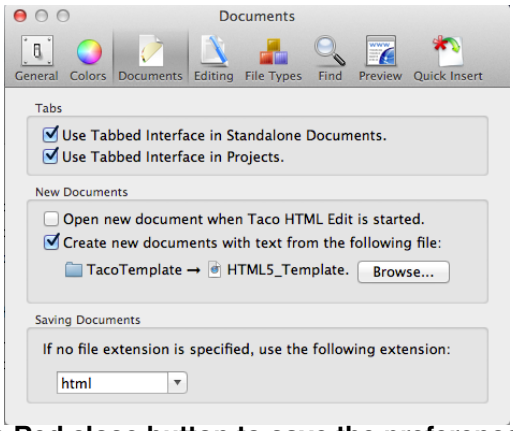
6 **Select the Preview preferences and choose Firefox to be the default preview browser.**



We use Firefox as our main development browser in this course because of the Web Developer's Toolbar.

7 **Select the Documents option.**

8 **Use the Browse button to select Home > COMP112 > TacoTemplate >** HTML5_Template.html **as the file to use to create new documents from.**



9 **Click on the Red close button to save the preferences.**

10 **Quit Taco.**

Activity 2.3 Starting HTML

1 Open Taco HTML Edit
A new document will be created using the document you specified above as a template.

```
<!DOCTYPE html>
<html>
    <head>
        <title></title>
        <meta charset="UTF-8">
    </head>
    <body>
    </body>
</html>
```

2 Note that the words are differently coloured, this helps you identify aspects of your markup.
HTML elements are used to mark-up content. Elements are indicated by the element name in < > brackets. These are known as "tags". A start tag (e.g. `<head>`) is placed at the start of a section and an end tag at the conclusion (`</head>`).
The syntax is as follows:

`<element-name>`*content*`</element-name>`

3 Notice the line:

```
<!DOCTYPE HTML>
```

This states what kind of document this is. This is an HTML5 document so the declaration is very short. If this were an earlier version then this declaration would also contain the URL at which the document type definition can be located.

Where possible your HTML markup should conform to a W3C standard. Our practice in this course will be to conform to the HTML5 standard and we will use the W3C *validator* to check that our markup does conform to that standard.

4 The following line tells the browser how the content of the file is encoded. Don't worry about it for the time being, we will explain its use in the next few labs.

```
<meta charset="UTF-8">
```

The rest of the contents of the file are a minimal HTML document albeit with no content yet.

5 Save your file as learning.html **in the** lab02 **in the COMP112 directory in your Home directory.**
(**learning.html** is your source file).
NOTE: Keeping a tidy and well ordered directory structure is an essential skill in computing and in the lab we will insist that you do so. So, all the files for this course should be in well named directories under <your home directory> > COMP112

Note: it is important to label an HTML file with the extension .html or .htm to identify it as an html document.

6 We need to add some content to our mark-up before anything will be visible in a browser. Modify learning.html as shown below. Don't try to make it bold in

your document, it is just done here to make it easier to identify which parts you need to add.

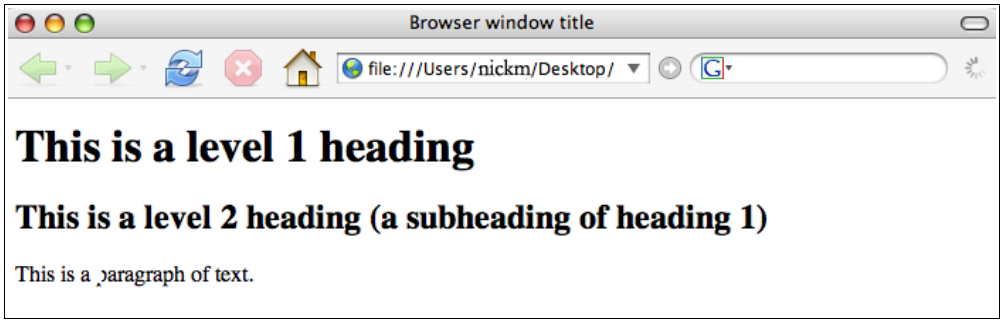
```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <!-- This is a comment. -->
      <title>Browser window title</title>
   </head>
   <body>
      <h1>This is a level 1 heading</h1>
      <h2>This is a level 2 heading (a subheading of heading 1)</h2>
      <p>This is a paragraph of text.</p>
   </body>
</html>
```

7 **Add your name and the date inside the comment tags (`<!--  -->`).**
E.g. `<!-- COMP112 Lab2 Nick Meek 01/01/2016 -->`
Comments are not displayed when the page is viewed via a browser but convey information to those who read the mark-up.

8 **Save your file.**

9 **Open Firefox then select** File> Open File... **and locate the file you just saved.**
You can use any browser you like to view your web pages: there will be a comparable command in the file menu that allows you to open a page.

10 **You should see the content of your `<h1>`, `<h2>` and `<p>` tags on your web page (see below).**



11 **If your page does not look like this there is probably an error in your HTML mark-up.**
Open the file in Taco and look for any mistakes; even the smallest thing, a missing '>' or '/' can make all the difference! When you have found and corrected the error re-save your file.

12 **Notice that the browser window title bar displays the content of the `<title>` `</title>` tags.**

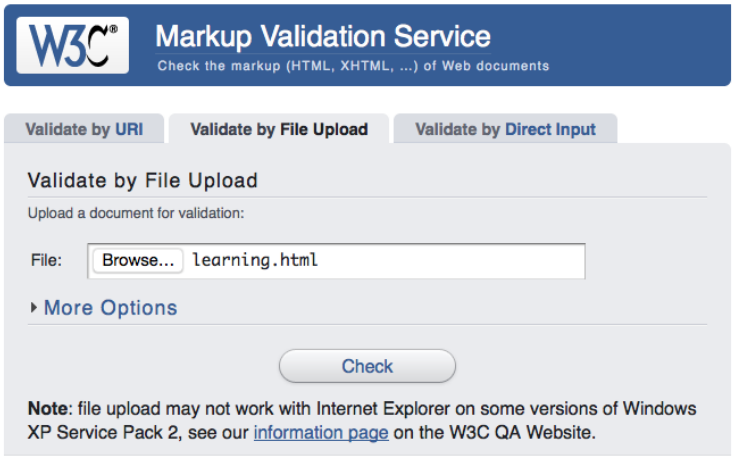
`<title>` tags only affect what the browser window title bar displays. The title tags belong in the head section of your web page. The `body` section is where tags that display the content of your web page should reside.

Activity 2.4 Page validation

Although your page displays properly in a browser that doesn't mean the markup you have written is 'valid'; browsers are very very forgiving and try hard to do something sensible with anything that is given to them. In this activity you will use the W3C's validation service to ensure that your markup is written in accordance with the HTML specification. The advantage of writing valid HTML is that it more likely to work in a predictable manner in current and *future* browsers.

1 **Open https://validator.w3.org/ in Firefox.**
This is the W3C's validation service.

2 **Select the By File Upload tab and browse to your learning.html file and click on the Check button.**



This will send your page to the W3C's validator and then display the results of the validation in the browser. If you have done everything correctly so far you will see:

Document checking completed. No errors or warnings to show.

if you look a little carefully you will see this:

This tool is an ongoing experiment in better HTML checking, and its behavior remains subject to change

This is generated because the HTML5 specification isn't final and neither is the validator.

3 **If you don't have any errors in your HTML well done! Now introduce an error, remove an ">" from somewhere. Revalidate your file and see if you can make sense of the error.**
It take some time to get used to the error messages the validator produces but they are your best tool for locating errors so the sooner you get used to them the better.

If you have made any errors you will see something like this.

Error **Saw < when expecting an attribute name. Probable cause: Missing > immediately before.**
At line 11, column 3
`ading 1)</h2↩ <p>This is a pa`

4 **If you have any errors check over your markup and fix the error.**
You should get in the habit of validating your markup frequently.

Activity 2.5 Empty (stand-alone) elements

There are lots of different elements, attributes and values that you have yet to learn, but you should be starting to understand the basic structure of an HTML document and how tags are used to mark-up content.

1 **There are a few elements that don't follow the convention of using a start and end tag.**
Among the HTML empty elements are the forced line break (
), the horizontal rule (<hr>) and image (<img>).

2 **Add a horizontal rule to your document after the paragraph of text:**

```
<!DOCTYPE html>
    ...
    <p>This is a paragraph of text.</p>
    <hr>
  </body>
</html>
```

3 **Save your work and reload it in your browser again.**

4 **Add the following mark-up below the horizontal rule in your page.**

```
<h3>Lists</h3>
<ol>
  <li>This is an ordered list - item 1</li>
  <li>Item 2</li>
</ol>
```

5 **Save your file and view your changes in a browser.**

6 **Validate your page.**

7 **Below the list add:**

```
<p>It's quite easy to <em>emphasise</em> text.</p>
<p>You can also indicate <strong>strong emphasis</strong>.</p>
```

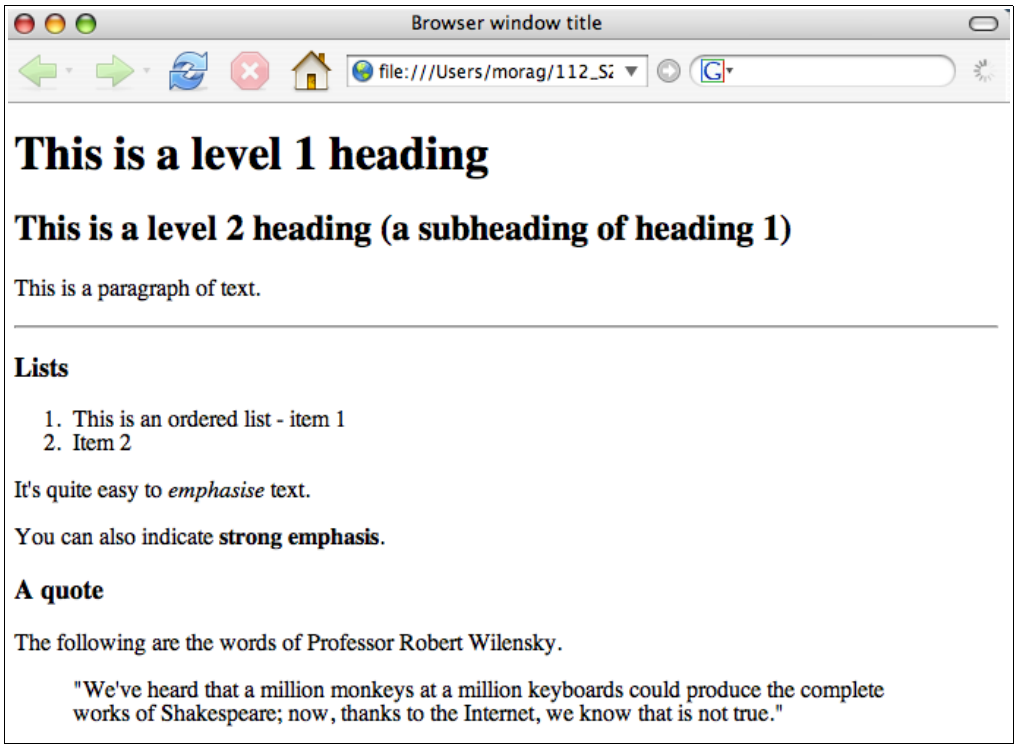
8 **Save your file and view your changes in a browser.**

9 **Now add:**

```
<h3>A quote</h3>
<p>The following are the words of Professor Robert Wilensky.</p>
<blockquote>
```

```
  <p>"We've heard that a million monkeys at a million keyboards could
produce the complete works of Shakespeare; now, thanks to the Internet,
we know that is not true."</p>

</blockquote>
```

10 View these changes in a browser. Your page should now look like this:



11 Ensure your markup still validates. Fix any errors.

Activity 2.6 Your Home Page

Now that we have covered the basic structure, let's develop a page about something a little more interesting. We will start development of the home page for *your* COMP112 website.
NOTE: You should always be careful about what information you publish about yourself. In the following exercises you should keep in mind how much you want to reveal, in particular your address, phone number and other personal information. Remember, this page will be on a publicly viewable web server; the whole world can see what ever you put on these pages. See http://www.netsafe.org.nz/wp-content/uploads/Staying-Safe-Online-NZ.pdf for some specific advice.

1 **Make a new directory in your COMP112 folder, call it "MyHomePages"**

2 **Start a new Taco document and save it as index.html to your COMP112 > MyHomePages directory.**

3 **Add a sensible phrase in the title tag, "Homepage of <your name>" maybe.**

4 **Add an h1 tag with your name.**

5 **Add an h2 tag "About Me".**

6 **Add two other h2 tags for "My Study" and "My Interests"**
See how the structure is being created. Now we just need to add content.

7 **Add a few short paragraphs (<p>) about you, (you can make it up if you wish or get some *Lorem Ipsum http://www.lipsum.com*) immediately after the first h2 element.**
Remember to keep yourself safe, don't give out too much personal information. Saying you like football and have two brothers and a dog etc., is probably fine. Giving your address, phone number and days when your flatmates are out is probably not a good idea.

```
    </head>
    <body>
        <h1>Nick Meek</h1>
        <h2>About Me</h2>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer
. Sed nisi. Nulla quis sem at nibh elementum imperdiet. Duis sagittis ipsum.
ta. Mauris massa. Vestibulum lacinia arcu eget nulla.
        </p>
        <p>Class aptent taciti sociosqu ad litora torquent per conubia nostr
libero. Sed dignissim lacinia nunc. Curabitur tortor. Pellentesque nibh. Aer
Sed convallis tristique sem. Proin ut ligula vel nunc egestas porttitor. Mor
, massa. Fusce ac turpis quis ligula lacinia aliquet.
        </p>
        <p>Mauris ipsum. Nulla metus metus, ullamcorper vel, tincidunt sed,
lass aptent taciti sociosqu ad litora torquent per conubia nostra, per incep
idunt mattis, tortor neque adipiscing diam, a cursus ipsum ante quis turpis.
c feugiat mi a tellus consequat imperdiet. Vestibulum sapien.
        </p>
        <h2>My Study</h2>
        <h2>My Interests</h2>
```

8 **Save your page and ensure your page validates**

Activity 2.7 Nested elements

Elements are containers; you put the content that you want to be affected inside a container (i.e. surrounded by tags). Quite often you will want more than one kind of element to affect the same content. When this is the case you can "nest" elements. You can put elements inside other elements so that your content is affected by both. There are some simple rules to follow when nesting elements:

- If you want to emphasise some text in a paragraph, nest your elements like this:

```
<p><em>Only some text</em> is emphasised.</p>
```

not like this:

```
<em><p>Only some text</em> is emphasised.</p>
```

- You must close elements in the reverse-order in which you opened them, i.e. the first element (tag) to open must be the last one to close.

So, this example correctly nests an element and a inside an <p> element.

```
<p><em>It is <strong>most important</strong> that the instructions are
followed exactly</em></p>
```

and this one doesn't, because the is after the .

```
<p><em>It is <strong>most important</strong> that the instructions are
followed exactly</p></em>
```

What happens if I don't nest elements this way?

Some element combinations will not function properly if you do not nest them correctly. Also your markup will not 'validate' if you don't follow the nesting rules – non-validating markup is bad, bad, bad.

Inline and Block level elements

Block-level elements are those that describe the structure of your page: headings, paragraphs, lists, divisions. Browsers typically render them on their own line. So , <h1..h6>, <ul>, <li> are examples of block-level elements.

Inline elements are those which describe a portion of that content for example: <a>, , , .

 http://www.w3schools.com/html/html_blocks.asp

In the example above with the example using and tags, elements are block-level elements, elements are inline elements.

Inline elements should always be inside a block-level element.

```
<p>Hello <em>you</em></p>
```

A block-level element should never be inside an inline element.

```
<em><p>Hello There</p></em>                    <<-- Never Like This
```

Inline elements can go inside an inline element.

```
<p><em>Hello <strong>you</strong></em></p>
```

Block-level elements may be inside another block-level elements so this is ok:

```
<body>
  <p>Hello you</p>
</body>
or
<ul>
  <li>
    <p>Hello you.</p>
    <p>Hello back.</p>
  </li>
</ul>
```

But this isn't:

```
<p>
  <h1>Hello you</h1>
</p>
```

There are specific rules which become apparent as you go along.

Activity 2.8 Skill application

1 **Under the My Study heading add a paragraph or two about what you are studying toward.**

2 **Add an h3 tag for the sub-heading "My Papers" and then an unordered list of the papers you are taking this semester.**
See the following reference for more information about lists and in particular nested lists.

 http://www.w3schools.com/html/html_lists.asp

3 **Inside that list nest another list detailing the lecture times for your papers, e.g.**

My Study

I am a first year undergraduate student studying toward a BSc majoring in Computer Science.

My Papers

- COMP112
  1. Monday 3pm
  2. Wednesday 3pm
- COMP150
  1. Wednesday 2pm
- Math170
  1. Monday 2pm
  2. Tuesday 2pm
  3. Wednesday 2pm
  4. Thursday 2pm
  5. Friday 2pm

Again you may want to think about whether you want to reveal your actual lecture timetable or not. Feel free to use the example above if you wish to preserve your privacy.

4 **Ensure your markup still validates. Fix any errors.**

Activity 2.9 How Taco displays HTML

Taco is just a text editor (like Notepad in Windows) but it does have some features that make it easier to use for writing HTML.

1 **In Taco, attributes of tags and comments are coloured for easy identification.**
You can see and alter the syntax colouring or turn it off under **Taco HTML Edit> Preferences > Coloring**.

Because Taco relies on lists of keywords to identify tags etc., sometimes a word will be coloured when it is not part of a tag. The colouring is a general indicator and is not a substitute for manually checking your markup.

2 **A key reason we use Taco is because it will auto-indent your markup for you. In the** Syntax **menu you will find** Organize Tags, t**his indents your markup 'properly'.**
Indenting mark-up makes it easier to read and easier to spot mistakes. You should always indent your mark-up. In this course we will insist that markup is correctly indented before we mark it. It is good practice and we shall *nag relentlessly* until you do as a habit.
The **Check Tag Syntax/Structure...** option in the same menu may be helpful, but it is not a substitute for validating your markup and should not be relied on.

Activity 2.10 Inserting an image

1 **Copy the image avatar.png from your lab02 directory to your MyHomePages directory.**

2 **Add the following markup to your `index.html` file just after the `<h2>About Me</h2>` tag**.

```
<figure>
  <img src="avatar.png" >
</figure>
```

`<img>` is a standalone element that will be replaced by an image (if that image can be located). We need to add a `src` *attribute* to the tag to tell the browser the **source** (location) of the image we want to display.
To validate under the doctype we are using `<img>` needs to be placed within a block-level element — here we use the figure element, (`<figure>.. </figure>`).

 http://www.w3schools.com/tags/tag_figure.asp

3 **View your page in a browser. A picture should be displayed, aligned to the left of the page.**
If you see a small graphic rather than a picture it means the browser can't find the file. If this is the case, check that you have spelt the file name correctly and that avatar.png is saved in the same directory as your source file i.e. index.html.

- **Case Sensitivity:** Ensure that your file names are not only correctly spelt, but use the correct case as well. Although some operating systems are not case sensitive with regard to file names, Unix is. You will be publishing your files on a Unix server (both in this course and often in the real world) so always treat case as important.

- Some browsers don't support images, and some users switch images off for faster loading times. The `alt` attribute provides text that can be displayed as an alternative to the image. You are required to use the `alt` attribute for all images if your page is to validate.

4 **Add the markup that is bold to your `<img>` tag:**

```
<img src="avatar.png" alt="Anonymous Avatar">
```

Your `alt` values should convey the same information as the image. **Don't use the same alt value for different images** - your users won't be able to tell them apart.

5 **To see alt descriptions you need to turn images off in your browser. Do this:**

- in Firefox by using the Images tool on the Web Developers' Toolbar you installed last lab – except this doesn't work for local images for some reason.
- in Safari by selecting Disable > All Images from the Developer bar.
- in Opera by selecting **Opera> Preferences...** . Select the **Web pages** tab and change the **Images** menu to **No Images**.

Because browsers cache images, you may have to reload your page or empty the cache for changes to show. Try holding down the <shift> key when you reload (refresh) the page.

6 **Don't forget to turn images back on when you are done!**

Add the width and height attributes to your `<img>` tag.

```
<img src="avatar.png" alt="Anonymous Avatar" width="100" height="140">
```

The values of the width and height are read as pixels. In this case the image is 100 pixels wide and 140 pixels high.

Note: the image must have height and width values set to a size big enough for the alt value to display (see the next step below). Also if you specify the size of an image as an attribute in the `img` tag the browser will reserve space for the image and the text around the image will load faster.

7 **Save your file and view it in a browser.**

8 **Ensure your markup still validates.**
Correct any errors there may be.

9 **Save you file again.**
You will add more to this file and upload it to the web server in the next lab.

TIP: To find out the size of an image right-click on the image icon in finder and choose Get Info (⌘ + I).

NOTE: that tags can contain multiple attributes, our `img` tag now has three attributes, `alt`, `width` and `height`.

Checkpoint

Have a demonstrator check your work now.
Please ensure all files are open and that all HTML is properly indented and validates using the w3 validator.

✔ learning.html
✔ index.html

Demonstrator:_____

Review activity

Demonstrators cannot help you complete review activities. However, they can help you with any other part of the lab work.

1 **Using the HTML skills covered in this lab, mark the text up in the file: lab02Review.html.**

Remember to use appropriate HTML tags to describe each element of content. This is an exercise in mark-up, not visual design. Notice how the line breaks are in the file. They are a clue as to what we think the natural structure of the document is.

2 **You should also:**

- ✔ Add your name in comment tags in the `<head>` section of the file.
- ✔ Add a title.
- ✔ Remember to indent your tags (use the **Organize Tags tool** in the **Syntax** menu).
- ✔ Use Syntax > Check Code Syntax/Structure, it will catch many 'silly' errors like forgetting to close tags.
- ✔ Ensure your markup validates.

3 **When you have completed your markup:**

- ✔ DemoCall a demonstrator so that it can be checked.

Lab Completion (1%)

Have a demonstrator check your work now.
Please ensure all files are open and that all HTML is properly indented and validates using the w3 validator.

- ✔ lab02Review.html

Demonstrator: _____
Date: _____

Commonly Used HTML5 Tags

 http://www.w3schools.com/htmL/html5_semantic_elements.asp

Block-level elements

Block level elements are usually rendered on their own line. They may contain other block-level elements and inline elements.

<h1>..</h1>...<h6>..</h6> Heading.
Used to identify key structural elements in a document. Strictly hierarchal, the first heading tag on a page must be <h1>. Other headings used to indicate equal level heading and sub-headings.
Look at this lab book as an example of heading use. The immediate heading of this passage is "<h1>..<h6>" and it is marked up us an h4. It is a sub-heading of "Block-level elements" which is an h3. "Commonly Used HTML5 Tags is an h2 as is the "Lab Completion above it. The entry after this (<p>) will be headed with an h4 as it at the same level of heading-ness as this entry and is still a sub-heading of the h3, "Block-level Elements".

** .. Paragraph.**
To indicate a paragraph of text. Paragraphs may not contain other block-level elements.

** Unordered List**
To contain the elements of an unordered list. All the elements of an unordered list must be list items (see below).

** Ordered List**
To contain the elements of an ordered list. All the elements of an ordered list must be list items (see below).

**
**
A list item is a container for the content of each list item. A list item may contain text or almost any other block-level or inline element. It can even contain another list (<ul> or <ol> or <dl>) . It cannot contain, directly, another list item.

<header></header> Header
The header tag is used to group information that is part of a section or article's "introductory content". This would probably include the document's or sections' <h1> plus maybe the document navigation etc.

<footer></footer> Footer
Similar to the header. A container for such information as author, terms of use, copyright, last updated etc.

** **

<article>

<aside>

<figure>

<nav>

<section>

<main>

Inline Elements

****

<a>

<img>

****

****

<input>

HTML I – INTRODUCTION (1%)

Lab 3 HTML II – Links (1%)

Reference

 Lecture 3

 http://www.w3schools.com/html/html5_semantic_elements.asp

Check list

Ensure that you are confident doing the following:

- ✔ Linking to an absolute URL
- ✔ Linking to a relative URL
- ✔ Using anchors to link to page fragments
- ✔ The `<nav>` tag.

Introduction

Links are fundamental to web pages. They are the 'hyper' part of hypertext. It is the links between pages that create the World Wide Web as we know it. Links (or hyperlinks as they are more correctly known) differentiate web pages from standard presentations of information. Printed material really only allows a linear or "flat" presentation, where sources or additional ideas can really only be referenced in footnotes and bibliographies. Hyperlinks in an online document allow a multi-dimensional presentation, where a reader can easily be diverted to other information that interests them without the presentation of the original document being compromised. Of course, such usage requires considered use of links and of the presentation of information. Correctly used links increase a web page's usefulness.

Activity 3.1 Lab Preparation

✔ Check the whiteboard at the front of the lab. Any changes or extra information about each lab will be written there.

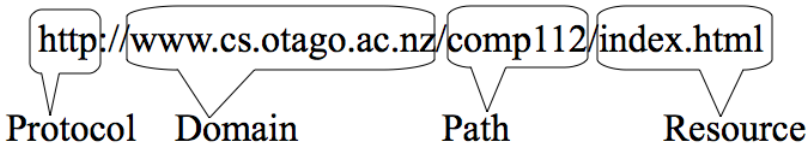
1 **Copy the file lab03.zip from coursework>coursefile112 to the COMP112 directory in your Home directory.**

2 **Double-click on the lab03.zip file you just copied to decompress it.**
It contains files you will need to complete today's lab.

3 **Delete the lab03.zip file in your Home>COMP112 directory.**
You now have a lab03 directory in your home directory that contains some files needed to complete the lab. Save other files you create in this lab to this lab03 directory.

Activity 3.2 Linking to an absolute URL

An *absolute* URL describes the protocol (e.g. http), domain, path to and name of the required page. For instance http://www.cs.otago.ac.nz/comp112/index.html is an absolute URL. When you link to a page which is not saved on the same machine as the page you are linking from you need to use an absolute URL.



1 **Start a new HTML document, save it to your** lab03 **directory as** links.html **and add the following text to the body.**

```
<p>The <a href= "http://www.cs.otago.ac.nz/comp112/resources.php">COMP112
resources</a> web page</p>
```

You will notice that the page you are linking to has a .php suffix rather than .html. We will tell you why that is later in the course.

2 **View your page in a browser and you should see this:**

The COMP112 resources web page

3 **Click on the link and your browser should open the Resources page on the COMP112 web site.**
If it doesn't then check your markup, it is important that your markup is accurate or it won't work.
This link is to an ***absolute URL***: it provides the *complete address of the file* you want to link to. This simple link is the fundamental way in which all documents which comprise the Internet are connected.

Often you will see a URL with no file name specified, http://www.otago.ac.nz for instance. In these instances the server will serve a pre-specified file, usually index.html, index.php or home.html.

1 **Add links to Google, w3School and the Otago University homepages.**

Activity 3.3 Linking to a relative URL

When you want to link between your page and other files stored on the same machine, you can provide just the *path from the current page to the linked file* instead of an absolute URL. This is called a *relative* URL, examples are given below.

1 **Add the following to your** links.html **file:**
It is important that you save this file to your lab03 directory and that it contains all the supplied lab03 files.

```
<p>This links to <a href="page1.html">page1.html</a> which is in the same directory as this page (links.html).</p>

<p>This links to <a href="./otherpages/page2.html">page2.html</a> which is in the otherpages directory.</p>
```

How to interpret **`./otherpages/page2.html`**

The dot refers to your current directory - in this activity this is your **lab03** directory.

The forward slash is similar to the > in our lab book instructions.

So, `./otherpages/page2.html` is interpreted as: "inside the current directory is another directory called `otherpages and` inside that directory is a file called `page2.html`".

Note: The pages supplied for this activity (page1.html.. page4.html) are written as HTML4.01documents not HTML5. This doesn't matter one bit but you may notice a different doctype declaration at the top of each file and some other small differences.

2 **View your page in a web browser and test it.**
Understanding relative and absolute urls is an important skill to learn. Make sure you understand them before you leave the lab today.

3 **Make links to page3, page4 and page5.html.**

Activity 3.4 Linking to fragments of a page

Sometimes you want to link to a specific section of a page. Marking a portion of a page (a fragment) with an *id* enables you then link straight to that section.
Create a link from links.html to the file ada.html (it is in the lab03 directory).

4 **Test your link and examine the page it takes you to.**

5 ada.html **is a long page, with many sections of information on it. Rather than just linking to the page generally, it may be more useful to link directly to a specific fragment on a web page. We can do this by inserting anchors into the markup.**

6 **Open** ada.html **in a text editor (e.g Taco) and scroll down until you find the bibliography section. Add the following markup:**

```
<h3><a id="bibliography">Bibliography</a></h3>
```

7 **Now alter your link to** ada.html **in your** links.html **file to read:**

```
<p><a href="ada.html#bibliography">The Bibliography section of ada.html</a></p>
```

8 **Make sure you have saved both pages, then view your** links.html **page in a browser and test your altered link. You should now be taken directly to the fragment of the page identified by the anchor name "bibliography".**

Fragments don't have to be on other pages: you can just as easily link to a fragment on your current page.

9 **In the** ada.html **file add a link at the top of the page (just above the 'Introduction' heading will do) like this:**

```
<p><a href="#bibliography">Go straight to the bibliography</a></p>
```

10 **Test your link in a browser.**

Activity 3.5 Linking to files

It is often useful to allow your users to download files by linking to them on your page.

1 **In your links.html file add the following:**

```
<p><a href="http://www.cs.otago.ac.nz/comp112/download.zip">
Download a little zip file (4KB) via http.</a></p>
```

2 **View links.html in a browser and test this new link.**
The file should download (to your desktop) very quickly as it is only about 4KB in size. The file is downloaded because it isn't of a file type the browser can display.

3 **When providing files for download you should state how big and what kind of file they are, so that your viewers can make an informed decision about downloading.**

Activity 3.6 Linking from an image

Images can act as links just like text can.

1 **Use an <img > tag to include** gorilla.jpg **in links.html. The image is in your** lab03 **directory.**
Remember to 'wrap' the `<img>` tag in a `<figure>` tag. Also don't forget the width, height and alt text attributes.

2 **Place link markup around the image to make it a link to a higher quality version on the image.**
For example:

```
<figure><a  href="gorillaHQ.jpg"><img  src="gorilla.jpg"  alt="A  gorilla
family." width="240" height="160" ></a></figure>
```

3 **View your page in a browser and test your link.**

4 **It isn't always obvious that an image is a link, so it is good form to inform readers of the fact.**

```
<figure>
  <a href="gorillaHQ.jpg"><img src="gorilla.jpg" alt="A gorilla family."
width="240" height="160" ></a>
```

```
  <figcaption>Click image to see a higher quality version.</figcaption>
</figure>
```

Activity 3.7 What to link from

Choosing the correct *anchor text* (the text you can click) is very, *very* important for a whole variety of reasons that will be explained in lectures but here is a taster.

1. Descriptive link text ranks better in search engines.
2. Descriptive anchor text gets clicked. For instance, in a website you might find either of the following:
   - See latest tech news here. Or
   - Latest: Worst worm since Slammer Spreading

   The second example is far preferred.

1 With a pencil underline the sections of text (below) that should be links and indicate where they should link to if the passage was part of an article on Wikipedia.

"An 'unhealthy correlation' exists between the construction of skyscrapers and financial crashes, according to a new report from Barclays Capital.

The construction of the Empire State building in New York in 1930, along with towers in Kuala Lumpur in 1997 and Dubai in 2010 have all been followed by economic crises, the report noted."

http://www.stuff.co.nz/world/6271275/Skyscrapers-point-to-financial-ruin

Activity 3.8 Form your Group

It is hard learning in isolation and so we want to encourage you to interact with your fellow students, and we want you to think of the people that sit close to you in the lab as your group. If there is something you don't understand or you want another opinion on your design, or you want someone to check that your page looks ok to another user then other members of your group are your first port of call.

1 Open MyHomePages > index.html in Taco. Toward the bottom of the document add another <h2>, "My CSnet".

2 Under that add a link to the csnet index page (http://csnet.otago.ac.nz/)
This page has links to all the csnet account home directories.
You will need to use absolute links for this link and the next one. Why is this do you think?

3 Add an h3 level heading "My Group" under the link you just made.

4 Under that add a *list of links* to the csnet pages of the three people who are sitting closest to you.
Think carefully about what the linking (clickable) text should be.
Lists of links are a common markup feature in web pages. They are used in situations like this and also for marking up site navigation.

5 Test that your links work. They will only show the default pages at the moment but in the next lab you will upload your own pages.

Activity 3.9 The <nav> tag

 http://www.w3schools.com/tags/tag_nav.asp

HTML5 introduced a new <nav> tag for blocks of navigation. Like many of the new tags exactly where and how it is supposed to be used is not entirely clear... yet. We expect its usage to become clearer over time. However, your My Group links seem like a suitable candidate. Other places where it should be used would be your main site navigation.

1 **Surround your list of links from the previous activity by <nav> tags.**

```
<nav>
      <ul>
            <li>A link</li>
            <li>Another link</li>
            …
      </ul>
</nav>
```

Checkpoint

Have a demonstrator check your work now.
Please ensure all files are open and that all HTML is properly indented and validates using the w3 validator.

✔ links.html

✔ index.html

Demonstrator:_____

Activity 3.10 Review activity

In this activity you will re-arrange the content of your MyHomePages homepage into several different pages and organise navigation between them.

1 **Open MyHomePages > index.html in Taco**
There is quite a lot of different information all on one page, even if it is all Lorem ipsum. It would be better to split this information across a number of pages.

2 **Make three new html documents and save them to your MyHomePages directory as: myStudy.html, myInterests.html and myGroup.html.**

3 **Add an <h1> to each document to act as the top level heading, "My Study", "My Interests" etc. for instance.**

4 **Add suitable content to the <title> tag.**

5 **Move the relevant sections of markup from index.html to the appropriate files.**

6 **Save everything!**

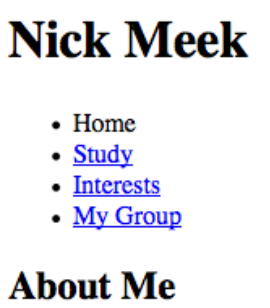
7 **Ensure each html document validates and fix errors as necessary.**

Now you have the content split up over a number of pages but no way to navigate between them.

8 **Just under the <h1> tag in index.html add a list of links to each of the other pages (don't forget the <nav> tag as well).**
When it is complete and tested, copy and paste it to each of the other three pages.

9 **A page shouldn't usually link to itself so remove the href property from the link to the current page. For instance on index.html the link should look like:**

```
…
<li><a>Home</a></li>
<li><a href="myStudy.html">Study</a></li>
…
```



10 Check that you can navigate between all the pages and that all the pages validate.

Lab Completion (1%)

Have a demonstrator check your work now.
Please ensure all files are open and that all HTML is properly indented and validates using the w3 validator.

Demonstrator: _____
Date: _____

HTML II – LINKS (1%)

Lab 4 Publishing Your Web Page (1%)

Reference

 Lecture 4

 http://www.boxuk.com/blog/the-ultimate-website-launch-checklist/

Check list

Ensure that you are confident doing the following:

- ✔ Using an FTP client
- ✔ Replicating a file hierarchy
- ✔ Working through a checklist in preparation for publication.

Introduction

Informative, well designed web pages aren't much use if they aren't accessible on the web by others. Computer Science has its own web server for student use: www.csnet.otago.ac.nz. You have your own personal space on this machine onto which you can upload your own web pages, from there they are viewable worldwide. In this lab you will learn how to upload files to the web server using YummyFTP.

Activity 4.1 Get the lab files

1 **Copy the lab files to your Home > COMP112 directory and unzip them.**

2 **Delete the .zip file.**

Activity 4.2 ftp clients

The usual method of transferring files between computers (such as Web pages from your machine to a Web server) is via File Transfer Protocol (FTP). There are many applications to help you do this (if you don't want to use the command line tools); in this lab we use YummyFTP.

1 **Open** Yummy FTP **(it is in the Applications directory).**

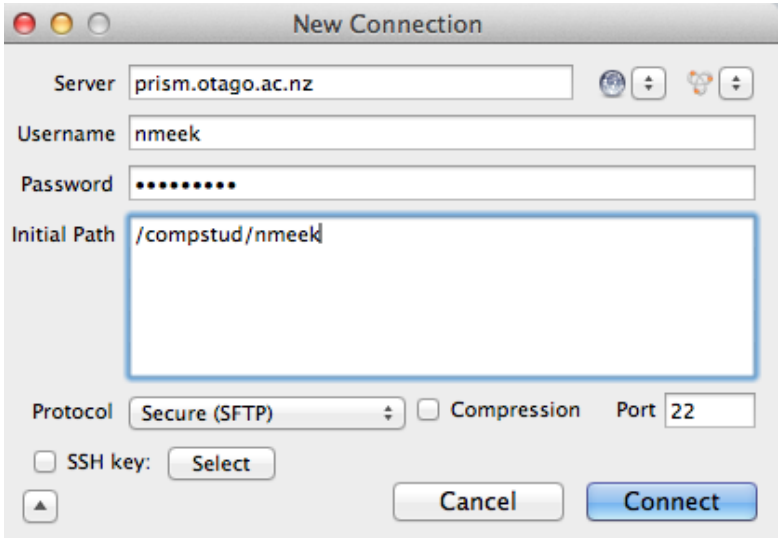
2 **You should be greeted by a** New Connection **window (if not select** New Connection **from the** File **menu):**

3 **Click the arrow button at the bottom left to reveal the full connection details.**

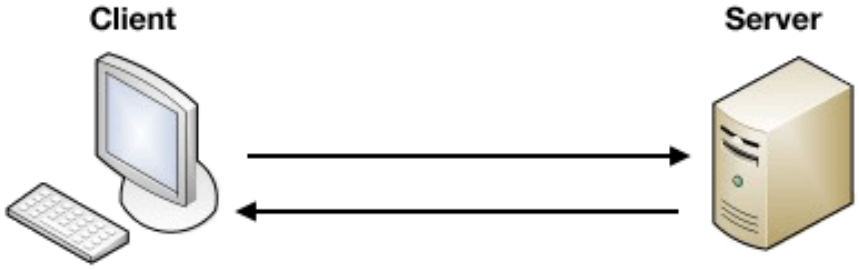
4 **Fill in the server name (prism.otago.ac.nz), your username, and password.**
These are the same as you used to log-on to the Mac in the lab.

5 **Check that the protocol reads Secure (SFTP):**

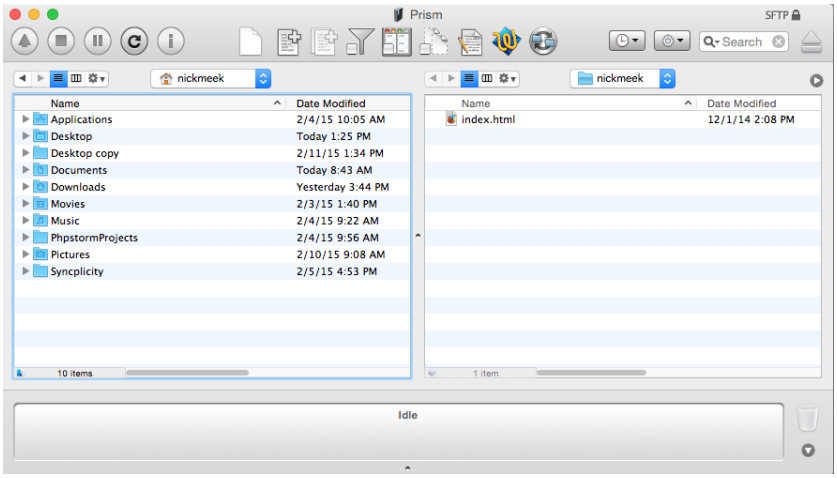
6 **Enter the correct Initial Path, I.e replace 'nmeek' with *your* CS username.**



7 **Click the** Connect **button.**
A connection with the server should be established, you are the client.



8 **The main Yummy FTP window will display the contents of two directories. Your local Home directory is on the left and the contents of your directory on the web server (Prism) are on the right:**



Notice that there is already a file on the prism side, that is the page at `www.csnet.otago.ac.nz/<username>/index.html`

1 **Open a browser and navigate to www.csnet.otago.ac.nz/<username>/index.html**
You should see a default home page, the very same one that you can see in the YummyFTP window. If we want to change the contents of this file we can do one of two things.
    1. Edit a copy of the file on your local (client) machine and then upload it to the server, overwriting the original version.
    2. Edit the copy that is on the server directly.

Let's try Option 1 first.

2 **In the left-hand pane of YummyFTP navigate to your Home > COMP112 > lab04 directory**

3 **Download a copy of index.html by dragging it from the server pane (on the right) to your lab04 directory.**
The file is copied from the server to your local machine.

4 **Open Taco and then File > Open lab04/index.html.**

5 **Make some trivial change to the file and save it..**
Just so that you can see that it is a different page.

6 **Upload the edited local version of index.html to the server by dragging it from the left pane (client) to the right pane (server).**

7 **Go back to your browser and refresh `www.csnet.otago.ac.nz/<username>/index.html` you might need to hold the <shift> key down to force a page refresh. If you don't see the new changed page in the browser something has gone wrong.**

8 **The Web Developers Toolbar in Firefox has an option to force a page reload every time. Look under the Disable > Disable Cache options.**

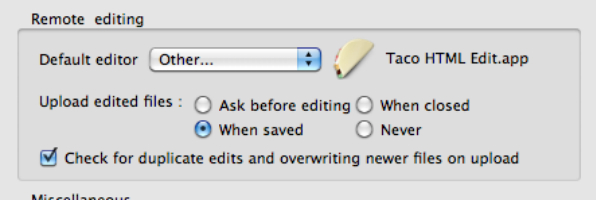
On the whole the above method of development is preferred. It has many advantages not least of which is that you always have two copies of the markup (one locally and the other on the server) so if you mess it up you can go back to an earlier (working) version. However, in some circumstances, for instance just experimenting with various techniques or looks, this can be a little laborious. In these cases you may choose to edit files directly on the server.

Activity 4.3 Editing on the server

In the previous activity you created/edited a page on your local machine and then uploaded it to the server. The advantage of doing this is that you always have a copy of what is on the server on

your local machine. You can edit files directly on the server, however, *you are working on the live version* and if things go badly wrong, and sooner or later they will, you don't have a way to step back to the previous working version.

1 **Open YummyFTP> Preferences> Server Options and select Taco as the default editor.**



2 **Use YummyFTP to connect to prism (if you are not still connected.**
You should see the index.html file as before in the right-hand window.

3 **Select index.html and then click on Edit.**
The file will be opened for editing in the usual fashion. Any changes you save now will be reflected immediately by the web server (after a page reload). Be careful when editing files like this, disasters are entirely probable!

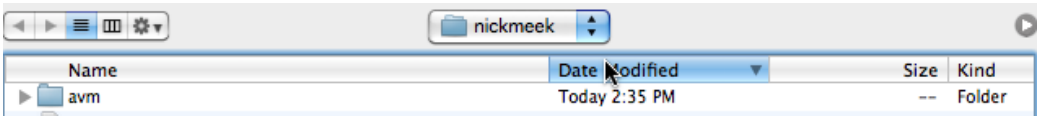
4 **Make some changes, save the file.**
Saving the file will overwrite the copy of index.html on the server.

5 **Refresh / reload the page in a browser, ensure you can see the changes you made.**

Activity 4.4 Deploy a web site

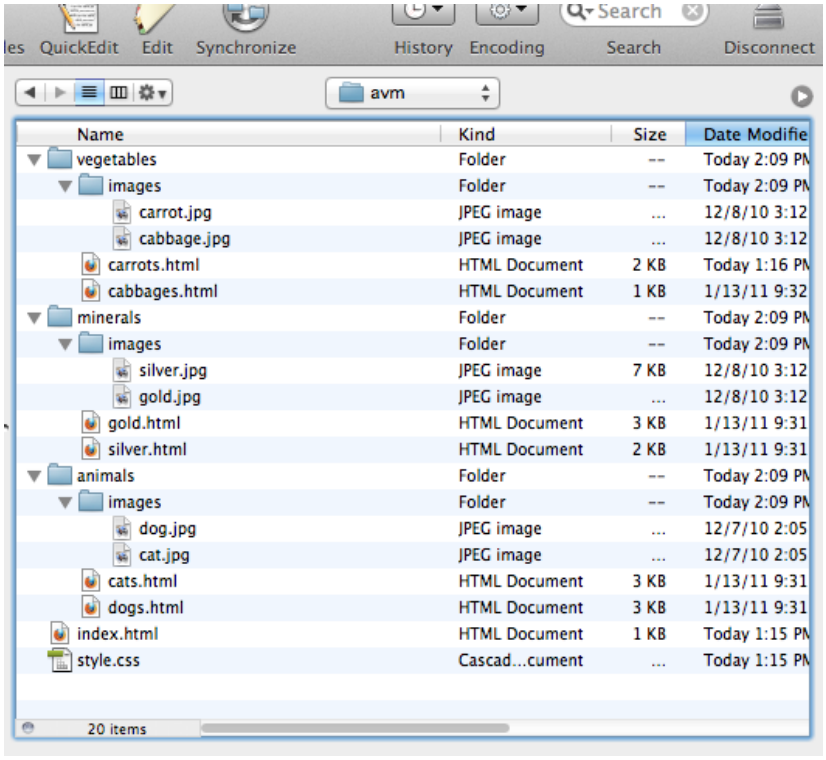
In the Lab04 directory you will find a directory called avmFiles (for Animal, Vegetable, Mineral). It contains all the files you need to recreate the following site.

1 **Using YummyFTP make a directory called "avm" at the root of your cnet account.**



2 **Inside that and using the supplied files, create the site shown below.**
You don't need to modify any of the html to do this.

Note: The pages supplied for this activity are written as HTML4.01 documents not HTML5. This doesn't matter one bit but you may notice a different doctype declaration at the top of each file and some other small differences.



3 **You should be able to reach the index page at the following url: http://csnet.otago.ac.nz/<*username*>/avm/**

4 **Check that all the links work properly and that all the images show.**

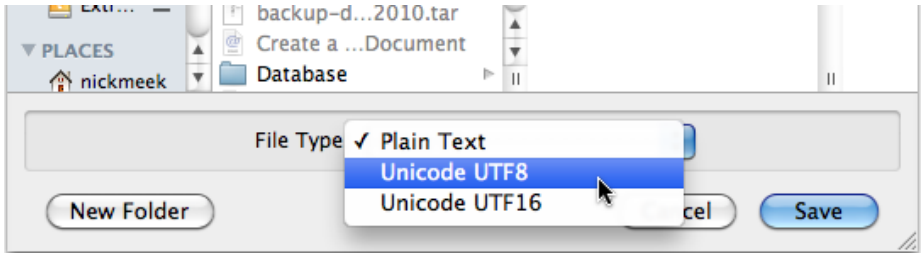
Activity 4.5 Character encoding

Let's return to the mystery tag in the header.

```
<meta charset="UTF-8">
```

This tag specifies the character encoding scheme that the file is saved in. Knowing this helps the browser interpret your document correctly. UTF-8 is a dominant character encoding scheme that is used by approximately half the world's web pages.

By default Taco saves your documents with ASCII encoded text. You should ensure that you choose UTF8 from the Save As.. menu in Taco when you save new documents for the first time.




The use of `<meta>` tags is discussed on http://www.w3schools.com/html/html_charset.asp

There are many other `<meta>` tags which, although not required for validation, provide useful information about your page.

1. **Open index.html, myStudy.html, myInterests.html and myGroup.html in Taco.**
2. **Save each file and choose Unicode UTF8 as the character encoding.**
3. **Upload the files to prism, this will overwrite the current index.html. That is fine.**
   You will need to upload the image as well.
4. **Check that your files uploaded correctly by navigating to http://csnet.otago.ac.nz/*<username>***
   Remember to check that all the links work and that you can see the image.

Activity 4.6 Style.css

Notice that one of the files you transferred in the avm exercise has a different suffix (.css) to the others. CSS stands for Cascading Style Sheet and these files provide information on how the browser should style your web page. Quite a lot of your time learning to write web pages will be devoted to working with CSS. In this activity we will take a very brief look at some simple CSS to prepare you for the next lab.

1. **Using Taco open your local version of style.css.**
   CSS files are just plain text (like html files) and so can be created and edited with any text editor. We will use Taco.
2. **Have a look through the style sheet.**
   Much of it may not mean anything to you at this point, don't worry. In a few weeks you will be reading and writing this stuff like a pro.
3. **Change the color in the first rule from #FAFFFA to #FF0000 (red), save the file, upload it to the server and then view the avm site in a browser. All the text should now be red.**


See http://www.w3schools.com/cssref/css_colors.asp for more about colour codes

4. **Change some of the other colour codes and see what happens.**
   You might like to close the local version of style.css and edit the version directly on the server.

Activity 4.7 Testing Checklist

A web site should not go live until it is finished. Consider the following as the minimum standard required for any site you create in this course and elsewhere. We will use this checklist when evaluating any of your work, including your project submission.

- ✔ Check that the source HTML and CSS validate to the required standards.
- ✔ All your links and anchors should work. Test them all. There are link checking programs that can help with this, they are especially useful on large sites.
- ✔ Ensure all images appear as they should. You may have mis-typed a height or width value or miscalculated the proportions in the first place.

- ✔ The pages display properly in a range of browser window sizes. Resize the browser window and make sure all elements on your page behave as you expect when viewed in a range of different window sizes. Pages should currently be designed to work well with screens at 1024 x 768 pixel resolution. Again the Developer toolbars have tools to simplify this.
- ✔ Does your page still make sense if you turn off images and CSS? This is how a screen reader or small screen/mobile device user might see your page. Use either of the Developers' Toolbars to check.
- ✔ Colour

Ensure that your use of colours is clear and easy to read. Your page shouldn't be hard on the eyes (red text on blue is a combination worth avoiding in most circumstances). A high contrast between background and text colour is desirable for legibility. Black on white (or vice versa) is always a safe stand-by but other colour combinations are good too. Don't forget about your link colours either, using the default link colours is okay, so long as they work with your colour-scheme. Don't forget to look at your links in all their possible states (unvisited, visited and active).

- ✔ Plagiarism and Copyright

Ensure your content is your own or suitably referenced. Under no circumstances should you copy the work of others; this includes, text, images, sound or video or even CSS and html markup. If you have any questions about plagiarism, copyright or referencing please ask. Remember: if it is not yours it is someone else's and you need permission to use it.

- ✔ Spelling/grammar

Pay careful attention to both spelling and grammar.

- ✔ Browser/platform testing

Just because your page looks good in Firefox on Mac OS 10.x , doesn't mean it will look good in anything else. You should test your web pages in as many browsers as you can; test it using different screen resolutions; test it on different platforms. It is unlikely that you will produce a page that looks exactly the same in all browsers and on all platforms, but it is important that your page be usable on most.
There are also some services which allow you to see what a page would look like on a variety of machines. www.browsershots.org is one such service, there are others.

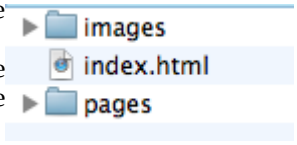
In COMP112 you are expected to produce work of a University standard (which includes accurate spelling and grammar and appropriate content and language).

Taco provides facilities to check your spelling (**Edit> Spelling...** and **Edit> Check Spelling**).

There are also spell checkers available online (e.g. `http://www.spellcheck.net/`).

Activity 4.8 Review Activity

1. **Delete the avm directory from the server.**
   It has served its purpose and is just making the place look untidy.
2. **Organise the pages in your MyHomePages directory using the following scheme:**
   You will need to modify the paths for the links and image location to reflect this new organisation.
   Keeping files organised in this fashion makes maintenance much easier and even for small sites, such as this, make development much easier.



3. **Check that everything works as it should using the checklist above.**

4 **Hopefully one of your group members will be on the same exercise, check through their sites and see if you find and errors. Be a pal and let them know if you spot any errors.**

Lab Completion (1%)

Have a demonstrator check your work now.

- ✔ AVM completed
- ✔ Own home pages created and uploaded to Prism.
- ✔ Peer checking
- ✔ Everything validates?

Demonstrator: _____
Date: _____

PUBLISHING YOUR WEB PAGE (1%)

Lab 5 CSS I (1%)

Reference

 Lecture 5

 http://www.w3schools.com/css/css_howto.asp

Check list

Ensure that you are confident with the following:

- ✔ Linking to CSS files
- ✔ Colour codes
- ✔ CSS structure
- ✔ Inheritance
- ✔ Style hierarchy (cascade)
- ✔ Comment syntax
- ✔ CSS validation
- ✔ Box properties
- ✔ Background properties

Introduction

Cascading **S**tyle **S**heets revolutionised the way web pages were formatted. Prior to the introduction of CSS, styling information was embedded in the HTML as tag attributes vastly increasing the size of HTML files. Using CSS not only slims down the size and complexity of HTML files but also enables the appearance of a web site to be quickly and universally altered and ultimately makes maintaining a site much easier. The use of CSS for styling information is required under the HTML5 doctype.

Activity 5.1 Linking Documents

Up until now all links have been visible to and clickable by the browser user. In the next activity we will add a link in the <head> section of the page, so it won't be a visible link. Stuff in the <head> section isn't displayed. This link is for the browser and tells it where some additional resources are located, these are usually CSS (*cascading style sheet*) files. CSS is a neat and scalable way of controlling the appearance of HTML documents. You may have noticed an example of one of these links in the previous avm example; there was a link to the .css file in the <head> section of each of the html files.

1. **Copy the** lab05 **files to your Home directory.**
2. **Copy your** lab02Review.html **file to your** lab05 **directory and rename it** cssExample.html**.**
3. **Open the file in a browser to remind yourself what it looks like.**
4. **Open cssExample.html in Taco and add the following markup to the** `<head>` **section after the** `</title>` **tag..**

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

5. **Save cssExample.html and view it in a browser.**
   The file should now look quite different.
   You have linked it to a cascading style sheet (CSS) that provides information to the browser about how the content of the page should be displayed.
6. **Open the file** basic.css **from your lab05 directory in Taco.**
   This is the CSS file you've linked to from your document.
7. **Change the code in** basic.css **as indicated in below:**

```
body{
  background-color: #d5feff;
  color: #595b52;
  font-family: Helvetica, sans-serif;
}
```

8. **Save the** .css **file and then reload** cssExample.html **in the browser. The background colour should change.**

Activity 5.2 Hexadecimal colour codes

Colours for fonts and backgrounds can be determined by name e.g. `Red` or by six-digit hexadecimal colour codes e.g. `#ff0000`, there are other ways which you will meet later.

There are 17 named colours featured in the HTML specification as listed below:

Name	*Hexadecimal Equivalent*
Black	#000000
White	#ffffff
Silver	#c0c0c0
Gray	#808080
Maroon	#800000

Name	*Hexadecimal Equivalent*
Red	#ff0000
Purple	#800080
Fuchsia	#ff00ff
Orange	#ffa500

Name	*Hexadecimal Equivalent*
Green	#008000
Lime	#00ff00
Olive	#808000
Yellow	#ffff00

Name	*Hexadecimal Equivalent*
Navy	#000080
Blue	#0000ff
Teal	#008080
Aqua	#00ffff

Hexadecimal colour codes take the form of a six digit code prefixed by a hash (e.g. #ff0000) = red, #00ff00 = green, #000000 = black, #800080 = deepish purple:

- The minimum value of any digit pair is 00
- The maximum value of any digit pair is ff (255 in base 10)
- the first pair of digits represent the red portion of the colour
- the second pair represent the green portion of the colour
- the last pair represent the blue portion of the colour

Hexadecimal numbers use base 16. Decimal numbers from 10-15 are represented by the letters A-F in the hexadecimal system.

 http://en.wikipedia.org/wiki/Hexadecimal

1 Alter basic.css **to change the text and background colours of elements in** cssExample.html**.**

Activity 5.3 Basic CSS

CSS is written and stored as plain text, just like HTML so you can use any text editor to create and edit CSS files. The application Taco provides a system of colouring your code for CSS just like it does for HTML and so may be useful but again you can use any other text editor to write CSS.

You define rules in your style sheet to control the way your linked HTML file(s) are displayed. CSS rules take the form of:

```
selector {
  property: value;
  property: value;
}
```

This is the preferred code layout style for CSS rules and we will insist that you follow it, it makes the CSS much easier to read. Unfortunately the 'Organise Tags' option in Taco doesn't work for CSS, you will have to do this manually but if you do it as you go along it isn't particularly onerous.

The selector is the element you want to affect. For example:

```
h1 {
  color: #003300;
  font-size: 160%;
}
```

In the above example 'h1' is the *selector*; 'color' and 'font-size' are *properties* and '#003300' and '160%' are legal *values* for those properties.

Activity 5.4 Internal styles

Styles can be added to an HTML document in multiple ways:

1. *Linking* to an external style sheet – as you have already seen.
2. *Importing* a stylesheet.[1]
3. *Embedded* within the <head> section of your HTML file where they apply just to that document.
4. *Inline* (within html tags much like old style (Transitional) HTML formatting),

We will look briefly at the internal use of styles 3 and 4 but ultimately it is the ability to *link* to an external style sheet that is most useful.

Inline styles

1 Open the internalStyles.html **file from your lab05**> Internal **directory and look at the markup. It is a very simple document. Notice the use of the HTML5 <header> element.**

2 Alter the `<h1>` **markup like this:**

```
<h1 style="font-size: x-large; font-family: Apple Chancery, cursive">
```

3 View your page in a browser. This *inline* use of style only relates to a specific item; if we want to apply a style to our whole document we need to do so either in the `<head>` **or by linking to an external style sheet.**

Embedded style sheet

1 Add the following inside the `<head>` **section of** internalStyles.html**:**

```
<style type="text/css">
  <!--
  p{
     font-size: small;
     font-family: helvetica, sans-serif;
  }

  header p{
     font-size: large;
```

1 "In theory, the only difference between them [@import and <link>] is that `@import` is the CSS mechanism to include a style sheet and `<link>` the HTML mechanism. However, browsers handle them differently, giving `<link>` a clear advantage in terms of performance."

 http://stackoverflow.com/questions/1022695/difference-between-import-and-link-in-css

```
    font-family: cursive;
  }
  -->
</style>
```

Note: The style rules are contained in HTML comment tags to hide them from old browsers.

2 **Save and reload your page in a browser. All your paragraph text should now be formatted as specified by the p style and the author's name should be in the style for <p> that are contained by <header> . Any tags you add to** internalStyles.html **will be styled as specified in the embedded style sheet.**

So, while the above methods (inline and embedded) of applying styles to elements are fine they are rather limited in that you have to write the CSS for every page, or every element if you use inline styles. This really isn't the point of CSS and so unless you have a particularly good reason for using internal styles, don't! Linking to external style sheets is where the true power of CSS lies.

Style sheet example

Let's look at an example of a site controlled by an external style sheet.

1 **Open** index.html **from your lab05>** cssSite **directory in a browser and examine the site (there are three pages including the front page). View the source to see how the pages are constructed.**
Notice in particular in the <head> section of each page the <link> tag linking the page to the CSS file CSSCentral.css.
Also notice the use of a new tag, <footer>.

 http://www.w3schools.com/tags/tag_footer.asp

2 **Open** CSSCentral.css **in a text editor.**
These are all the CSS rules that govern the styling for any page which links to it.

3 **Change the** h1 **text color and reload the site in your browser to see the change.**
Any changes made to the CSS file will affect the whole site because every HTML file in this site is linked to the same CSS file. This is one of the most important points of using CSS;
You only use one file to control the appearance of many web pages.
This makes the pages look consistent and makes site-wide changes easy.

4 **Alter some of the other values in the style sheet and notice the results in the browser.**

Don't feel overwhelmed by what you see in the CSS file at the moment; you will develop your CSS skills as the course progresses and reading this will be easy sooner than you would think.

Style a page with a linked CSS

In the lab05 directory is a file basic.html. Open it in a browser. In this activity you will add some styling to the page using an external style sheet.

1 **Using Taco create a new file,** style.css **in your** lab05 **directory.**
Remove all the html template stuff Taco puts there automatically. CSS doesn't have any of that, you start with an empty document. Let me say that again: CSS doesn't have any of that, *you start with an empty document*.

2 **Open your lab05>** basic.html **in Taco.**

3 **Link** basic.html **to** style.css **by modifying the link in the head section.**

4 **Now open** basic.html **in a browser. You won't see any impact from the style sheet yet as it doesn't yet contain any rules.**

Remember, rules take the form:

```
selector {
  property: value;
  property: value;
}
```

5 Type the following code in your style sheet and save the file:

```
html {
  color: black;
  background-color: #f2f2ff;
}
```

6 Reload your HTML page.
The background colour should have changed. If it haven't something has gone wrong. Check your CSS for errors.

7 Add the following CSS to your style sheet and view the change:

```
h1{
  font-size: x-large;
  color: #4d4da6;
  background-color: #f2f2ff;
  font-family: serif;
}

h2{
  font-size: large;
  color: #4d4da6;
  background-color: #f2f2ff;
  font-family: serif;
}

p{
  font-size: medium;
  color: #4d4da6;
  background-color: #f2f2ff;
  font-family: sans-serif;
}
```

Comments

Comments in CSS are marked up differently to those in HTML.

When you wish to add comments to your CSS use:

```
/* Comments */
```

1 **Add a comment to the top of** style.css **that contains your name, the date and the lab number.**

```
/* Nick Meek, Dec 2015. Style sheet for Activity 5 */
```

2 **Save the CSS and reload basic.html in a browser. The effects of the style rules should be apparent.**

The cascade

Recall that CSS stands for '*cascading* style sheet'. Did you wonder what the cascade was?

 https://developer.mozilla.org/en-US/docs/Web/CSS/Cascade

"The cascade is a fundamental feature of CSS. It is an algorithm defining how to combine properties values originating from different sources. It lies at the core of CSS as stressed by its name: Cascading Style Sheets."

As you have just seen you can implement styles in multiple ways within a single HTML document. For instance a single HTML document may have a link to an external style sheet **as well as** some embedded styling. The reasons why this might be done will not be clear to you yet but it is a legitimate thing to do in web development.

Some of the rules in the multiple style sheets could conflict. Now a problem arises for the browser, if two rules do conflict (the linked style sheet says `p{color: black;}` and the embedded style sheet says `p{color: red;}` for example) then which rule should be applied?

The cascade specifies the precedence (importance) of styling information. In the example above the `<p>` text would be red as embedded styling has precedence over linked styles and would therefore override it.

Most specific style, overrides all others	**! important (user)**
	! important (author)
	Inline styles
	Embedded styles
	@import
	Linked style sheet
	User style sheets
Least specific style, overridden by all others	**Default browser settings**

Activity 5.5 Inheritance

Style sheet elements inherit properties from parent elements higher in the HTML hierarchy. This can mean a great saving in typing, for example:

If in your CSS you specify:

```
body {
  color: red;
  background-color: white;
  text-align: justify;
}
```

And in your html you have:

```
  <body>
    <h1>Some Heading</h1>
    <p>A paragraph of content</p>
  </body>
```

Then, since `<p> and <h1>` are contained within `<body>` they inherit the body's properties and you don't need to specify them all over again for those elements.
In style.css instead of repeating the same font-family values in both the h1 and h2 styles we should do this:

```
html{
  color: black;
  background-color: #f2f2ff;
  font-family: serif;
}
h1{
  font-size: x-large;
  color: #4d4da6;
  background-color: #f2f2ff;
}

h2{
  font-size: large;
  color: #4d4da6;
  background-color: #f2f2ff;
}

p{
  font-size: medium;
  color: #4d4da6;
```

```
  background-color: #f2f2ff;
  font-family: sans-serif;
}
```

By setting the `font-family` for the `html` element we effectively set the `font-family` for all elements as every element is a child of `html` and automatically inherits some of its properties.

1 **Modify your style sheet to take advantage of inheritance.**
You can get rid of the repeated setting of the `color` and `background-color` by setting those for the html element as well.

Inheritance and colour

`Background-color` and `color` inherit from parent element to child element automatically. So if you want the `background-color` and `color` of a `<p>` element (for instance) to be the same as its parent element then you don't have to do anything. It just inherits automatically. This is true for many properties. However if you do want to change an element's `background-color` or `color` then you should always make a conscious decision about foreground and background colour combinations.
`color` and `background-color` should always be presented as a pair. If you wish one to be inherited from a parent element, simply state that explicitly. If you specify either without the other then you will get a CSS validation warning. This is not acceptable so just get into the habit of specifying both (if you specify either).

1 **Modify your style sheet so that h1 elements have a slightly different background colour but the same font colour.**

Activity 5.6 Values & Measurement Units

 http://www.w3schools.com/cssref/css_units.asp

Using CSS you can specify the width of an element in a multitude of ways.

```
h1 {width: 80%}
h1 {width: 500px}
h1 {width: 400pt}
h1 {width: 34pc}
h1 {width: 300em}
h1 {width: 350ex}
h1 {width: 10in}
h1 {width: 250mm}
h1 {width: 25cm}
```

However, when selecting a measurement unit remember to consider usability. Often a relative measurement (such as percentage or em) is more appropriate than an absolute one (such as centimetres). Absolute values are useful, but use them wisely so you don't limit the accessibility of your pages.

Font sizes

Deprecated HTML font size was limited to a range of 1 - 7 . In CSS you you have `em`, `ex`, `pt`, `px`, `pc`, `mm`, `cm` at your disposal as well as percentage (e.g. `20%`), and terms such as `x-small` and `large` or `smaller` and `larger`.

 http://www.w3schools.com/cssref/pr_font_font-size.asp

Carefully consider which units of size are best to use in each situation.

Colour

As with deprecated HTML you can use colour by name (e.g. `blue`), or hexadecimal number (e.g. `#0000ff`). You can also specify it using RGB values.

 http://www.w3schools.com/cssref/css_colors_legal.asp

Using the colour Red as an example, show an example of each of the different ways to represent colour in a style sheet:

1 _____

2 _____

3 _____

4 _____

5 _____

Activity 5.7 Skill application

1. **Open basic.html and style.css in Taco if they are not already open.**
2. **Make the following additions/changes to your style sheet (**style.css**):**

- ✔ Set the font face of all text to Geneva.
- ✔ Make the Heading 1 text italic.
- ✔ Make the Heading 2 text underlined.
- ✔ Set the font size of the Heading 2 text so that it is smaller than Heading 1 but not as small as the paragraph text.
- ✔ Change the colour of the Heading 1 text.
- ✔ Centrally align the “Little Red Riding Hood” text.
- ✔ Indent the first line of each paragraph of text.

Activity 5.8 Font usage

You can specify all the fancy fonts you like but unless these are installed on your viewers’ machines they serve no purpose. (This is changing with CSS 3) It is advisable to stick to fonts that you can reasonably expect to be installed on a standard machine.

 http://www.w3schools.com/cssref/css_websafe_fonts.asp

Specifying a selection of fonts

You can specify multiple fonts so that in the event that your viewers' system does not have the first font, the second will be called upon and so forth through all the fonts you specify.
Your final specification must always be a generic font family name. In the following example sans-serif is the generic font family.

CSS:

```
font-family: Geneva, Helvetica, sans-serif;
```

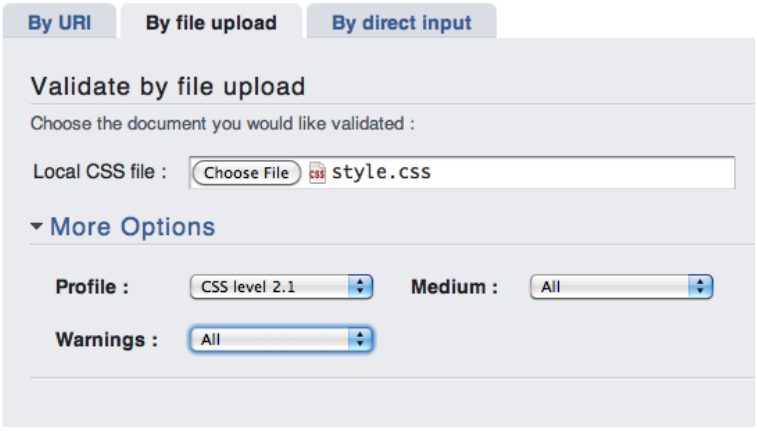
If you are viewing a page with this specification on a Macintosh then chances are your browser will display the Geneva font; if you are using a Unix breed of machine you will probably see Helvetica (as it probably won't recognise Geneva and therefore will ignore it). If you are using Windows then it may be that neither of the first two fonts are installed on your system and your browser will pick another a sans-serif font it has installed instead.

✔ **Change the font family specifications in** style.css**.**

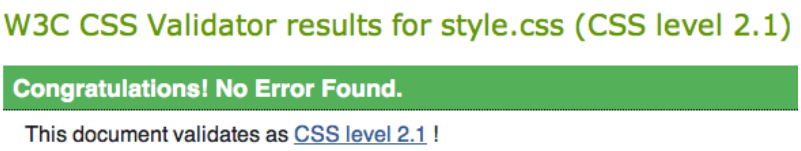
Activity 5.9 CSS Validation.

Not surprisingly there are rules about how CSS should be written in exactly the same way that there are rules for writing HTML; and just like your HTML your CSS should also be valid. Both the Firefox and Safari developer bars contain quick ways of validating your CSS. However they don't perform the most thorough check as they both disable 'warnings' on the validator. To perform the most thorough check of your CSS you have to go to the W3C's validation service or alter the Options in the Developers' Toolbar.

1. **Open a browser and navigate to** http://jigsaw.w3.org/css-validator/
2. **Select the 'By file upload' tab and choose style.css.**
3. **Click on More Options.**
4. **Ensure the Profile is CSS level 2.1.**
5. **Ensure the Warnings are set to All**



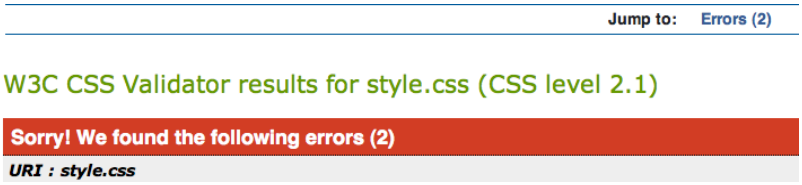
6. **Click 'Check' to have your CSS validated. Be careful however as this:**



looks very much like this:

Which is not a satisfactory result.
Note the 'Warnings (1)' in the upper right hand corner.

This of course is just plain wrong.



7 **Ensure your style sheet validates to CSS level 2.1 with no errors and no warnings by working through and eliminating any problems.**
In future all your CSS document should validate to this standard in addition to all pages validating to HTML5.

Checkpoint

Have a demonstrator check your work now.

- ✔ Colour codes
- ✔ style.css
- ✔ basic.css
- ✔ validation of both css files

Activity 5.10 Box Properties

CSS uses a "box-oriented formatting model". This is a *very* important concept in CSS so make sure you understand it clearly and thoroughly. From here on CSS is pretty much all about the 'box model' so if you don't 'get it' the rest of CSS is going to be mysterious and frustrating

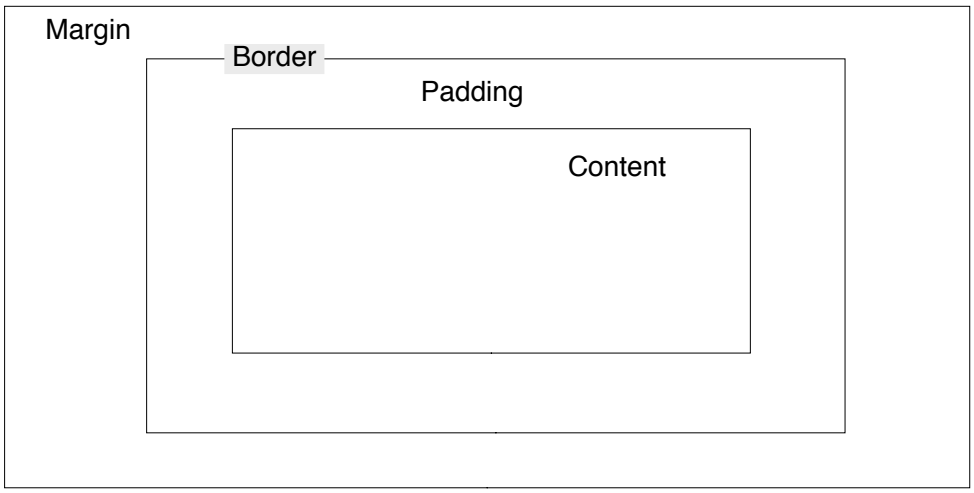
 http://www.w3schools.com/css/css_boxmodel.asp

Put plainly, a box-oriented formatting model treats **every element** it formats as if it were inside a box. Each and every element is a box; every paragraph (<p>), heading (<h1>, <h2> etc.), list (<ul> and <ol>) and list-item (<li>), link (<a>), image (<img>) *ad infinitum*, is in a box, well several layers of boxes actually. There is a box where the content is displayed, and around that there are boxes for padding, border and margin respectively.

1 **Given the following:**

```
margin: 10px;
padding: 5px;
background-color: gray;
border-top: 4px dotted black;
```

Indicate the dimensions and format on the following box diagram.



 http://www.w3.org/TR/CSS2/box.html

Firefox has a nice tool for visualising the box-model. With any web page open select Tools> Web Developer> Toggle Tools. From the window that appears select the Inspector tab on the left and then Box Model tab on the right.

Activity 5.11 Review Activity

In this activity you will add a stylesheet link to your MyHomePages homepage and add some styling rules to it.

1 **Open your local version of your MyHomePages homepage, index.html. It should be in your COMP112> MyHomePages directory.**

2 **Start a new empty document and save it as style.css to the same directory.**

3 **Add a link in the `head` section of index.html to the stylesheet you just created.**

4 **Set the background colour of the body to red, save both files and then preview index.html in a browser. The background should be red.**
It is always a good idea to do something obvious initially when linking documents just to make sure you managed to type everything correctly.
If your background isn't red find out what the problem is. Are the files in the same directory? Is the filename in the `<link>` tag correct?

- ✔ **Change the background colour to something you prefer.**
- ✔ **Change the background colour of the Heading 2 box.**
- ✔ **Make the h1 element 50% wide, change its background colour, give it a 15px dotted border in a contrasting colour and a 50px top margin.**

- ✔ **Centre the h1 text in its 'box'.**
- ✔ **Centre the h1 element in the body horizontally.**
  Hint: You don't use 'center' for this one. Think about the h1 element as a box and look back to how the `body` was centred in CSSCentral.css.
- ✔ **Add combinations of padding, margin and border properties/values to the other elements until you are confident with the boxes' properties.**

- ✔ **Upload both files to the web server and check you can still see index.html at the appropriate url.**
- ✔ **Copy the css link from index.html to the other three .html pages and upload them. You should now have a consistent style on all pages and all regulated by a single, central .css file!**
- ✔ **Ensure all files still validate.**
- ✔ **Check with members of your group to ensure that your colour combinations / font family / font size are easily readable.**

Lab Completion (1%)

Have a demonstrator check your work now.

- ✔ style.css

Demonstrator: _____
Date: _____

CSS I (1%)

Lab 6 HTML III – Tables (1%)

Reference

 Lecture 6

 http://www.w3schools.com/tags/tag_table.asp=]

Check list

Ensure that you are confident doing the following:

- ✔ Creating data tables
- ✔ Using images in tables
- ✔ Spanning rows in a table
- ✔ Spanning columns in a table
- ✔ Affecting table appearance
- ✔ Nesting tables
- ✔ Sensible use of table markup
- ✔ Making tables accessible

Introduction

The original intention of tables in HTML was simply to display data in tabular form. Web developers soon learnt to use tables for more interesting purposes and they were a mainstay of formatting a page for a long time. Thankfully those times are (more or less) over and professional web developers now use CSS to control page look and layout; tables can go back to doing what they are best at, displaying tabular data.

Activity 6.1 A basic table

1 **Copy the lab06 directory to your home drive.**

2 **Create an HTML document called** lab06.html i**n your lab06 directory.**

3 **Type the following into the body section of your document:**

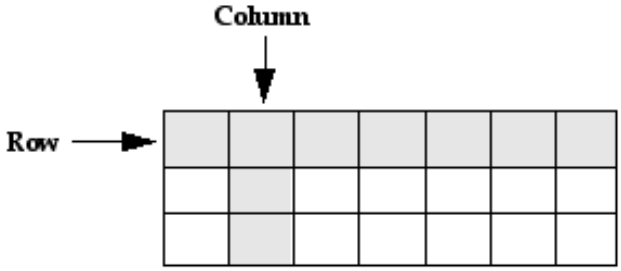
```
<table border="1">
  <tr>
    <td>row 1, cell 1</td>
    <td>row 1, cell 2</td>
  </tr>
  <tr>
    <td>row 2, cell 1</td>
    <td>row 2, cell 2</td>
  </tr>
</table>
```

Note the use of the `border` attribute in the table opening tag, this will set a 1 pixel around every cell in the table. If you validate this document it will suggest you use css for 'presentational markup'. However just for the purpose of demonstration it is ok.

4 **Save your markup and then view it in a web browser. You should see this:**

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

5 **Tables are made of rows and columns:**



6 **The** `<tr>` **tag starts a new table row, and the** `<td>` **tag creates a new table data cell, forming columns across a row.**

7 **A table of the form:**

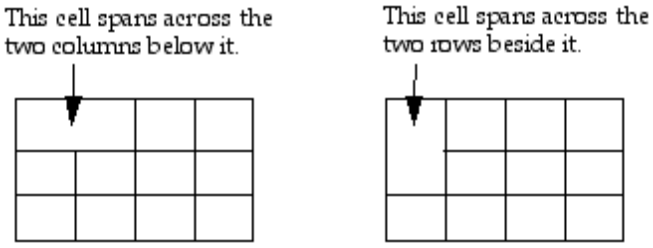
cell 1	cell 2	cell 3
cell 4	cell 5	cell 6

would be created by the following markup:

```
<table border="1">
```

```
  <tr>
    <td>cell 1</td>
    <td>cell 2</td>
    <td>cell 3</td>
  </tr>
  <tr>
    <td>cell 4</td>
    <td>cell 5</td>
    <td>cell 6</td>
  </tr>
 </table>
```

8 We can make a table even more useful by allowing some cells to span across others:



9 Two attributes are used to allow such spanning to happen: rowspan and colspan. Add to your document as shown:

```
<table border="1">
  <tr>
    <td>row 1, cell 1</td>
    <td>row 1, cell 2</td>
  </tr>
  <tr>
    <td>row 2, cell 1</td>
    <td>row 2, cell 2</td>
  </tr>
  <tr>
    <td colspan="2">row 3, cell 1</td>
  </tr>
</table>
```

10 Save and reload. Your table should now look like this:

row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2
row 3, cell 1	

This is a very basic table and probably doesn't look very impressive to you. However, tables can be a very useful tool in HTML.

The primary function of tables is to allow for the display of data in a meaningful fashion; to present related data in meaningfully labelled rows and columns. However, even today you will see many web sites that use tables to control the structure of their web pages.

Activity 6.2 A table of images

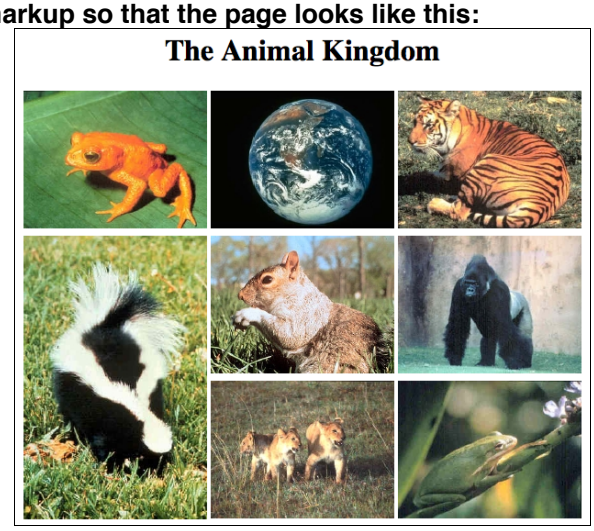
Look at the image below. Here a table is used to control the layout of some images. It may look more impressive than the table we created in the previous exercise but the markup for this table is only slightly more involved. Until recently a table would have been the method of choice for laying out images as shown. As we shall see soon we can dispense with tables as layout devices through the use of CSS.

- This layout above was created by using a table with no borders.
- The title spans across all three columns.
- The picture of the skunk in the middle spans across the two lower rows.

Again, let me stress that using a table to control the layout of image is **not** a good idea, we only do it here as an easy way to visualise the structure of a table.

1 **Open the lab06>** animalkingdom.html **file in a browser. It should look like the image above.**

Alter the markup so that the page looks like this:



Checkpoint

Have a demonstrator check your work now.

- ✔ lab06.html
- ✔ animalkingdom.html

Demonstrator:_____

Activity 6.3 Width/height using CSS

Cells conform (by default) to the size of their content. Add more content and the cell will expand to fit it. The widest cell in a column determines that column's width.

Net Weekly Income after all deductions
$287.00
$355.99
$175.45

We can stipulate the width of a table using an accompanying style sheet with the following rule:

```
table{
   width:80%;
}
```

Remember that the widest cell in a column determines that column's minium width. Likewise the cell with the greatest height determines the height for that row.

Tables and the box model

As with all other elements all elements of the table (<table>, <tr>, <td>, <th>) come with their own CSS box. That is each element has (working from the inside) a content area, then some padding, then a border and finally a margin. So styling a table will involve working out which of these properties

need setting to achieve the desired result. In addition there are other properties you may find useful like `height` and `border-collapse` model.

 http://www.w3schools.com/css/css_table.asp

Alignment

Use the `text-align` and `vertical-align` properties in CSS to control alignment of content.

Activity 6.4 A Data Table

The original intent of HTML tables was to facilitate the presentation of tabular data. Remember HTML was designed for the use of the scientific community. No one had any idea that the WWW was going to spread and change as it did and continues to do.

1 **Start a new file in Taco called weather.html.**

2 **Write code to display the table below.**
There is no need to make the borders look like those in the picture. Normal table borders will be fine.

Local Date	Saturday 20th Dec. 2015			Sunday 21st Dec. 2015		
Local Time	01h	09h	17h	01h	09h	17h
Wind Direction (deg)	300	310	310	290	190	180
Wind Speed (kts)	15	18	25	25	5	15
Wave Height (M)	1	1.1	2	1.9	0.9	1

Checkpoint

Have a demonstrator check your work now.
✔ weather.html

Demonstrator:_____

Activity 6.5 Table Accessibility

In this activity you will add accessibility features to your HTML table to make it available and useful to a wider range of people. Remember, not everyone that might want to access your web page has perfect vision, perfect hearing or an understanding of the English language. Increasingly they may be accessing your page using a non-visual browser or other assistive technologies. It requires little thought and very little effort we can make tables far easier for everyone to use.

Many of the features listed here are to assist people who use 'readers', that is applications that literally read out the contents of a web page. Have a look at the movie darren.mpeg in the Movies directory for a demonstration of someone using screen reading software.

Table Caption `<caption>`

The table caption is similar to a heading tag, the contents of the `<caption>` tag are rendered by the browser. The `<caption>` tag must come immediately after the `<table>` tag.

```
<caption>Dunedin Marine Conditions Information</caption>
```

1 **Add a caption to your table.**

Table Heading <th>

Table headings <th></th> are used in place of <td></td> but denote that the contents of the cell are to be read as headings rather than data. By default browsers tend to emphasise <th></th> content. Table headers can be used to denote rows or columns.

1 **Change the appropriate <td></td> tags to <th></th> ones.**

Scope attribute

The scope attribute specifies whether the heading applies to a row (or rows) or a column (or columns). so for instance:

```
<th scope="row">Local Date</th>

or

<th scope="col">01h</th>
```

1 **Add appropriate scope attributes.**
There is one rider here. You can't apply a scope to an <th> that spans a number of rows or columns without using colgroups. We won't ask you to do this in this exercise as it is a little fiddly, however you need to be aware of the issue. The link below explains it all quite well.

 http://www.w3.org/WAI/tutorials/tables/irregular/

<abbr>

 http://www.w3schools.com/tags/tag_abbr.asp

The <abbr> tag defines an abbreviation or an acronym For sighted users a browser usually shows a 'tool tip'. A screen reader will be able to read the contents of an <abbr>. You don't need to use them in this document but they may be useful later.

Checkpoint

Have a demonstrator check your work now.
✔ weather.html

Demonstrator:_____

Activity 6.6 Sensible use of table markup

It is important to approach table markup sensibly and efficiently. Avoid the following pitfalls.

Incomplete tables

Your tables should be complete: each row and column should be of equal height or width. The best way to determine if your table markup is complete is to temporarily set a border width for the table and view it in a browser or use the Outline tool in the Web Developers' Toolbar.

1 **View lab06> IncompleteTable.html in a browser. It looks ok.**

Contacts		
Aaron	021 1234567	22 Forth St
Alice	022 123456	24 Dundas St
Amy	022 852456	241 George Street

2 **Apply a border to the table and you will see the problem:**
Or use in the Web Developers Toolbar in Firefox Outline > Outline Tables > Outline Table Cells

Contacts		
Aaron	021 1234567	22 Forth St
Alice	022 123456	24 Dundas St
Amy	022 852456	241 George Street

3 **There is clearly a cell (or two) missing. This is just poor markup. We have the attributes `colspan` and `rowspan` for exactly these scenarios.**

4 **Add the appropriate colspan value to the top cell so the table becomes fully formed:**

Contacts		
Aaron	021 1234567	22 Some Street
Alice	022 123456	24 Some Other Street

Using empty cells

Just as `rowspan` and `colspan` stop the necessity for incomplete tables they also eradicate the need to use empty cells in most circumstances.

5 **Truly empty cells (`<td></td>`) don't display properly in some older browsers (e.g. Internet Explorer and Netscape) so they aren't sensible.**
The use of `colspan` in the cell beside, or a `rowspan` in the one above (whichever is appropriate in a particular situation), will negate the need for an empty cell in many circumstances.

6 **Almost empty cells such as: `<td> </td>` or `<td><br></td>` are better in the sense that they display properly in browsers. But if a sensible table structure is employed they are not often necessary.**
If you find yourself tempted to use one of these, stop and think very carefully – there is probably a better solution. There are occasions when an almost empty cell is unavoidable, but don't use them indiscriminately.

7 **Open BadTable.html from the lab06 directory and look at the markup.**
In the browser the markup looks like this:

Contacts: Aaron 021 1234567 22 Forth St
Alice 022 123456 24 Dundas St
Amy 022 852456 241 George Street

But the markup isn't very sensible.

1 **Modify the markup so it doesn't use empty cells but looks the same.**

Activity 6.7 Review Activity

Look at your MyHomePages> myStudy page. A table would be a far better way of presenting your (fake) timetable information.

1. **Replace the list of papers and lecture times with a table displaying the same information.**
2. **Add some style rules to style.css to make the table look a little nicer.**
3. **Ensure your table validates and is accessible.**

Lab Completion (1%)

- ✔ layout.html
- ✔ Contacts code
- ✔ myStudy.html + style.css

Demonstrator:_____

HTML III – TABLES (1%)

Lab 7 CSS II (1%)

Reference

 Lecture 8

Check list

Ensure that you are confident doing the following:

- ✔ Utilising box properties
- ✔ Utilising background properties
- ✔ Positioning elements
- ✔ Contextual selectors
- ✔ Grouping selectors

Introduction

Having learnt the basics of CSS structure in earlier labs, this lab revises those concepts and then takes you through some of the more advanced material. In this lab we learn how to position elements using CSS. By the end of the lab you should have a good overview of CSS and be ready to put your skills to use in the next CSS lab.

Activity 7.1 CSS structure

CSS enables much greater control over formatting than straight HTML, but you still have to be aware that your page will look different depending on your viewers' preferences and environment. Firstly, lets just do a quick recap of linking to an external style sheet and adding a few style rules.

1 **Copy lab07 files to your Home directory.**
There are a lot of files and directories in the zip file.

2 **Open your** lab07 > basic.html **file in an editor.**
It should be familiar.

4 **Make a new empty file 'style.css' in your lab07 directory.**

3 **Add a link to** style.css **in the** `<head>` **section of** basic.html**.**

4 **Type the following in your style sheet:**

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

5 **Reload your HTML page to make sure your everything is working as it should.**
It is always worth doing something trivial when you first link an HTML and CSS document, just to make sure things are working. This can save a lot of frustration.

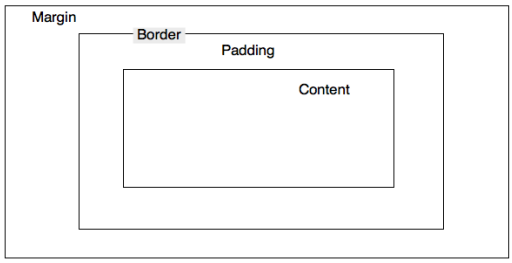
Activity 7.2 Box Properties - recap

Recall CSS uses a "box-oriented formatting model". Put plainly this means that it treats every element it formats as if it were inside a series of boxes. Each element box has an area where the content is displayed, and around that there are areas for padding, border and margin.

1 **Make the following additions/changes to your style sheet (**style.css**):**

```
html{
  background-color: #eee;
}
header{
  margin-top: 25px;
}
h1{
  text-align:center;
  background-color: #eef;
  padding:25px;
  margin:25px;
  border:5px dashed black;
}
```

5 **Can you explain how the above style rules are applied in terms of the CSS box model?**



If you were to add the following rule how big would the gap be between the borders of two adjacent paragraphs? (The Web Developers Toolbar has a ruler under the Miscellaneous option).

_____pixel gap.

```
p{
    margin-top: 30px;
    margin-bottom:30px;
}
```

6 What else is needed to make this CSS validate?

Activity 7.3 Advanced use of selectors

 http://www.w3schools.com/css/css_combinators.asp

Contextual selectors (W3Schools calls them 'combinators') allow you to be particularly specific about formatting.

Normally when making CSS declarations you use simple selectors like this:

```
h1 {
    color: blue;
    font-size: large;
}
```

Contextual selectors

Contextual selectors take the form:

```
selector selector {
    property: value;
}
```

If you added the following to the Fruit example below:

```
ol li {
    color: red;
```

```
}
```

Then the contents of <li> tags found directly or indirectly inside <ol> will be red. The contents of <li> inside <ul> tags will be unaffected unless they are themselves in an <ol>.

If we added the following style rules to the Fruit markup

```
nav a{
    text-decoration: none;
    color: #62537a;
}
```

Only links (<a>) found inside <nav> elements are affected.

```
ol ul li {
    font-style: italic;
}
```

This code sets the content of <li> tags found inside <ul> tags which are in turn contained within <ol> tags to be italic:

```
<h1>Fruit</h1>

<ol>

    <li>Banana</li>

    <li>Apple

    <h2>Varieties:</h2>

    <ul>

        <li>Granny Smith</li>

        <li>Eve</li>

        <li>Braeburn</li>

    </ul>

    </li>

    <li>Kiwifruit</li>

    <li>Orange</li>

</ol>
```

Fruit

1. Banana
2. Apple

Varieties:

- *Granny Smith*
- *Eve*
- *Braeburn*

3. Kiwifruit
4. Orange

Grouping Selectors

Selectors and contextual selectors alike can be grouped together to further economise code by separating each selector with a comma. For example:

```
h1, ol li, h2 em {
  font-style: italic;
}
```

declares that the content of `<h1>` **tags are in italics, as are** `<li>` **tags contained within** `<ol>` **tags and any** `<em>` **tags contained within** `<h2>` **tags.**

Grouping selectors in this way and using contextual selectors wherever possible is preferred and what you should do when you can.

Activity 7.4 Class and ID

To expand the power of CSS the *class* and *id* attributes are provided. These provide an easy way to 'name' particular elements so that you can refer to them easily in your CSS. These are often referred to as 'hooks'. This is useful when you can't refer to a particular element or set of elements easily and sensibly using contextual selectors.

Class

In your HTML markup `class` **behaves like a tag attribute:**

```
<p>What is the most expensive Internet domain name to date?</p>
<p class="answer">In 2010 the domain name sex.com was sold for US$14 million!</p>
```

`class` **has been used to control the format of the second paragraph. The style sheet that accompanies this markup contains the following rules:**

```
p {
  font-size: small;
  color:black;
  background-color:inherit;
}
.answer {
  color: #339933;
  background-color:inherit;
}
```

`class` **is denoted by the use of a full-stop before the class name. When used as above it is referred to as an anonymous class and can be applied to any element. It can also be used like this:**

```
p.answer {
```

```
  color: #339933;
}
```

in which case it will only apply to paragraph elements that have the class value 'answer'.

You can refer to a class as many times as you require within your HTML document.

1 **Open the directory** ClassID **and open the HTML and CSS files it contains. Examine the markup and code.**

2 **The elements in** ClassID > activity.html **that are marked by** `<!--Class-->` **need to be formatted differently from the standard text.**

3 **Establish a class in** ClassID > **activity.css that will set all the text to display:**

- ✔ in italics
- ✔ with a 20 pixel left margin.

Ensure you use a sensible, descriptive name for your class. Class and ID names should describe the element they affect, not the appearance produced by the style. See how '.answer' was used in the example above.

4 **Set the class attribute for the marked elements in** ClassID > **activity.html.**

5 **View your page in a browser to test it. Only the marked elements should be styled by the class you have created.**

ID

ID performs a similar function to class but it is used in instances where the format is used only once within an HTML document. Because of its single-use nature it is often used for fundamental parts of the page structure.

For example our HTML markup might include this:

```
<p id="author">Page created by ...</p>
```

and our style sheet would refer to this id like this:

```
#author {
    margin: 2% 10% 2% 10%;
    background-color:#faf0e6;
    color: inherit;
}
```

ID is denoted by the use of a hash # before it. ID can also be stipulated with its HTML element included like this:

```
p#author {
    margin: 2% 10% 2% 10%;
    background-color:#faf0e6;
    color: inherit;
}
```

but because IDs are unique there is no need to do this. Adopt whichever usage suits you best.

1. **In** ClassID > activity.html **we use** `<p>` **many times, but we want one particular instance to be different from the others. We need to format the author paragraph differently to distinguish it.**
2. **As we only ever intend to have one paragraph on our page that is different in this way we can use** `id` **to identify it rather than class.**
3. **Establish an** `id` **in** ClassID > activity.css **that will make the *footer* paragraph in** ClassID > activity.html **marked by** `<!--ID-->`**:**
   - ✔ grey,
   - ✔ with centrally aligned text
   - ✔ 10px top padding
   - ✔ and a 1 pixel wide top border.
4. **Set the** `id` **attribute for the paragraph in** ClassID > activity.html **to correspond to the id style you have created.**
5. **View your page in a browser to test it.**

Checkpoint

Have a demonstrator check your work now.

- ✔ ClassID _activity.css
- ✔ ClassID _activity.html

Demonstrator:_____

Activity 7.5 Div and Span

By themselves the HTML elements, `<div>` and `<span>`, do not alter your HTML document, but when affected by a style sheet they can radically alter the appearance of a document.

`<div>`


`http://www.w3schools.com/tags/tag_div.asp`

This is the division tag. It is useful in denoting the structure of your HTML page. `<div>` is a *block-level element*, so it places its content on a new line and leaves white space above and below it (like the `<p>` or `<h1>` tags).

A `div` can contain other block-level elements and is used to sensibly group block-level elements together. The `<div>` is often referred to as a *generic block-level element.* In HTML5 many more block-level elements were introduced and so the need to use a `<div>` to group other elements is much reduced. With HTML5 elements can often more sensibly be grouped with tags such as **`<article>`**, **`<section>`**, **`<header>`**, **`<footer>`**, **`<aside>`** or **`<figure>`** which are far semantically richer. However sometimes you just need a generic container.

****

 http://www.w3schools.com/tags/tag_span.asp

 is an *inline element* (like <a> and). The tag enables you to format sections of markup.

In HTML5 you would use the following.

In your HTML Document:

```
<p>The use of <span class="markup">monospace font like this</span>
indicates that the text is a markup example.</p>
```

In your style sheet:

```
.markup {
    color: blue;
    background-color: inherit;
    font-size: small;
    font-family: courier, monospace;
}
```

1. **Open your lab07>** DivSpan **directory. Open the HTML file within it and examine the markup.**
   You will notice that it uses the <font> tag to format parts of the markup. Currently this markup validates as HTML 4.01 Transitional (note the doctype declaration) but we want to make it validate as HTML5.
2. **Create a style sheet that establishes a class that will do the job that the** <font> **tags currently do. Use** **(with your new class) to replace the** <font> **tags in** activity.html**.**
   Remember to use sensible, descriptive names for your class.
3. **View your page in a browser to test it.**
4. **Alter the** doctype **declaration so that the page is an HTML5 document.**

Activity 7.6 Pseudo-selectors and Pseudo-elements

Selectors are normally HTML elements, but there are a number of pseudo-selectors in CSS that enable a variety of formatting effects that do not relate to the markup that appears in your HTML source file. Pseudo-selectors are denoted by the use of a colon (:).

 http://www.w3schools.com/css/css_pseudo_elements.asp

There are four pseudo-classes that relate to the way links in different states appear. Hint, they are in the 'All CSS Pseudo Classes' list at the above reference and:

 http://www.w3schools.com/css/css_link.asp

List them here in the correct order:

1 _____

2 _____

3 _____

4 _____

Controlling Layout

To be fair, creating exciting layout with just HTML is difficult, it was never intended to be used to present documents with a complicated layout. Consequently early developers used tables to control the layout of their web pages. There is a simple example of a table used to control layout in the lab files, tableLayout.html. With the advent of CSS and particularly the *float* and *position* properties that difficulty went away. Even though the float property has been available since 1996 there are still many many millions of web pages that use tables to control the layout of their pages. There really is no excuse to do this. Using tables to control layout shouldn't be done for two main reasons.

- It produces non-semantic markup, the use of a table implies tabular data.
- It makes it difficult to change the layout of a site later (add, remove or move elements) without a lot of work. cssZenGarden (http://www.csszengarden.com) just wouldn't be possible if tables were used for layout.

In the remainder of this lab you will learn about the *position* property. In the next lab you will explore the *float* property.

Activity 7.7 Positioning

 http://www.w3schools.com/css/css_positioning.asp

Relative Positioning

1 Open RelativePosition> relative.html and relative.css in Taco. They are very simple files.

2 In the CSS make the position property of divTwo 'relative':

```
#divTwo{
  background-color:#C7F0D2;
  color:inherit;
  position: relative;
}
```

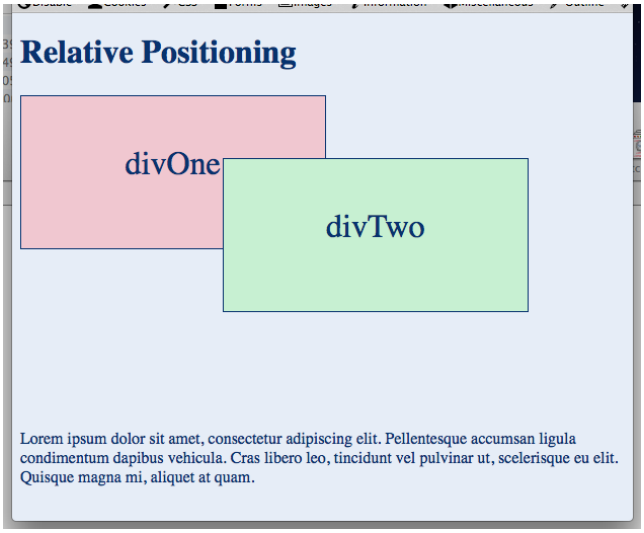
When an element has this position type you can move it relative to its current position using the 'top', 'bottom', 'left', 'right' properties.

3 Add the following:

```
#divTwo{
  background-color:#C7F0D2;
  color:inherit;
  position: relative;
  bottom:100px;
  left:200px;
```

```
}
```

Have a look at relative.html in a browser. If all have gone well you should see:

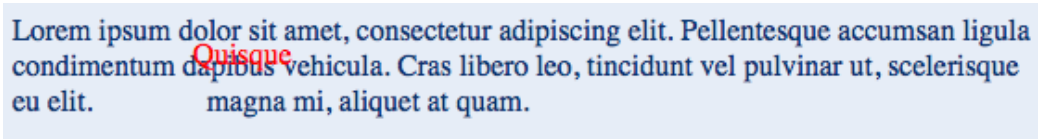


You will notice of course that divTwo has moved across (from the left) 200px and up (from the bottom) 100px. In doing so it now overlays divOne. Notice also (very importantly) that divTwo's *space in the flow is preserved*. That is the <p> haven't moved up to fill the empty pace left by divTwo moving up. The <p> still acts as though divTwo is in its original position; its *space in the flow is preserved*.
Now lets try it on another element.

4 Modify the CSS to include the following rules:

```
span{
    position:relative;
    bottom:25px;
    left:50px;
    color:red;
    background-color:inherit;
}
```

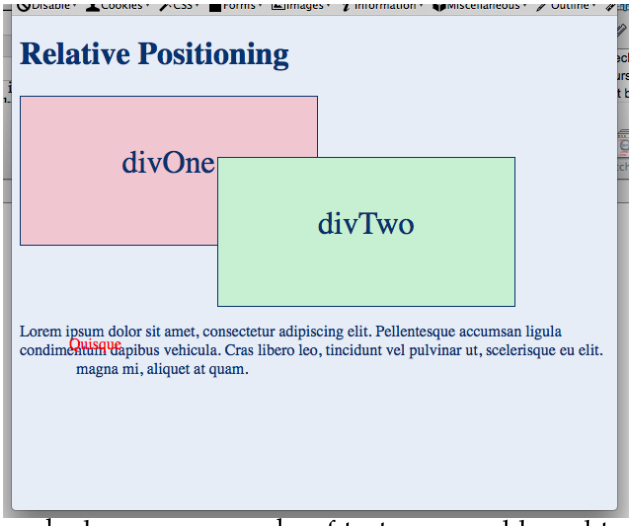
You should see have something like this. Yours may be slightly different due to the width of your browser. Resize the browser window to see what I mean.



Notice again that the space in the flow is preserved and that the selected element moved the appropriate number of pixels from its original location.

5 Experiment with different values.

6 Close up the space between the paragraph and the boxes.

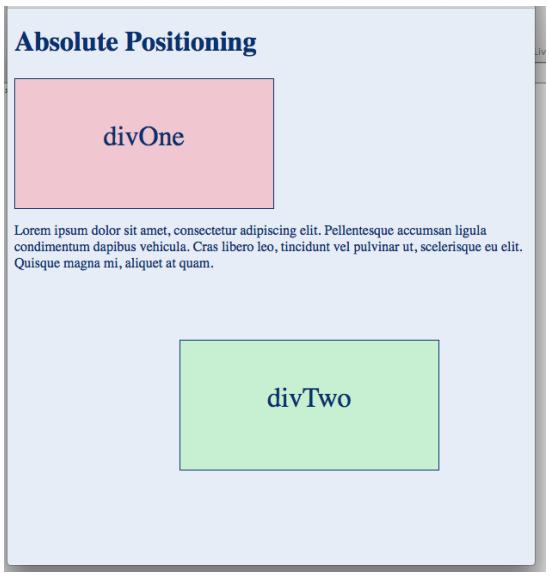


If you had more paragraphs of text you would need to relatively position each one up by 100px to maintain the illusion of normal flow. These flow-on effects can be problematic and so you need to be careful where you use relative positioning.

Absolute positioning

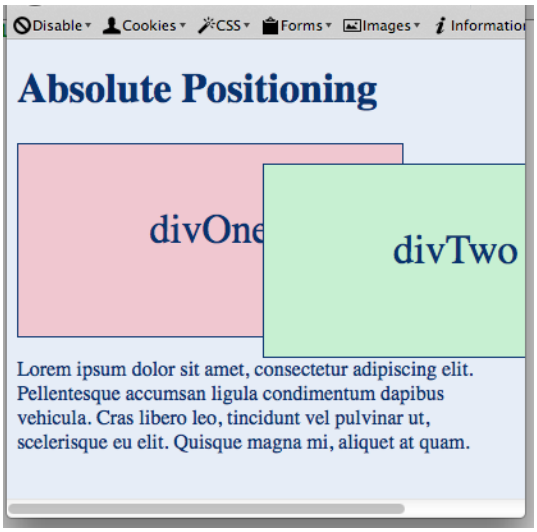
1. **Open AbsolutePosition> absolute.html and absolute.css in Taco. They are the same very simple files.**
2. **Let's try applying the same rules as we did before to divTwo except this time use *absolute* positioning:**

```
#divTwo{
  position: absolute;
  bottom:100px;
  left:200px;
}
```



Well that is different. What has happened this time? For a start notice that divTwo's space in the flow has not been preserved; The <p> has moved right up below divOne as if divTwo wasn't there. DivTwo has been *removed from the flow*. Also divTwo's new position doesn't seem to bear much relation to where it was before. divTwo has been in fact repositioned relative to it's *nearest non-static ancestor*. The nearest non-static ancestor of divTwo is <html> (which is the same as the browser viewport). So divTwo is positioned 100px up from the bottom of the viewport and 200px to the right of the lefthand side of the viewport.

1 Resize the browser viewport to quite small.

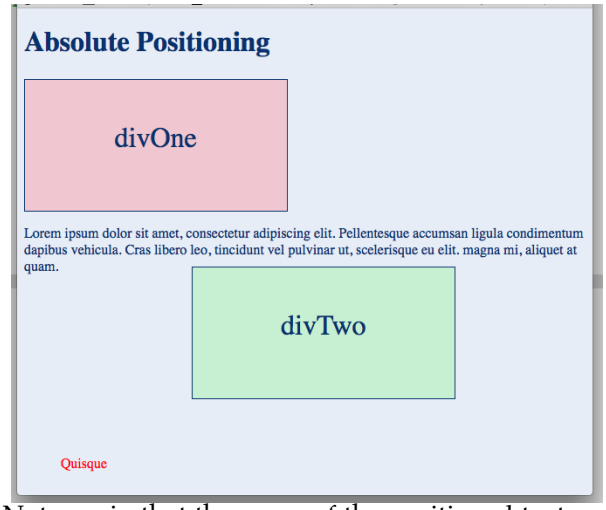


Notice how divTwo keeps a fixed distance from the left and bottom of the browser.

2 Try some other values. What happens if you set the top and/or right instead? Can you use negative values?

3 Let's look at the again. Add the following CSS:

```
span{
    position: absolute;
    bottom:25px;
    left:50px;
    color:red;
    background-color:inherit;
}
```

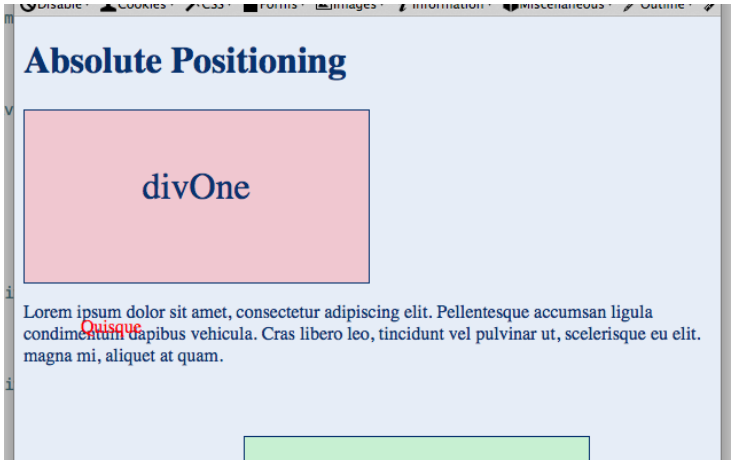


Note again that the space of the positioned text was not preserved and that it too is positioned relative to the bottom left of the viewport.

Changing the nearest non-static ancestor

Let's change the nearest non-static ancestor of an element. By default all elements have a position type of *static*. This means it just works normally in the flow.
In simple terms the ancestors of an element are those which contain it. For instance the ancestors of the `<span>` are (in order) `<p>` `<body>`, `<html>`. But they all only have their default position type, static. So lets just make the `<p>` element's ancestor's position type non-static. That is to say not static, something other than static. Well what are the other position types, they are listed in the w3Schools reference given earlier.

4 Try setting the position type of the `<p>` element to other (legal) values until you achieve this with the spanned element positioned using the `<p>` as its datum.



Lab Completion (1%)

Have a demonstrator check your work now.

- ✔ ClassID _activity.css
- ✔ ClassID _activity.html
- ✔ DivSpan exercise
- ✔ Position exercise.

Demonstrator: _____
Date: _____

CSS II (1%)

Lab 8 Catch Up

Introduction

This is an opportunity to catch-up on any work you may have missed so far. Remember, you can have labs marked late.

There will be no demonstrators in the lab on Easter Friday but labs will operate as usual on Wednesday and Thursday.

Lab 9 CSS III(1%)

Reference

 Lecture 8

Check list

Ensure that you are confident doing the following:

- ✔ Floating elements
- ✔ Inline and inline-block

Introduction

In this lab you will continue your exploration of CSS layout properties.

Activity 9.1 Float

Float was originally introduced to allow text to flow around images and it does this very well. However it also has other uses. In this activity you will look at a number of typical uses for float.

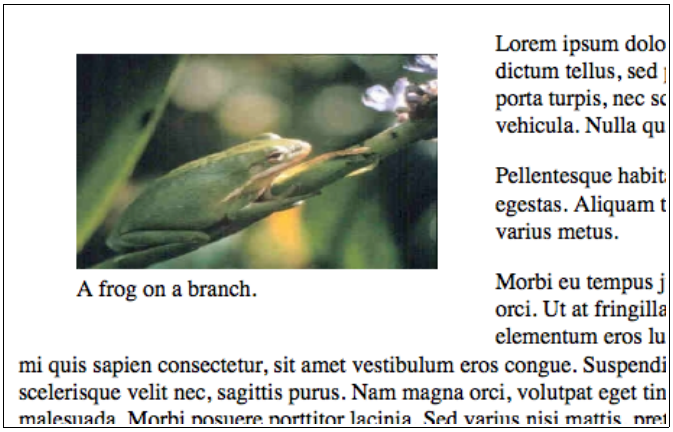
Floating Text Around Another Element.

1. **Open lab09> Float> float.html and floatStyle.css in Taco.**
   FloatStyle.css is intentionally empty.
2. **View float.html in a browser.**
   Everything is displayed on separate lines, just as you would expect.
3. **Now make the <figure> element float to the left and see what happens.**

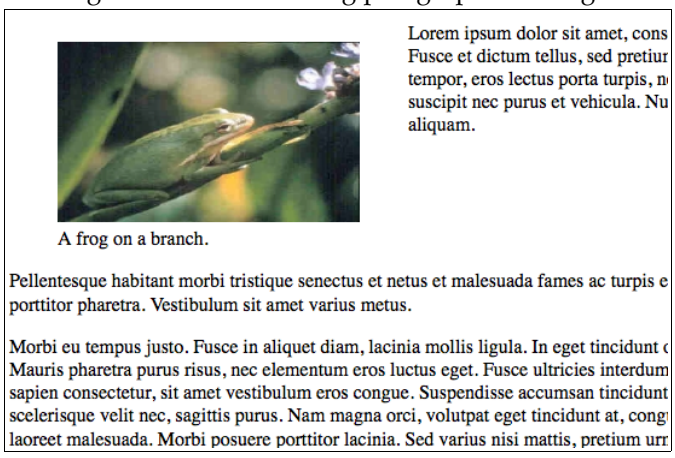
```
figure{
   float:left;
}
```

The text wraps around the figure element. The gap is the default margin that the browser is applying to `<figure>`.

4. **Change the float value to right. Does it behave as you expected.**



So that works very nicely. But let's say I wanted something like the following image with only the first paragraph next to the figure and the following paragraphs starting under the figure?



Currently the figure is floated and so all the following paragraphs move up next to it. How do you stop the floating behaviour?

What we want is for the second <p> to ignore the invitation to move up next to <figure>.

5 Give the second <p> a class name <p class="clear"> and then add the following to your CSS.

```
.clear {
    clear: left;
}
```

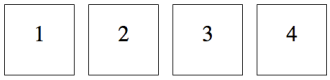
6 View the page in a browser, it should appear as above.
Clearing is a very important part of floating. It allows you to stop the floating behaviour and allow subsequent elements to act in their usual fashion.

Floating Block-level Elements

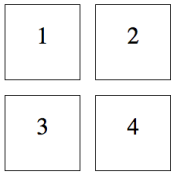
In the example above we were floating elements of no fixed size; they were as high and as wide as they needed to be to hold their content. In this example we will be using elements with a fixed width.

1 Open floatBoxes.html and boxesStyle.css in Taco.
A is a block-level element so it exists on it's own line Hence the boxes are stacked one on top of the other.

2 Make all the divs float left.



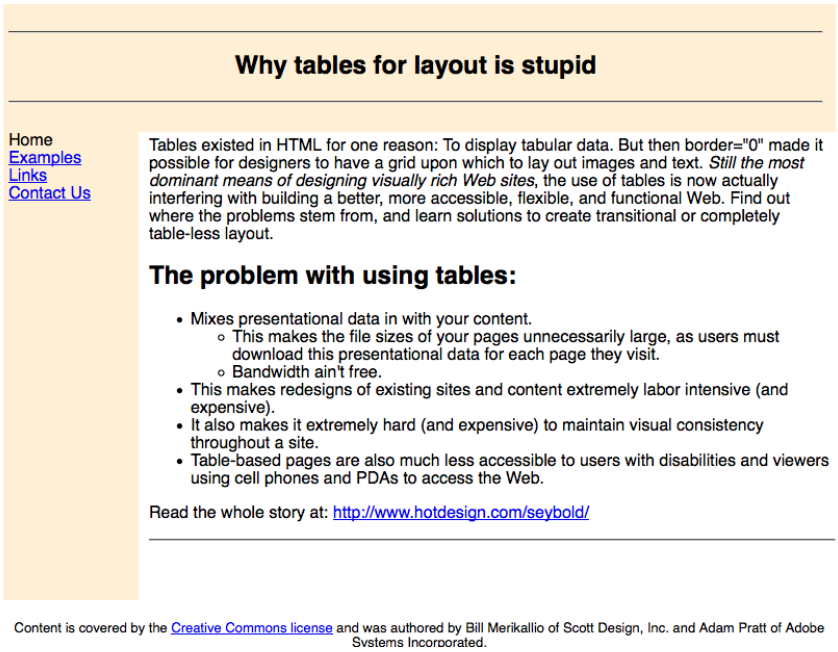
3 Use the clear property to make the boxes float like this:



Being able to float elements like this is very useful for making such things as photo-galleries or for making list items display horizontally,. This might be useful if you wanted to have your list of navigation links display across the page, rather than the default list behaviour of displaying each list item on a new line.

A Practical Application of Float

1 Open practicalFloat.html and practicalFloat.css in Taco. Preview the page in a browser.
This was an important article from 2001 whose message is still relevant.
Look carefully at the html. It is very reasonable semantic markup. Notice the use of the new HTML5 elements. The navigation is inside a <nav> element which browsers treat as a block-level element. Hence the content (in the <section> element, which browsers also treat as block) is on the next line. Obviously not what was intended. What we want to end up with is something like the picture below. Note that the main text has some space on the left. Also note that the copyright information is below the bottom of the both columns.



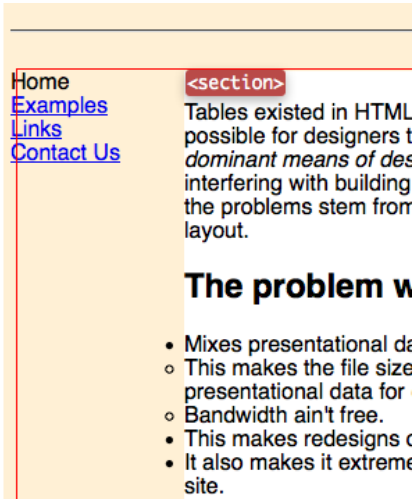
2 Add CSS so that the `<nav>` floats left.
This will allow the `<section>` to come up into the empty space next to the `<nav>`.

That should leave you with the content pushed hard up against the navigation column. So you just need to add a bit of left padding to the `<section>`.

3 Add 10px of margin to the left of the `<section>`.

```
section{
  margin-left: 10px;
}
```

4 Why didn't that work? In the Web Developers Toolbar select Outline> Outline Custom Elements and then type *section* into the top box. All the other boxes should be empty. Press return.



So we can see that although the content of the `<section>` reacts to the `<nav>` the <section> itself doesn't and so rather than the edge of the section having a 10px

margin from the <nav> it is actually from the edge of the <body>. OK, well the <nav> is 130px wide so add another 130px of left margin.

5 Add another 130px of margin.

```
section{
  margin-left: 140px;
}
```

Finally there is just the <footer> to deal with. We don't want that to float up next to the <nav>.

6 Clear the <footer>.

Activity 9.2 Inline elements

You have already met plenty of inline elements <a>, and for example. CSS allows you to make block-level elements to behave like inline elements, that is not start on a new line. This can alleviate the need to use float (with its associated complications) in some circumstances.

1 Using Taco open lab9> inline.html.
It is a very simple document, just five divs. Open the page in a browser you will see the five s, each on a separate line (because they are block-level elements) and each the full width of the page. Block-level elements by default go the full width of their containing element.

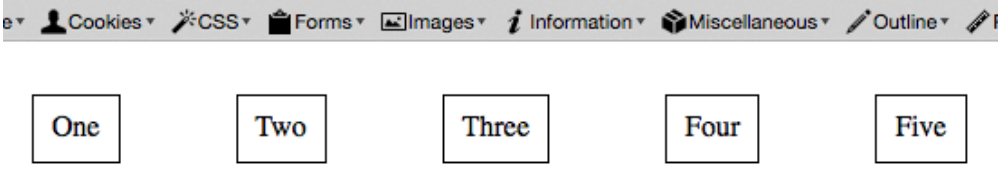
2 Open style.css, it is in the same folder as inline.html

3 Add the following rule to the style sheet, save style.css and view inline.html in a browser.

```
div{
    border: 1px solid black;
    display: inline;
}
```

The s items are now displayed as if they are inline item.
The small gap between the s isn't (as you might think) the 's margin. It is in fact the line breaks from the being on separate lines in the HTML.

4 Add some padding and margin to the elements so that your list looks like that below.
The gap between the top of the s and the browser viewport is 30px of padding on the <body>.

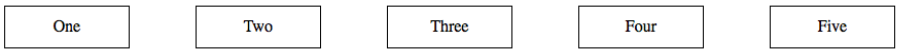


Currently all the s are different widths because of the different lengths of the content. Let's make them all the same width.

5 **Add CSS to make all the `<div>`s 100px wide.**
That didn't work, the `<div>`s stayed exactly the same size. Why? Because you can't set the width of an inline element. It's a rule.
Luckily there is an easy way around this.

6 **Set the display property to inline-block.**
Elements whose display property is inline-block are like block-level elements on the inside, (you can set width and height), but like inline elements on the outside, they don't display on their own separate line.
Your s should now look like this.

7 **Centre the text in the `<div>`s.**



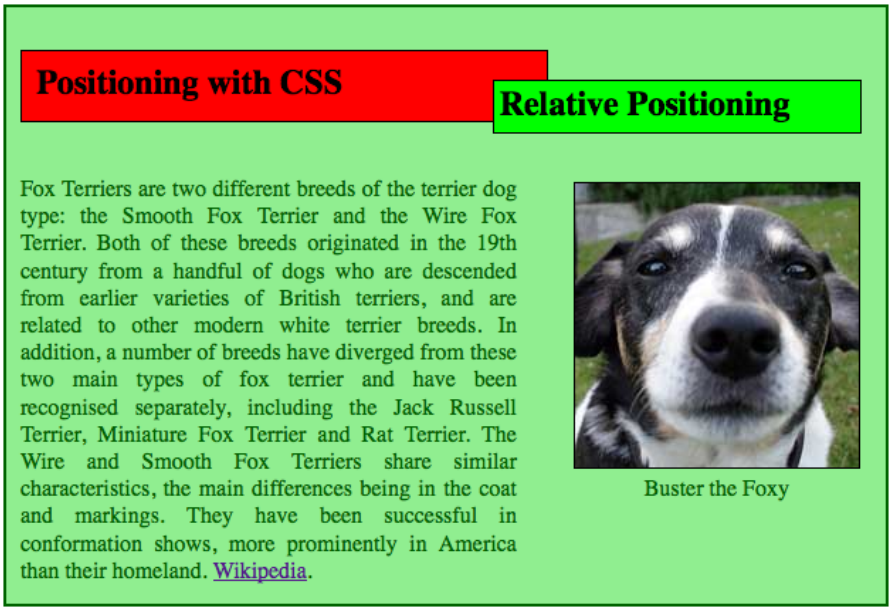
Does it look like any other common web page feature? It looks mighty like some horizontal navigation to me. If those `<div>`s were `<li>`s (remember navigation is marked up as a list of links) and each `<li>` contained a link (`<a>` element) then you would have it. And in fact this is a very common way to markup and style navigation. Another popular method would be to float each of the `<li>` for pretty much the same effect.

Assessment (1%)

Relative Positioning

1 **Open lab09> Assessment> relative.html and relativeStyle.css in Taco.**

2 **Using only relative positioning make the page look like the image below.**
The way to test if it is working properly is to re-size the browser window and make sure everything stays in the right place.
Don't use any margin or padding for this, it can be done entirely with relative positioning.
Note that the edges of the elements all line up nicely; yours should line up nicely to.

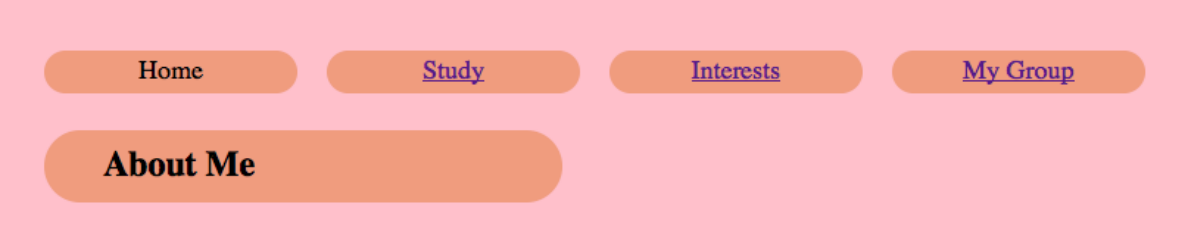


Absolute Positioning

1. **Open lab09> Assessment> absolute.html and absoluteStyle.css in Taco.**
2. **Using only *absolute* positioning make the page look like the image above (except it will say "Absolute Positioning" instead of "Relative Positioning".**
   You will need to set one element to another position type to set the nearest non-static ancestor.
   The way to test if it is working properly is to re-size the browser window and make sure everything stays in the right place.
   Don't use any margin or padding for this, it can be done entirely with absolute positioning.
   Note that the edges of the elements all line up nicely; yours should line up nicely to.

Float

1. **Using the float property modify the navigation in your MyHomePages site to a horizontal layout by using float.**
   You will need to identify which element it is you need to float, that is which elements are responsible for starting the new lines.
   The rounded corners of the <h2> and <li> elements are achieved with the CSS 3 border-radius property.



Lab Completion (1%)

Have a demonstrator check your work now.

- ✔ relativeStyle.css
- ✔ absoluteStyle.css
- ✔ MyHomePages site navigation uses float.

Demonstrator: _____
Date: _____

CSS III(1%)

Lab 10 CSS IV (1%)

Reference

 Lecture 8

Check list

Once you have completed this lab ensure that you understand the following:

- ✔ controlling layout

Introduction

In this lab you will put your CSS skills to the test by moving a table based layout to a CSS controlled layout.

Activity 10.1 Move a page from HTML4.01 Transitional to HTML5

Put all your CSS skills to the test and move the page we provide from HTML 4.01 Transitional to HTML5.

1 Download and save the lab10 files.
ChocRoughFinal.jpg is a screenshot of what the final page should look like.

2 Open lab10 > recipeTable.html in a browser and then in Taco.

3 Examine the html.

Notice that:

- This document uses a HTML 4.01 Transitional doctype (yes it does validate)
- All the style information is contained in the HTML as tag attributes and `<font>` tags.
- A table is used to control the page layout.

4 Answer the following questions:

- What technique is used to realise the vertical line?

  -->_____

- What is the function of the empty cells?

  -->_____

Your task for this lab will be to recreate the same page look using a HTML5 doctype and without using a table to control the layout.

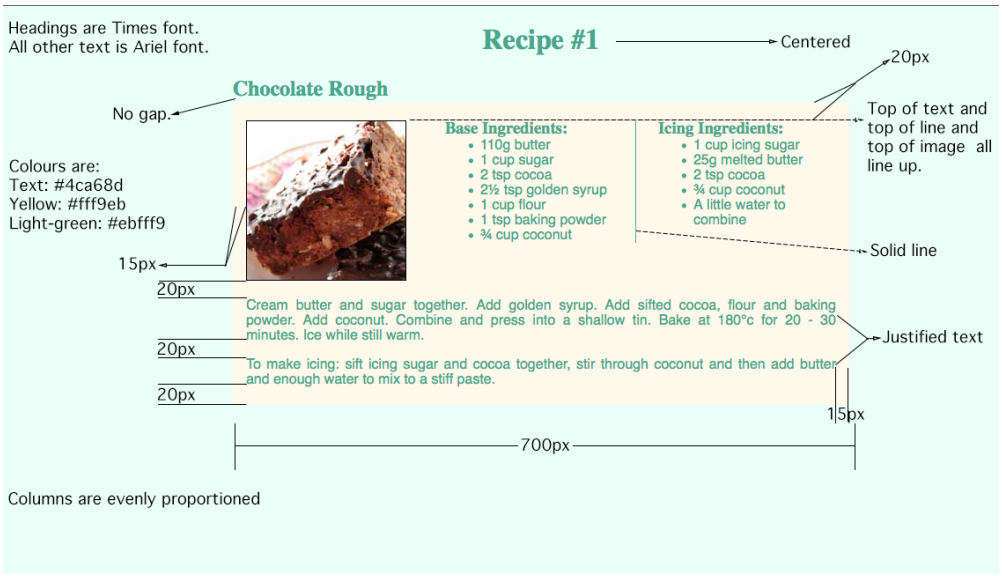
5 Study the specification below before you start.

a. Firstly will need to identify how to mark-up the content with semantic HTML5.

b. Then decide how those various elements are grouped.

c. Lastly what CSS hooks will you need?

How are you going to create the three columns? Absolute or relative positioning?, Float? We suggest that float is the most appropriate technique.



Checkpoint

Have a demonstrator check your work now.
✔ **Plan**

Demonstrator: _____

1 **Remove all the deprecated and table markup from recipeTable.html.**
To save you all that trouble we have supplied a file with everything already stripped out and saved it as recipe.html.

2 **Open recipe.html in Taco. Add some descriptive structural tags to the bare content so that the markup will validate and be sensible.**

3 **Change the doctype to HTML5 and add a link to an external style sheet.**
You will of course need to create this external style sheet.

8 **Recreate the look of ChocRoughFinal.png without using a table and that validates to HTML5 and CSS 2.1.**

9 **Pick one browser to work in and work in it consistently. We will check your work in the browser you have chosen.**

Lab Completion (1%)

Have a demonstrator check your work now.

✔ Sensible, semantic HTML

✔ Well organised, commented and concise CSS.

✔ CSS and HTML files validate.

✔ Page built to specification.

Demonstrator: _____
Date: _____

CSS IV (1%)

Lab 11 Make a Web Site I (1%)

Reference

'A Guide to High Quality Web Development' In Appendix C of this manual.

Introduction

Over the next two labs you will bring together everything you have learned about HTML and CSS so far and put together a simple website about creating good Web sites. We have provided all the text and graphic content so you can concentrate on design and implementation.

Site Specifications

- ✔ Your site should use a novel layout and design, i.e. ***not*** any of the designs used previously in this course.
- ✔ Your entire site should consist of no more than 5 html files but be no less than 3.
- ✔ You should use **all** of text.txt and both images. You may use additional content if you wish. (Some decorative images are also supplied which you may wish to use)
- ✔ Your entire site should be no more than 500KB in total.
- ✔ The file that is the entry point for your site should be called **index.html**.
- ✔ Your site should behave as expected in Firefox and Safari in OS X on the COMP112 lab machines.

Required features:

- ✔ CSS to control page layout.

Required standards:

- ✔ Your entire site must validate as HTML5.
- ✔ Any CSS you use should validate as CSS 2.1.
- ✔ You should consider the usability concepts covered in the course to date.

Additional considerations:

- ✔ markup layout
- ✔ Sensible use of comments
- ✔ Efficient and sensible use of markup
- ✔ Navigation
- ✔ Sensible image use

In this first lab you will undertake three tasks:

- ✔ Marking-up the content
- ✔ Designing the structure of your Web site
- ✔ Designing the look of your Web site

Activity 11.1 Marking-up the Content

1. **Locate the text 'A Guide to Quality Web Development' in Appendix C at the back of this lab book and read it if you haven't already done so.**
2. **Go back through the document and this time mark-up the content of the document.**
   - ✔ **Literally, with a pencil** indicate what is a paragraph, what is heading and what level heading it is, what is a list and so on.
   - ✔ The content will span a number of web pages so remember to indicate where pages start and finish.
   - ✔ Include a suitable top level header (h1) for each page.
   - ✔ Don't think about how pages might look at this stage, this is purely an exercise in indicating the parts and hierarchy of the document. Think about the role

each piece of text plays in the document; is it a heading, if so what level, is it a paragraph, a list item?

Checkpoint

Have a demonstrator check your work now.

✔ Activity 11.1: marked-up text in Appendix C

Demonstrator:_____

Activity 11.2 Designing the Structure

1 **Draw diagrams that shows what content will be on what page and how links between those pages will operate.**

Activity 11.3 Design the Look

1 **Using whatever application you wish (PowerPoint or Adobe Photoshop or Illustrator etc.) produce an image or images of how your Web site breaks up the space in the browser window, a 'wireframe'.**
There is a movie, "Illustrator in 5 minutes" in the lab files for this lab. You might find it very helpful.

2 **Choose a colour scheme.**
Try **www.colorschemedesigner.com** to help pick colour schemes.

3 **Again using software of your choice produce a more detailed image of how one particular page *will look when it is finished*. Be sure to include:**

- ✔ Dimensions and positioning (in pixels or percentages) of things such as text blocks, tables and images.
- ✔ Colours (specify using RGB values.)
- ✔ Font families and sizes.
- ✔ Navigation features; how will you indicate something is a link or what page you are on?
- ✔ Text alignment.

Your aim here is to **produce a picture that shows, as closely as possible, what a final page will look like** and how typical content would be laid out in it. There are examples on the lab walls of the images that were produced as part of the Computer Science web site development.

You must use an original layout design, e.g. you can't use the 'Why Tables for Layout is Stupid' layout.

Lab Completion (1%)

Have a demonstrator check your work now.

✔ Activity 11.1: Marked-up content.

- ✔ Activity 11.2: Diagram of links.
- ✔ Activity 11.3: Design drawings.

Demonstrator: _____
Date: _____

Lab 12 Make a Web site II (2%)

Reference

'A Guide to High Quality Web Development' In Appendix C of this manual.

Introduction

In this lab you will finish making the Web site you started in the previous lab. Before you begin you might like to review the specifications.

Activity 12.1 Create and Test a Template.

1. **Using the design you created in the last lab write an HTML page that will act as a template for all the other pages in your Web site.**
   This page will contain only markup that is common to all pages in the site, for instance structural elements such as divs, navigation, headers and footers and so on. Think carefully about how you will handle navigation in the template.
   You will need to develop the CSS file in conjunction with the HTML template.
   Remember to make sure both the HTML and CSS validate.
2. **Add the marked-up content for one of your pages and ensure it displays as you intended in both Safari and Firefox on the lab Macs.**
   Fix any errors before proceeding.
3. **Strip out the content of the page (so that you have just the structure left and save it as template.html.**

Checkpoint

Have a demonstrator check your work now.

✔ Activity 13.1: Template created and tested.

Demonstrator:_____

Activity 12.2 Create the site

1. **Using template.html as a base create the Web site by adding the marked up content.**
   Remember to use the images too!
2. **When you are confident that it all works properly upload it to the Web server.**
3. **Check it still works properly when viewed from the server.**

Checklist

Before asking a demonstrator to mark your site check the following:

Site is uploaded to CSnet.

All HTML validates to HTML5

All CSS validates to CSS 2.1 (with no warnings)

There is a footer on every page containing at least the author's name and the last updated date.

The <p> on the Construction page uses character entities.

There is no link to the current page.

You have spell checked the content.

Lab Completion (2%)

Have a demonstrator check your work now.

✔ A complete, valid Web site uploaded to a Web server.

Demonstrator: _____
Date: _____

MAKE A WEB SITE II (2%)

Lab 13 Project Plan Marking (3%)

Introduction

In this lab we will mark your project plan. You will not be able to do all the work required for this lab in the allocated lab time, this lab session is set aside for marking your plan, not producing it.
The documents you are required to produce are the same as those for the Make a Website lab earlier in the course.

Requirements

You need to produce documents for the following:

- Site Map
- A number of wireframes showing the general design.
- Two (or more) detailed designs for (at least) the index page and a typical content page.

Marking

You must get your plan signed off during your own scheduled lab session!

A demonstrator will come and see you during your lab time and discuss your plan with you and they will award a grade between 0 and 3%.

Finally

Put your design file in a folder and compress it. You will need to submit these when you submit your Project later in the semester. The compressed file is date stamped so please don't change the files in the compressed folder otherwise the date stamp will change. If you need to change your plan you will need to produce a new plan and have that change approved.

Lab Completion (0 - 3%)

Have a demonstrator check your work now.

✔ A complete plan

Demonstrator: _____
Mark _____

PROJECT PLAN MARKING (3%)

Lab 14 Catch-up

Introduction

This is an opportunity to catch-up on any work you may have missed so far. Remember, you can have labs marked late.

Lab 15 HTML IV – Forms I (1%)

Reference

 Lecture 15

 http://www.w3schools.com/html/html_forms.asp

Check list

Ensure that you are confident doing the following:

- ✔ Basic form structure
- ✔ Using check boxes and radio buttons
- ✔ Creating lists and menus
- ✔ Using text fields
- ✔ Using buttons
- ✔ Using scripts with forms
- ✔ Controlling form layout

Introduction

Forms allow your viewers to interact with your web site. Often forms are used to enable information to be sent from the web site user to your web server; however, forms can be used for other purposes including serving as navigation tools.
Learning how to use forms is the last of the key areas of HTML we will cover.

Activity 15.1 A basic form

Forms enable you to interact with your users and gather information from them. Combined with other technologies forms can be very powerful. Forms may contain a variety of elements including text fields, check boxes, radio buttons and text areas.

Make your own Google search

1 Create a new HTML document called myGoogle.html **and add the following markup to the body.**

```
<form method="get" action="http://www.google.com/search">
  <p>My own <img src="http://www.google.com/logos/Logo_40wht.gif" alt="Google
  Logo."> search site</p>
<!-- Form input will go here. -->
  <p><input type="submit" name="btnG" value="Search"></p>
</form>
```

2 Add the following in the head section.

```
<style type="text/css">
  <!--
    img{vertical-align:middle;}
    form{border:1px solid green;}
  -->
</style>
```

3 View `myGoogle.html` **in a browser.**
You will see a form that has a submit button (with the label "Search") but nowhere to enter any input.
Firstly lets add a simple input field.

4 Add the following markup where indicated by the comment.

```
  <p><input type="text" name="q" size="12" maxlength="255" value=""></p>
```

5 Fill in the form and submit it. Google processes this form for you.
You can have any form processed by Google's search engine provided you specify the form method and action as used above and use the name "q" for the form element collecting search terms and the name "btnG" for the submit button. **These names are specific to the Google search forms; it would not be sensible to use them in other forms.**

Activity 15.2 Check boxes and radio buttons

1 Replace the markup you added above with the following.

```
<p>
  HTML <input type="radio" name="q" value="html">
```

```
  Web Design <input type="radio" name="q" value="web design">
</p>
```

2 **View your altered page in a browser and notice that you can only select one of these options at a time.**
Radio buttons are ideal when you want to limit choice to only one item from a selection.

3 **Change the type of these two inputs to** `checkbox` **and reload you page to see the difference.**
Notice now that you can select both options if you wish to.

4 **Fill in the form and submit it.**

Activity 15.3 Lists and menus

 http://www.w3schools.com/tags/tag_select.asp

1 **Replace the markup you added above with the following to provide a list of options:**

```
<p>
  <select name="q" multiple>
    <option>Resources</option>
    <option>Free</option>
    <option>Tutorial</option>
  </select>
</p>
```

2 **View your page in a web browser and you should see a list displaying the options. Because we have specified** `multiple` **within the** `<select>` **tag we can select as many items on the list as we want; to do this press the command key as you click.**

3 **Add the attribute** `size` **to the** `<select>` **tag and set its value to** `1`**. Remove the** `multiple` **attribute (**`<select name="q" size="1">`**).**

4 **Reload your page in the browser and you will see that the list is now a pop-up menu which allows only one selection to be made.**

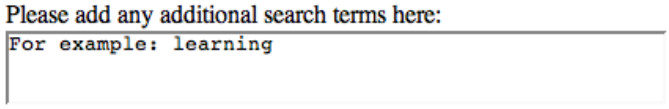
Activity 15.4 Larger text fields

Earlier we created a one line text field with the input tag. The <textarea> tag can be used to create a text field of multiple rows.

1 **Replace the markup you added above with the following.**

```
<p>Please add any additional search terms here:<br> <textarea cols="50"
name="q" rows="2">For example: learning</textarea></p>
```

2 **Reload your page in the browser and you should see this addition to your page:**



3 **Add some text to the field. When you have exceeded the visible portion of the field notice that a vertical scroll bar appears (in some browsers both horizontal and vertical scroll bars appear).**

4 **Notice that any text that is placed between the** `<textarea>` **tags appears in the field in the browser.**

5 **Quite commonly no default text is needed (or wanted) in a text field. You can achieve this by not using any content between the tags. Delete the "**For example: learning**" text and add a placeholder attribute to the** `textarea` **tag,** `placeholder="Search Terms Go Here".`

Activity 15.5 Buttons

1 **You have already seen how to add an all important submit button to your form. It is courteous to also add a reset button especially to larger forms. A reset button allows your user to delete any information they have entered into your form. Add the following markup to your form below your submit button:**

```
<input type="reset" value="Reset">
```

2 **Reload your page in the browser, enter some values into your form and then hit the reset button to see the effect.**

3 **You can also add a generic button that you can customise to do what you please.**

Checkpoint

Have a demonstrator check your work now.

✔ myGoogle.html

Demonstrator:_____

Activity 15.6 Using scripts with your forms

Forms really come into their own when the data they collect is used by a script (that's a computer program to you and me) on the server. Writing scripts is really beyond the scope of this course (you get a brief look at PHP later in the course) so we will provide a script for you to use. The script we provide accepts information (from a form) and then sends that information to an email address specified by the form.
First you will write the form that gathers and sends the information.

1 **Create a new HTML file and save it as emailForm.html and add the following markup in the body section:**

```
<form method="post" enctype="application/x-www-form-urlencoded"
```

```
action="http://www.cs.otago.ac.nz/112bin/112formslab.cgi">
```

The `<form>` tag above contains a number of attributes; they are important! In particular the `action` attribute specifies the location and name of the script that will process the form data. The `method` attribute specifies how the data is packaged. See the following reference for more about forms.

 http://www.w3schools.com/tags/tag_form.asp

Now we need to add some elements to our form to instruct it how to be processed:

The data we need to send is specific to the script that will process it. The script *expects* there to be data from an input called 'recipient' (see below). Different scripts would require different inputs; forms are tailor made for the scripts they interact with.

2 First we need to provide an email address that the form will be sent to. Add the following to your form:

```
<input type="hidden" name="recipient" value="you@student.otago.ac.nz">
```

Replace *you@student.otago.ac.nz* with the email address you want to receive the output of this form at.

Notice that this element is of the `type` "hidden". This means that it will not show on your web page – it is part of the form by necessity, but is not something you need (or potentially want) your viewers to see.

3 With this particular script we can specify the subject of the email, you do this by passing it an input whose name is "subject":

```
<input type="hidden" name="subject" value="Subject of resulting email">
```

In this example we are going to make a form that collects a users name and email address. Replace *Subject of resulting email* with the an appropriate title for the emails you will receive.
We need to provide the location of the page that our users will be directed to after they click the submit button. Add:

```
<input type="hidden" name="followup-page" value="http://www.cs.otago.
ac.nz/comp112/followup.html">
```

You must use an absolute URL). Normally such pages thank the viewer for their information and (more importantly) provide a link back to the site they came from.

Even though hidden input fields don't display in the browser, they still need to be inside a block-level element to validate. You should choose the most appropriate tags for each situation you use this script in. In this instance placing a division (`<div>`) around the hidden inputs is sensible.

4 If you want to collect your user's email address (so that you can reply to them), you should request their email address like this:

```
<input type="text" name="sender">
```

Their email address will appear in the "From:" field of the email the form produces making it very easy for you to reply to them.

5 **Add some suitable text that prompts your viewers to type their email address into this box.**
Note that the address entered by your viewer may contain errors, so although in an email application it will look like it has been sent from the address, you may receive an error if you try to reply to it.

6 **Add other elements that collect the first name and surname of your user.**
It may be sensible to put these above the email address request. Add appropriate text to explain how to use the form.

7 **Don't forget to add a submit button.**

8 **When you have finished constructing your form, view it in a browser. Fill in your form and submit it. Then check your email to see the output.**

The information you have requested (except the email address) should appear in the body of the email – the name of the form element followed by the value entered. For example:

```
firstname=Nick

surname=Meek
```

This script for sending form information is fairly flexible. So long as you include elements specifying the *recipient*, the *follow-up page* and the *subject* as we have done above, you can use add as many other `<input>`s as you want.

2 **Add two more inputs to collect other information, phone number and address maybe. Don't forget to give the `<input>` elements sensible name attributes.**

Checkpoint

Have a demonstrator check your work now.

✔ Form works

Demonstrator:_____

Summary of 112formslab.cgi usage

This script allows the information entered into a form to be sent to an email address of your choosing.

The content of the form appears in the body of the resulting email as element name-value pairs. For example, if the following appeared in your form:

```
<input type="text" name="surname">
```

And your user enters the word “Smith” into that text field then the name-value pair returned to you in the resulting email will be:

```
surname=Smith
```

Required markup:

These elements are required for the script to operate correctly. They are specific to the script we are

using and should be properly commented, as indicated

- `<form method="post" enctype="x-www-form-urlencoded" action="http://web112.otago.ac.nz/112bin/112formslab.cgi">`
- `<input name="recipient">` – the email address to which the results of the form will be sent.
- `<input name="subject">` – the subject of the email resulting from the form.
- `<input name="followup-page">` – the page the user is sent to after they submit the form. This must specify an absolute URL.

Select an appropriate type for each input element (e.g. type="hidden"). The script will work regardless of the type stated but you should make sure your choice is sensible.

Optional markup:

- `<input name="sender">` – use this if you wish to be informed of the sender's email address. The value of this field is displayed in the "from:" field of the email resulting from the form.
- Any other input fields, textareas, check boxes etc.

Activity 15.7 Forms and Accessibility

The layout and other visual clues in forms give a lot of help and guidance to sighted users on how to use and navigate the the form. However people using reader software can find forms an incoherent nightmare.

1 **Using Taco open toygerOrderForm.html.**
It contains the basic HTML to create a form. Note the *action* attribute is left empty, you will fill this in later. During the rest of this activity you will write HTML that will add mark-up usability and accessibility features

Activity 15.8 Fieldset and Legend

<Fieldset>

`<fieldset>` is a block-level element that allows you to group related form elements together. This is not only useful as an accessibility feature but also has very positive implications for using CSS as, naturally, `<fieldset>` elements can be styled using CSS in the same way as any other element. For instance:

```
<form method="get" enctype="x-www-urlencoded" action="">
  <fieldset>
    <ul>
      <li>Firstname: <input type="text" id="firstname"></li>
      <li>Surname: <input type="text" id="surname"></li>
      <li>Email address:<input type="text" id="sender"></li>
      <li>Phone number: <input type="text" id="phone"></li>
```

```
    </ul>
  </fieldset>
```

2 **Add <fieldset> elements to your document.**
Remember, they are supposed to group related form elements,

3 **Save and view your page.**
By default <fieldset> are often shown with a border. To disable the border you would add the following to your CSS:

```
fieldset{ border:0px;}
```

<legend>

The inline element <legend> can only appear within a <fieldset>. In fact to validate as Strict an <fieldset> must have a <legend>. It is used to add a label or title for a <fieldset> which is useful to both sighted and non-sighted users. For instance:

```
  <fieldset>
    <legend>Contact Details</legend>
```

4 **Add appropriate <legend> elements to your document.**

5 **Save and view your page.**
If you didn't want the legends to be visible you could add the following to your style sheet:

```
legend{   display:none;}
```

<Label>

Sighted users can usually see what the label for each data entry element is and so know what is expected by the page layout. Users unable to see the forms layout can be assisted by the <label> element which allows you to associate some text with a form entry element. As an additional advantage once a label is applied to to an element clicking on the label has the same effect as clicking in the entry field. This is especially useful for check boxes and radio buttons which can be hard to click first-time every-time.

Note that <label> can be implemented in one of two ways. One way wraps the control and its label up together. For instance:

```
<ul>
  <li>
    <label><input  type="radio"  name="contactpreference"  value="email">
Email</label>
  </li>
```

The second way is useful if the control and its description text are located some distance apart either visually or in the HTML; the different ways of using <label> can also be exploited in CSS as we shall see later.

```
<ul>
```

```
  <li>
    <label for="firstname">Firstname:</label> <inputtype="text"
id="firstname">
  </li>
```

You should use `<label>` for all input fields as a matter of course.

6 **Add labels using the id method to the fieldset containing the textfields and the wrap-around method to the other inputs.**
The reason why we are doing this will become clear later.

Checkpoint

Have a demonstrator check your work now.

- ✔ Fieldsets
- ✔ Legends
- ✔ Labels

Demonstrator:_____

The `tabindex` attribute.

The tabindex attribute is added to input elements to set the order in which they are visited when 'tabbing' through them. Again this is especially useful for non-visual browsers but also makes using forms easier and more efficient for sighted users as well. Input elements naturally tab in the order they appear in the HTML. If that is the way you want them to tab then there is no need to add a tab index. However if you want input elements to tab in a non-natural order add a tabindex attribute to the input element as shown.

```
<li>
  <label       for="firstname">Firstname:</label><input       type="text"
id="firstname" tabindex=”1”>
</li>
```

The `title=" "` attribute

The title attribute can be added to many elements including input fields, images, links, labels and so on. Their purpose is to add extra information about the contents or use of the element. Readers may read the title aloud, browsers often display them as tool tips. For instance:

```
<ul>
  <li><label>First  Name  <input  type="text"  tabindex=”1”  name="fname"
title=”Your first name and any middle names”> </label>
  </li>
  <li><label>Last  Name  <input  type="text"  tabindex=”2”  name="sname">
</label>
  </li>
```

```
  <li><label>D.o.B.     <input     type="text"     tabindex=”3”name="dob"
title=”yyyy-mm-dd”> </label>
  </li>
</ul>
```

7 Add any tab indexes or title attributes you think appropriate.
Think carefully before adding tabindexs, they are rarely needed and only usually because of a columnar layout.

Activity 15.9 Review Activity

While it is often argued that using tables to structure forms is justifiable, really it isn't. Your html and CSS skills should now enable you to style forms without the use of tables.

In this review activity you will take the accessible, well structured mark-up you created in the last activity and finish it by styling it with CSS and adding the necessary html so that it works with the script, 112formslab.cgi

Attach a style sheet to accessabilityForm.html and add code to achieve the following:

- ✔ Remove bullets, padding and margin from the lists.
- ✔ Set background colours for `html, form, fieldset` and `legend`.
- ✔ Set the font size and face (14px, Arial in our example).
- ✔ Set the form width to 450px.

To nicely line up the label and input fields of the text inputs we need to be able to style them differently than the other fieldsets.

8 Set the Contact Details class:

```
<fieldset class="textInputs">
```

```
  <legend>Contact Details</legend>
```

9 Then add the following to your style sheet:

```
.textInputs label{
  float: left;
  width: 150px;
}

.textInputs input{
  float: left;
  width: 230px;
}
```

Because the text `<input>` element is not contained within the `<label>` tag we can size and position the label and the input separately. So, we make the `<label>` elements 150px wide and `float` them left, then we `float` the `<inputs>` next to them.

Of course when `float` is employed it is important to set the `clear` property of elements around the floated items appropriately. In this case we need to set `clear` for `li`.

```
.textInputs li{
  clear:both;
}
```

10 You will also need to set a `height` for the `li` (we used 1.5em) and provide some `margin` to space them apart (we used a bottom margin of 5px).

11 Add extra html and CSS as you think necessary to make the page look like the example.
Hints:

- The list items `<li>` that contain the checkboxes and radio buttons are all floated left using a single class.
- The whole <form> is floated left (inside the <body>) and a containing the image is floated left next to it.

12 Ensure your page displays correctly and validates.

13 Add the code necessary to toygerOrderForm.html so that it uses the 112formslab.cgi script and sends information entered in the form to your *student email address*.
A demonstrator will want to see the email sent by the script to your student email account, please have webmail open.

Lab Completion (1%)

Have a demonstrator check your work now.

Demonstrator:_____

Lab 16 HTML V – Forms II (1%)

Reference

 Lecture 15

Check list

Ensure that you are confident doing the following:

- ✔ Using HTML5 form elements.

Introduction

The HTML5 specification includes new attributes and input types. Through the use of new attributes and input types, a browser will validate certain form fields automatically. In addition there are some new form input types (such as calendar) which provide an easy and standard way for collecting some information from the user again without the use of JavaScript.

However, current implementation of many of these new features is patchy and inconsistent. As newer versions of browsers are released this will improve. For the moment use these new attributes with care and quite possibly use a JavaScript fallback.

In this lab you will explore some of these new capabilities. In the first set of activities you will update your form from the previous lab to take advantage of the new form attributes. After that you will learn about some other new types and attributes.

1 **Open your form from the previous lab in Taco.**
In the first set of activities you will add some HTML5 elements to that form.

Activity 16.1 Validate the name inputs

If you want a browser to ensure a particular input text is filled in use the new '`required`' attribute. The browser will not submit the form until all `required` inputs are filled in. The browser will display some sort of error message but exactly what that is will be different from browser to browser.

1 **Add the required attribute to the first input field.**

```
<input type="text" id="firstname" name=”firstname” required>
```

This attribute `required` (which doesn't require a value) lets the browser know that it should apply the fields validation rules (if any) that apply to this type of field and to not submit the form if the field doesn't validate. In the case of text fields all they need to do to be valid is not be empty, even a single space is sufficient to pass validation.

2 **Save the file and preview it in Firefox.**

3 **Don't fill in the First Name field and click submit.**
The form doesn't submit and a browser generated error will be shown.

4 **Try viewing the page using different browsers.**
The error message you get may be different or you may not get one at all. As I said, support is patchy at the moment.

5 **Add the required attribute to the Surname field too.**

6 **Save your file.**

Activity 16.2 Validate the phone number

HTML5 also has a telephone number input type, `tel`.

1 **Change the input type of the Phone number input to `tel`.**

```
<input type="tel" id="phone" name="phone">
```

2 **Remember you will need to include the required attribute if you want the browser to validate the field.**

3 **Save and test your page.**
There is no special validation for `tel`, how browsers treat this input type is up to the individual browser, the iPhone for instance displays a telephone number input pad when this input field has focus.

Activity 16.3 The pattern attributes

As you have seen from the last two examples the built-in validation supplied by a browser may be insufficient or inappropriate. The `pattern` attribute allows you to specify a pattern for the required input. This pattern is expressed as a *regular expression*.

1 **Add a pattern attribute to the Phone Number input and use the following regular expression.**
This pattern means that the input should be made up of one or more digits (0-9), spaces, hyphens and parentheses.

```
<input type="tel" … required pattern=”[0-9 ()-]+”>
```

2 **Check your form again.**

3 **Add a pattern to the First and Last name input fields that ensure that names consist only of upper and lower case letters, spaces, hyphens and apostrophes.**

Activity 16.4 Validating the email field

Checking the email field could hardly be easier, HTML5 provides an email input type.

1 **Change the email field input type to '`email`' and add the `required` attribute.**

2 **Check your page again.**

3 **What will the browser allow as a valid email address?**
If you think the browsers built-in validation pattern is a little lax you can of course add a more rigorous pattern of your own. Google for more on this topic, there is lots and lots.

Activity 16.5 Ensuring a radio button is selected

We need to ensure that at least one of the radio buttons for contact preferences is selected. Once more the required attribute will work well for us.

1 **Add the required attribute to each of the radio button inputs.**

2 **Check your form again.**

Activity 16.6 The placeholder attributes

A practice that has become common is to place some text in an input field that guides the user as to what type of input is expected. Previous to HTML5 this was either done through some tricky JavaScript or by giving the input field an initial value, both methods had drawbacks. HTML5 has solved the problem with the `placeholder` attribute.

1 **Add a placeholder attribute to the First Name input field.**

```
placeholder="Please enter your first name"
```

2 **Add sensible placeholders for the other text inputs.**

Using only HTML5 you now have a form which validates in a manner very similar to that which could be achieved using JavaScript, but with a lot less effort and without requiring that the browser have JavaScript enabled.

Activity 16.7 What browser are you using?

In the next few activities you will explore some of the other new types and attributes. However as you will discover the support across browsers varies considerably. In the following activities you should check your page in at least Firefox, Opera and Chrome on OSX and note the differences in each activity.

The browsers in the labs are installed close to the first day of semester, some time after I write this so I can't say with any certainty what browser versions will be installed on the lab machines.

1 **Record the version numbers of the browser versions installed on your lab computer.**
You can find the version from the About <browser name> menu item under the <browser name> menu item.

Firefox: _____

Safari: _____

Chrome: _____

Opera: _____

Activity 16.8 Exploring the new HTML5 elements

1 **Open Taco and start a new HTML5 document.**

2 **Compose a valid HTML5 document that demonstrates the following HTML5 elements and record how they function in the browsers you listed above.**
See http://www.w3schools.com/html5/html5_form_input_types.asp and http://www.w3schools.com/html5/html5_form_attributes.asp for lots of good material.

- **<input type="time">**

  Firefox: _____

  Safari: _____

  Chrome: _____

  Opera: _____

- **<input type="date">**

  Firefox: _____

  Safari: _____

  Chrome: _____

  Opera: _____

- **<input type="url">**

  Firefox: _____

  Safari: _____

  Chrome: _____

  Opera: _____

- **<input type="number">**

  Firefox: _____
  Safari: _____
  Chrome: _____
  Opera: _____

- **<input type="range">**

  Firefox: _____
  Safari: _____
  Chrome: _____
  Opera: _____

3 In one of the above elements include the 'min' and 'max' attributes.

Lab Completion (1%)

Have a demonstrator check your work now.

- ✔ HTML5 form validation
- ✔ Other new HTML5 form elements

Demonstrator: _____
Date: _____

Lab 17 JavaScript (1%)

Reference

 Lecture 16

 http://www.w3schools.com/js/js_intro.asp

Introduction

JavaScript is a client-side technology that can be used to greatly enhance the user experience of a web page. As it works client-side the functionality provided by JavaScript occurs in 'real time' without requiring a page reload.

In this lab we can give you but the merest taste of JavaScript.

One final point; don't worry too much about the details of what is going on in this lab. The point of these JavaScript labs is not to teach you how to program (we have other papers to do that). The point of these labs is to give you an idea of the sorts of things that can be accomplished.

Activity 17.1 Embedding JavaScript

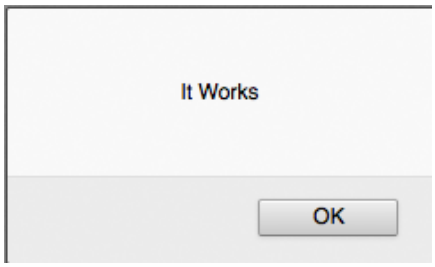
Like CSS JavaScript can either be embedded in an HTML document or linked to it, as with CSS the preference is to link, however embedding has its place. We will take a very quick look at embedded JavaScript first.

1 **Open Taco and start a new HTML5 document.**

2 **In the <head> section of the document add the following:**

```
<script>
  alert("It Works");
</script>
```

3 **Save the document as JSTest.html and view it in a browser, a dialogue box should appear as the page opens.**



If it has that means JavaScript is installed and enabled on your browser. Well done, you have successfully added a JavaScript program to a web page. If you had further JavaScript you wished to add you would just put it between the `<script>` tags.

Activity 17.2 Linking to a JavaScript file

As with CSS the preference is to link JavaScript files.

1 **Remove the script from the head of JSTest.html and instead include the following which links a JavaScript file.**

```
<script src="test.js"></script>
```

2 **Add an alert to test.js.**
Again you just want to make sure that the documents are linked correctly.

```
alert("It Still Works");
```

So, what has happened here? As the browser was parsing the HTML it came to the link to the JavaScript file. It downloaded that file and then executed the first JavaScript command it found.

What this has told **you** is that the file is linked correctly and that the browser you are using can/will execute JavaScript. Doing a simple test like this can save much frustration later on.

3 **Open JSTest.html in a browser, the alert should show as before.**

Activity 17.3 Finding errors

This lab is only intended as the briefest of introductions to JavaScript and we don't expect you to do

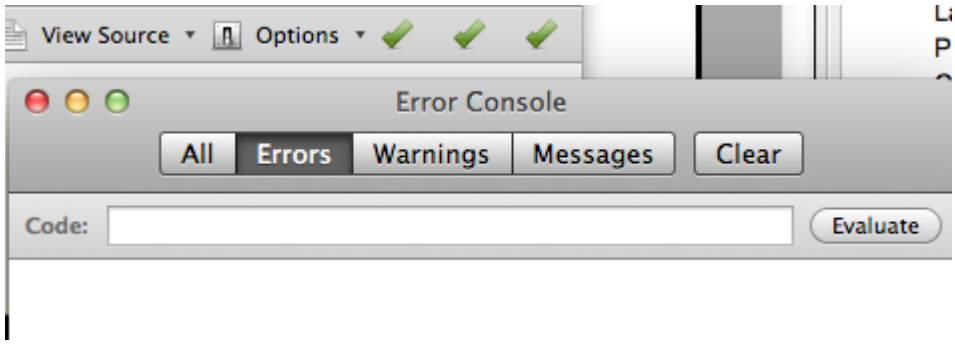
any large scale debugging, however, a quick look at the Error Console won't hurt.

1 **Open JSTest.html in Firefox.**

2 **Notice the three green ticks at the right-hand end of the Web Developers Toolbar.**
These indicate that the HTML, CSS and JavaScript respectively contain no errors.

3 **Click on one of the ticks and the Error Console appears; select the Errors tab.**
The window probably contains lots of errors from pages you have viewed previously. Don't worry about them.

4 **Press the Clear button.**

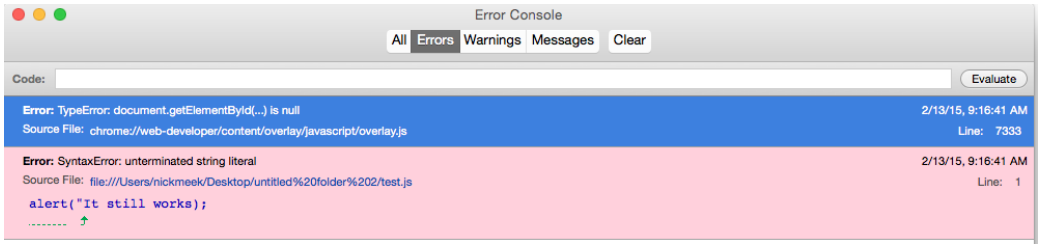


You will now introduce an error to your JavaScript and see what happens.

5 **Introduce an error to your JavaScript by removing one of the quotation marks.**

```
...
alert("It Works);
...
```

6 **Save test.js and reload JSTest.html in Firefox, you should see an error similar to the one below.**
Actually I get two errors for some reason. The first one is nothing to do with my page. I can tell this because the error occurs on line 7333; my JavaScript only has one line. Also the source file (chrome://web-developer/...) is not one I have created. The second error message is my one as can be seen from the "Source File" part of the error message.



Understanding error messages is something that takes practice. In the following activities if your JavaScript doesn't work you should always check the Error Console before calling a demonstrator. Clearing the console and reloading a page often makes it easier to understand which errors are relevant.

7 **Fix the error and re-save the file.**

Activity 17.4 Comments

Just as in HTML and CSS your code should be commented. Comments in JavaScript come in two flavours:

```
// Single line comments


/* Comments that are so long that they need more than one line to record
the fullness of their message */
```

You should put comments to your code as you think appropriate throughout the rest of this lab. Every JavaScript file should contain a comment at the top saying who wrote it, when and what it is for.

Variables

In the next section you will use 'variables" to store some values. You can think of variables as named containers (little pieces of computer memory) in which you can store values (data or results) while your program is running. Often the things stored are simple things like a numeric value, 7 or 22.91 for instance or the thing stored might be some text, "my name". It might even be a boolean value, a boolean value is either 'true' or 'false'. Sometimes the things stored in variables are much more abstract, we won't worry about them. Some variables can hold more than one value. An "array" is an example of such a variable. We will use one later in this lab. Don't be concerned with the details of variables, we are just giving you a taste here of what is possible with JavaScript.

Activity 17.5 A Simple Image Carousel

In this activity you will make a simple image carousel of the sort you might see on TradeMe or to make a home page appear more dynamic. See the short movie in the lab files to see what the end result should look like.

1. **Copy the appropriate lab files to your COMP112 folder.**
   In the lab files you will find a number of landscape images and a left and right arrow image.
2. **In the same folder use Taco to create three new files, carousel.html, carousel.css and carousel.js.**
3. **Add html to carousel.html so that you have a page as shown below.**
   You should use a `<figure>` element to contain each of the images. To get them to display on the same line I added CSS to make the display property of the `<figure>` to be inline.

You might also like to give the `<body>` element a width to stop the layout collapsing when the browser window is made smaller than the combined width of the elements.

4 **Add an id to each of the `<img>` elements so that you can grab them with the JavaScript later.**
I used id="leftArrow", id="picture" and id="rightArrow".

5 **Connect carousel.html to the javascript file to by adding the appropriate `<script>` tag in the `<head>` section of carousel.html..**

6 **Add an alert to the JavaScript file, just to make sure you connected them correctly and ensure that the alert shows when you view carousel.html in a browser.**

```
alert("Success");
```

If that works properly congratulations; if it doesn't work fix the problem. Is the JavaScript file in the correct folder? Have you connected the files correctly. Are there any html or JavaScript warnings or errors?
Doing these simple checks should become part of your normal work practice, it can save hours of frustration later.

7 **When you have the JavaScript properly connected remove the alert from the JavaScript.**
Now that you are sure that the documents are properly connected you can start to build the JavaScript.

8 **Add the following JavaScript to the page:**

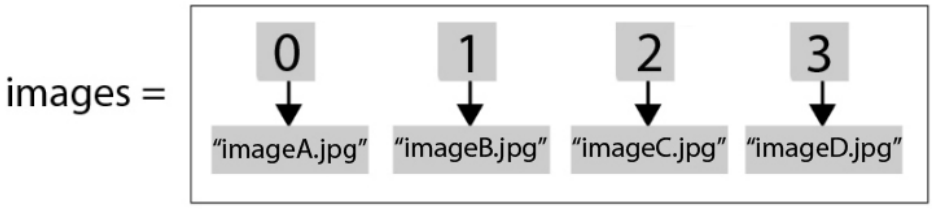
```
if (document.getElementById){
    window.onload = setup;
}
```

The purpose of this code is to check that the browser supports JavaScript (by checking that it can understand the `document.getElementById` statement). If it does then, after the page has finished loading a function called `setup` is called. You will write this function later.

9 **Now you need to add some code to store the names of the image files you want to use. Add the following code at the top of your JavaScript file:**
You will use an array for this. At its simplest an array is a collection of data values stored in numbered (indexed) array elements. The elements are numbered from 0.

```
var images = new Array("imageA.jpg", "imageB.jpg", "imageC.jpg",
"imageD.jpg");
```

You can visualise the array that is created like this:



The individual elements of the array are accessed via their index number. Thus images[0] = "imageA.jpg" and images[3] = "imageD.jpg".

10 **You will also need a variable to keep track of the current index number. Add the following code underneath the previous line:**

```
var index = 0;
```

11 Now you should add the setup function that will be called after the page loads. Add the following code after the variable declarations but before the if() statement.

```
function setup(){
    var leftArrow = document.getElementById("leftArrow");
    var rightArrow = document.getElementById("rightArrow");
    leftArrow.onclick = goLeft;
    rightArrow.onclick = goRight;
}
```

The first line declares the function, the function is called setup.
The second and third lines create a couple of variables which reference the HTML elements with the ids "leftArrow" and "rightArrow".
The fourth and fifth lines attach event handlers to those elements. These event handlers (called goLeft and goRight) will be called when the relevant elements are clicked.

Now the last thing to do is create the two event handling functions.
To start with add two function stubs as shown below. The reason you do this is, as previously, to ensure everything is connected correctly.

12 Add the code for the goRight and goLeft function stubs above the setup function.

```
 function goRight(){
    alert("goRight function called");
}


 function goLeft(){
    alert("goLeft function called");
}
```

13 Save carousel.js and open carousel.html in a browser. Try clicking on the arrows and ensure the appropriate alert message is displayed.

14 If that worked as it should then remove the alert code from the goRight function and add the following code:

```
function goRight(){
    var img = document.getElementById("picture");
    index = index + 1;
    if(index == images.length){
        index = 0;
    }
    img.src = images[index];
```

```
}
```

Here is what that code does:
Line two creates a variable called "img" and stores a reference to the HTML element with the id "picture" in it.
Line three adds 1 to the value of the index variable.
The "if" statement checks if the current value of index will fall off the end of the array. `images.length` will return the number of elements in the array, in this case 4. However remember that array indexes start at 0 at so (referring to the earlier diagram) the last element in the array is at position 3. So, if index equals images.length then we have reached the end of the array and need to reset the index back to the first element, 0.
Finally the last line sets the `src` property of the element referred to by the variable `img` to what ever is is in the array at position `index`.

15 Save your file and open carousel.html in a browser. Try clicking on the right button, the images should change.
Click on it lots of times to make sure that after the last image is displayed the next image goes back to the first image.

Now let's think about the other button. The basic premise here is exactly the same as the right button, except that we want to go the other way through the array. So, the code for the `goLeft` function will be very similar except for the following:

1. You need to decrease `index` each time the function is called rather than increasing it.
2. Rather than falling off the high end of the array you might fall off the low end. That is if `index` becomes negative you will try to access array elements that don't exist, so you need to guard against that.

16 Add code to the function goLeft.
The code for this is the bottom of the page but you might like to see if you can work it out for yourself first. It really is very similar to the goRight function.

Lab Completion (1%)

Have a demonstrator check your work now.

- ✔ Carousel.html and carousel.css validate.
- ✔ Carousel works in both directions.

Demonstrator: _____
Date: _____

```
function goLeft(){
    var img = document.getElementById("picture");
    index = index - 1;
    if(index < 0){
        index = images.length - 1;
    }
    img.src = images[index];
}
```

JAVASCRIPT (1%)

Lab 18 Accessibility Challenge (2%)

Reference

 Lecture 18

 http://www.w3.org/WAI/

 https://www.w3.org/TR/WCAG20/

 http://www.vischeck.com/

Introduction

In this lab you will learn more about accessibility issues associated with delivering content via the web. Firstly you will try to navigate and 'read' a web site as if you had a serious sight impairment. To do this you will use VoiceOver, the Mac OSX built-in screen reader. You will then add some accessibility aids to the site and again navigate the site using VoiceOver

> ***The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.***
>
> *-- Tim Berners-Lee, W3C Director and inventor of the World Wide Web*

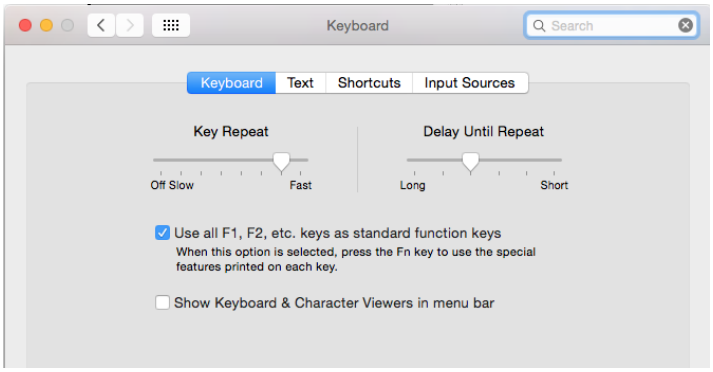
Activity 18.1 VoiceOver Basics

VoiceOver is the Mac OS built-in screen reader application. In this activity you will use VoiceOver to experience visiting a web site as a blind user.

To see someone using a screen reader view the movie ScreenReaderAccessibility.mov in the coursework/movies directory. This is a rather excellent presentation by someone who uses a screenreader in their everyday work.

The default setup for Mac OSX in this lab is for the Function Keys (top row of the keyboard) to act in special ways, to activate Expose and other Mac OS features. We will reset them to act as standard Function keys.

1 **From System Preferences > Keyboard ensure the function keys will act as standard by checking the box.**



2 **Quit all running applications.**

3 **Put on the headphones.**

4 **Start VoiceOver, Cmd+F5.**
VoiceOver will start and then describe the current active window. Click OK.

5 **Start the VoiceOver Quick Start Tutorial by pushing <control>+<option>+<Cmd> +F8 and go through the tutorial.**

There are some laminated Quick Reference guides available in the lab.

Some Useful VoiceOver Commands

Note: VO = <control> + <option>

- Screen Curtain : VO<shift> - F11

Basic

- Turn VoiceOver On and Off : Command-F5
- Start keyboard help : VO-k
- Close a menu, stop an action etc. : VO-Esc.
- Read all text : VO-a
- Read all the contents of a window : VO<shift>-W

Actions

- Read everything visible in the window : VO<shift>-W
- Orientation
- Application summary : VO-F1
- Application Chooser : VO-F1 (twice)
- Window summary : VO-F2
- Describe he selected item : VO-F6

Navigation

- Interact with an item : VO<shift>-Down Arrow
- Stop interacting with an item : VO<shift>-Up Arrow
- Move up : VO-Up Arrow
- Move down : VO-Down Arrow
- Move to previous : VO-Left Arrow
- Move to next : VO-Right Arrow
- List the links on a page : VO-u
- Hear a link's URL : VO-<shift>U

Activity 18.2 Browsing with VoiceOver and Screen Curtain

1 **Start Safari; VoiceOver works best with Safari.**

2 **Start VoiceOver (Cmd+F5).**

3 **Operate the Screen Curtain by pressing <control>+<option>+<shift>+F11.**
You can turn the Screen Curtain off by pressing the same keys.

4 **Untangle your fingers.**

Keep the Screen Curtain closed as much as possible for this activity, only open it when you become hopelessly lost and then close it again as soon as you locate yourself..

5 **Navigate to the W3C Homepage at http://www.w3.org/ and explore the page a little.**

6 **Navigate to www.cs.otago.ac.nz/comp112/accessibility/page1.html.**
This page was written with complete disregard for accessibility concerns.

7 **Using just VoiceOver (Screen Curtain closed) navigate the site and answer the following questions... if you can:**

1 **What is the site's name?** _____

2 **What is the site about?** _____

3 **Is the navigation easy to find?** _____

4 **Is the navigation easy to use?** _____

5 **Are the pictures useful?** _____

6 **What is the table about_** _____

7 **Is the form easy to use, why?** _____

Checkpoint

Have a demonstrator check your work now.

✔ Activity 18.2: Use VoiceOver with the screen curtain closed.

Demonstrator:_____

Activity 18.3 Install the WAVE toolbar

There are a number of tools that can help in identifying accessibility issues in web pages. You can either use http://wave.webaim.org/ or you can download the Chrome extension. In this lab we use the Chrome WAVE toolbar.

1. **Open Chrome and navigate to http://wave.webaim.org/extension/ and install the WAVE Evaluation Tool for Chrome.**
   The WAVE toolbar has a variety of tools that will assist you in checking a web site's accessibility.
2. **Restart Chrome and navigate to www.otago.ac.nz.**
3. **Click on each of the WAVE functions in turn and note the results.**
   The extension really does provide a lot of useful information. Click on all the tabs and identify the errors and warning on the page.
4. **Copy the lab files to your home directory.**
   This directory contains all the files that make the site you looked at earlier.
5. **Upload the directory to your account on CSnet using YummyFTP.**
6. **Using Chrome navigate to your page1.html and then click on the WAVE icon. There are lots of errors.**

Activity 18.4 Make a Site Pass WCAG 2

In this activity you will modify page1.html to make it more accessible and pass all of the WCAG Level A and Level AA success criteria. These checks are not as clear as the checks for valid HTML ones are and some items can't be checked by a validator; is the contrast ok?, is the alt text sufficiently descriptive?, etc.

See http://www.w3.org/TR/WAI-WEBCONTENT/full-checklist.html for a full list of checkpoints and https://www.w3.org/WAI/WCAG20/quickref/for a customisable list of criteria.

1. **Visit https://www.w3.org/WAI/WCAG20/quickref/ in a browser. This page has all the WCAG guidelines. There are a lot. You might find it helpful to set the filters (on the left) so that it only shows for Level A and Level AA.**
   Many on the required criteria don't apply to the page you are checking but you should read through the list of criteria and pick out those that you think apply.
2. **Remember to use the WAVE extension to help you.**
3. **Firstly ensure that the page is valid HTML.**
4. **Now work through the Level A and Level AA success criteria (that are relevant to this page) one-by-one.**
   Generally this will mean going back and applying the accessibility techniques that were covered in previous labs. But as a starter consider the following:
   - The page has organisational issues.
     The heading seem to be use rather haphazardly and the navigation appears very late in the document flow.
   - There may be contrast problems.
   - Form accessibility.
   - Table accessibility.
   - Image accessibility.

Lab Completion (2%)

Have a demonstrator check your work now.

- ✔ Passes WCAG Level A and AA criteria.
- ✔ Passes demonstrator/manual check.

Demonstrator: _____
Date: _____

ACCESSIBILITY CHALLENGE (2%)

Lab 19 Graphics I (1%)

Reference

 Lecture 19/20

 http://webstyleguide.com/wsg3/11-graphics/5-web-graphics-formats.html

 http://stackoverflow.com/questions/392635/website-image-formats-choosing-the-right-format-for-the-right-task

Check list

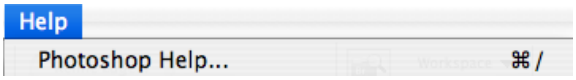
Ensure that you are confident doing the following:

- ✔ Using basic tools in Photoshop.
- ✔ Using the palettes especially the History and Layers palettes.

Introduction

In the Graphics series of labs you will be introduced to Adobe Photoshop CS6; the premier and industry standard digital image manipulation application.

Important: These labs only introduce you to some basic aspects of Photoshop, it has many features and options not presented here. You should become familiar with the Help feature and refer to it frequently. There are also many excellent on-line sources of information, tutorials and add-ons.



Activity 19.1 Introduction to Photoshop CS6

In this lab and some subsequent labs you will watch Desktop recordings which demonstrate the activities you are to undertake. The movies (.mov files) are located in the Coursework>COMP112>Movies directory

You can watch the movies by double-clicking on them in their directory, you should not copy them to your Home directory. If you do download a movie by accident can you please put it in the Trash and empty the Trash, this stops the server getting filled up with lots of copies of the same (large) file.

The .mov files are often quite large so please don't download them.

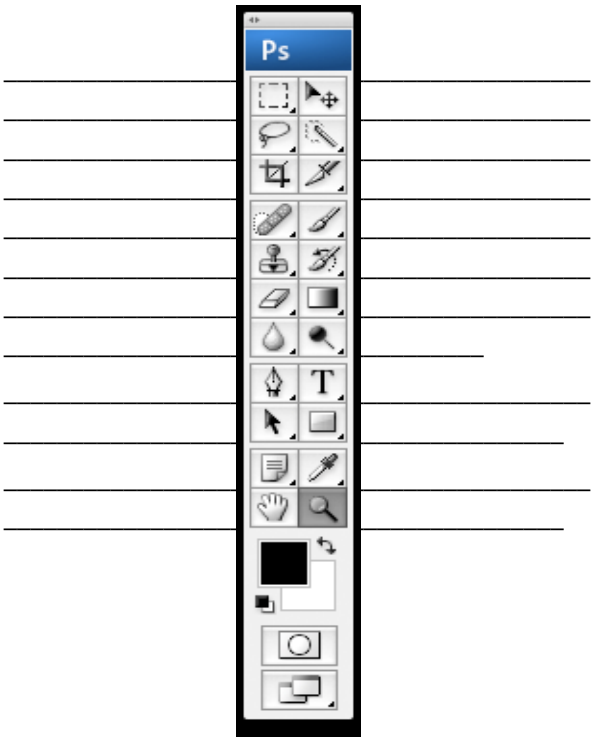
The movies have a soundtrack, use the headphones supplied (or use your own if you prefer).

Watch the movie Photoshop_Introduction.mov.

1 Open Adobe Photoshop CS6.

2 Examine the Toolbox.

3 Fill in the names of the tools.



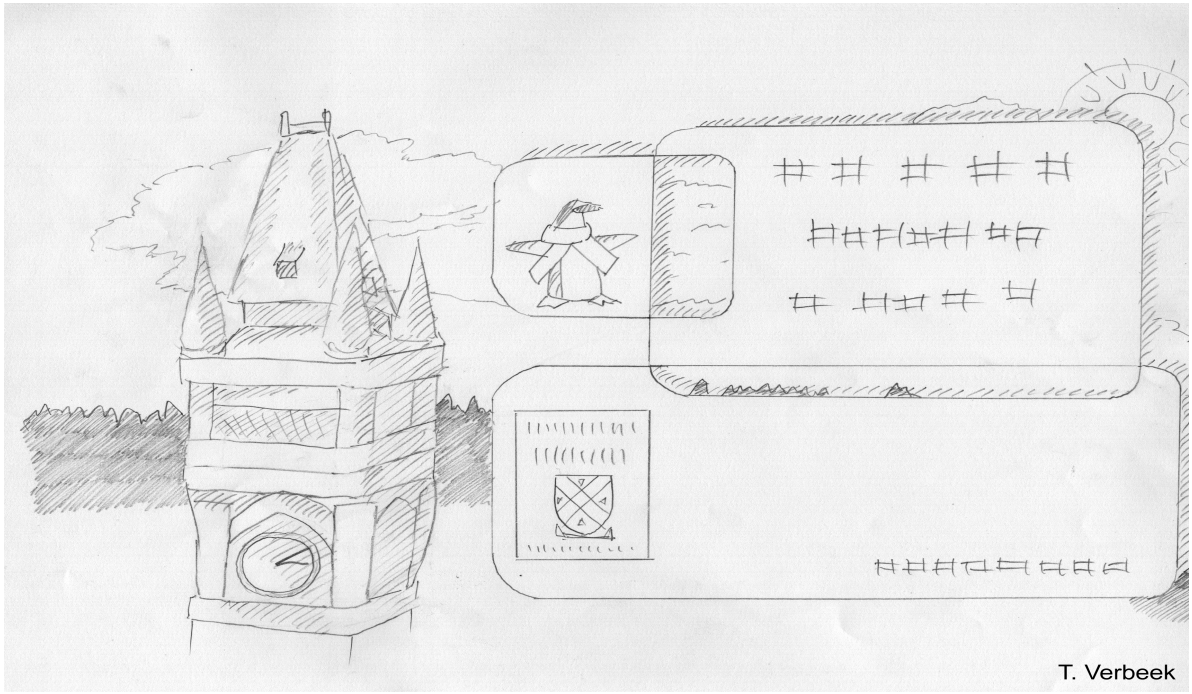
Activity 19.2 Task Outline

Let us suppose that **an image is required for use on web pages** as part of an on-line advertising drive to attract more students to Computer Science. A concept drawing has been created, you just need to make the image.

In this set of activities you will make an image similar to the one pictured in the concept drawing below. Making this image will help you become familiar with some basic tools and processes such as using layers, adding text, retouching etc., as well as giving you time to familiarise yourself with the Photoshop environment.

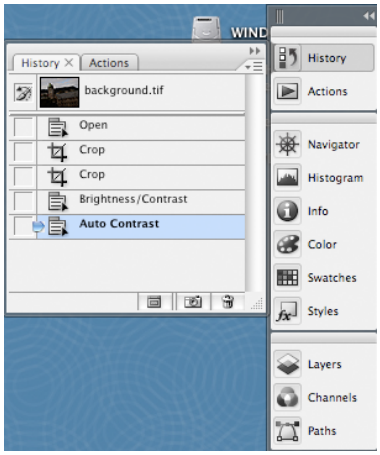
The finished size of the image will be 500 pixels by 200 pixels and of an appropriate quality, (file) size and format.

Concept Drawing



The History Palette

Probably the single most useful tool that Photoshop has is the History palette. You should get used to using it a lot. The History palette contains a list of your previous 20 actions so you can easily undo up to the last 20 things you did. Never be afraid to experiment with Photoshop, the History palette will *almost* always get you out of trouble.



Locate the History **palette.**
If you haven't done anything yet other than open background.tif there won't be any actions recorded apart from 'Open'.

Leave the History **palette toggled open and keep an eye on it as you complete the following activities.**

If you make a mistake or aren't completely satisfied with the result of an action go back a few steps by clicking on the action you wish to go back to and try it again.

Activity 19.3 Rotation, Cropping & Selection

Watch the movie Photoshop_Crop_Select.mov.

1. **Copy the relevant files from coursework to your home directory.**
2. **Select** File > Open**, navigate to your copy of the lab directory and open** background.tif**.**
   This file is the background of the image you are creating, you will firstly work on it, then add other layers to it to build up the final image.
3. **Straighten the image.**

The image is obviously crooked and this needs correcting.
Your image should be cropped to show the clock tower close to the left-hand edge. This is important so that other elements have space to fit properly. See the concept drawing.

10 **Unlock the background layer.**

11 **Select the sky using the magic wand tool and delete it.**

12 **Save your file as** activity1.psd**.**
.psd is Photoshop's native format, it preserves layer information and is lossless; files are large. Use this format for files you may wish to edit again, e.g. original compositions.

Activity 19.4 Brightness and Contrast

Watch the movie Photoshop_Light.mov.

1 **Adjust the brightness and contrast of** activity1.psd**.**

2 **Save the** activity1.psd **file.**

Activity 19.5 Retouching I

Images you wish to use may have small blemishes such as red-eye, scanning artefacts or the composition may be less than perfect, someone has a power pole apparently growing out of their head for instance. These issues and many others can be rectified by retouching the image.

There are two issues with the image we have been given, the large tree behind the right-hand spire and the rust stains on the roof.

Watch the movie Photoshop_Retouch_I.mov.

1 **Remove the tree and replace the spire.**

2 **Save your file as** activity1.psd**.**

Activity 19.6 Retouching II

Watch the movie Photoshop_Retouch_II.mov.

The image **shows the rather ugly rust staining on the roof of the clock tower building.** In this next exercise we will remove this staining and make it appear as if the roof is unblemished.

1 **Remove the rust stain until on the roof of the clock tower.**
Use a combination of replace colour and the Clone Stamp tool.

2 **Save the file.**

Activity 19.7 Resize the Image

Watch the movie Photoshop_ImageSize.mov.

1 **Open** activity1.psd **and save it as** activity1Sml.psd**.**
Now we have a copy of activity1.psd that we can resize and further work on.

2 **Resize** activity1Sml.psd **to 500px wide and 200px high.**

3 **Save the image as** activity1Sml.psd.

Activity 19.8 Select and Resize

Watch the movie Photoshop_MoreSelect.mov.

1 **Open unilogo.tif.**

This image is the wrong size and has a blue background which we wish to remove. The final logo needs to be 42 pixels wide and proportionately high.

2 **Remove the blue background from the image, resize it and add it to your** activity1Sml.psd **file.**

3 **Open** blueSky.jpg **and add it as a new layer to** activity1Sml.psd**.**

4 **Save your file.**

This work continues in the next lab. Make sure you save all your files carefully.

Lab Completion 1%

Demonstrator:_____

GRAPHICS I (1%)

Lab 20 Graphics II (1%)

Check list

Ensure that you are confident doing the following:

- ✔ Using the measuring tools
- ✔ Shapes and Opacity
- ✔ Adding text

Introduction

In this lab you will finish creating the image you started on last lab.

Activity 20.1 More Selection

Your next task is to tidy up and resize the Computer Science Penguin logo. The penguin logo is supplied as a .jpg file, which is not ideal. This format does not support transparency and consequently the image has a white background. The penguin needs to be removed from its background, resized, and then imported as a new layer into activity1.psd.

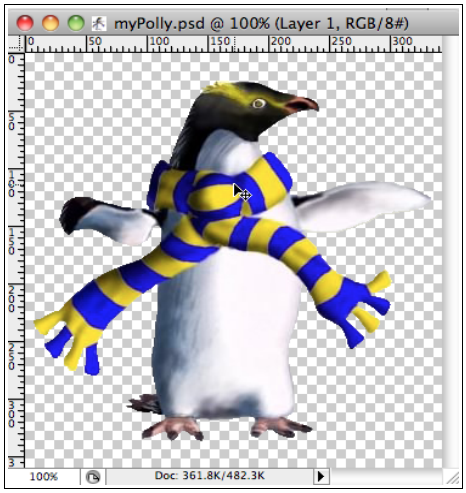
Watch the movie Photoshop_MoreSelection.mov.

1 **Open** HalfDone.psd**.**
This file has already had a lot of the background removed for you.

2 **Add a new layer and give it a dark background**

3 **Remove the remaining background from** HalfDone.psd **and save the file as** polly.psd.
Remember to zoom in close and use a small (1px – 3px) eraser for the detail work and to view Actual Pixels to see the overall effect.

4 **Check** polly.psd **one last time, this will be the 'reference copy' of the image and needs to be as good as you can make it. You will not alter this image any more. When you are completely satisfied save** polly.psd **one last time.**



5 **Add Polly to** activity1Sml.psd **as a new layer.**

6 **Name the new layer** Polly**.**

7 **Re-size Polly to the correct size, 42px wide and proportionately tall.**

8 **Save** activity1Sml.psd**.**

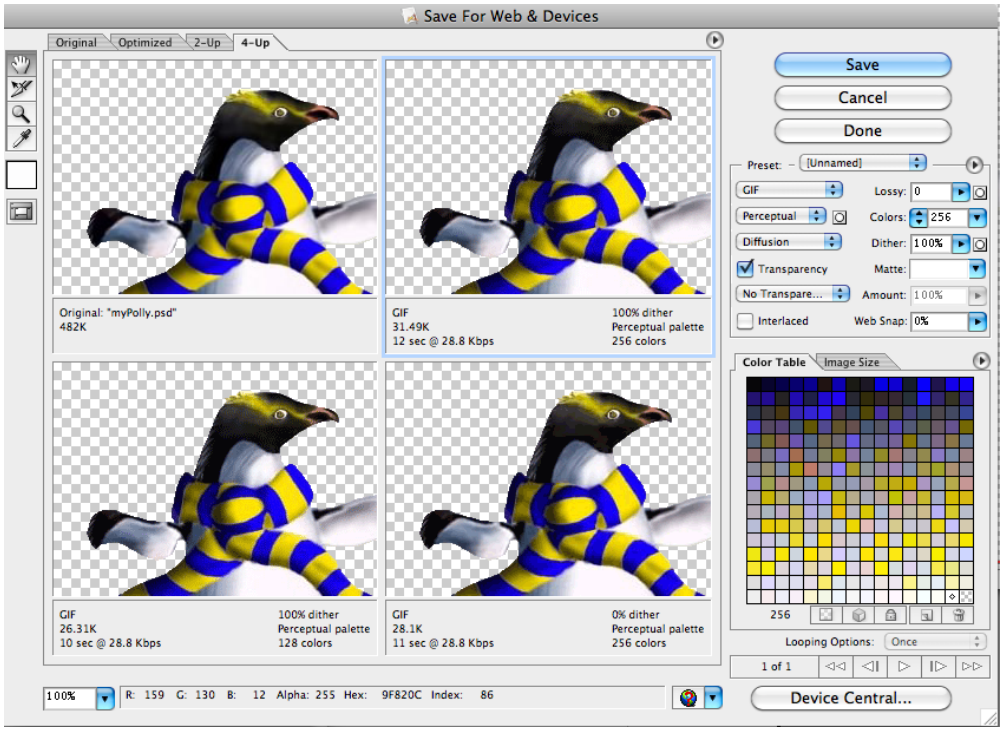
Activity 20.2 Saving as GIF

At this point we might like also to make a version of Polly that has a transparent background for use in other web environments. Only two commonly used formats support transparency, GIF and PNG-8. We will save the image as a GIF but the process for saving as a .PNG is very similar.

1 **Open** polly.psd**.**
Remember: the colour model for polly.psd is 8-bit per channel RGB (24-bit colour), GIF uses an indexed palette of up to 256 colours.

2 **Choose** File > Save for Web & Devices**.**

3 **Now click on the** 4-Up **tab.**
Photoshop will display four different versions of Polly, one PSD version for reference and three other image versions.
Note, you can change the format as well as other settings for each of the image versions independently.
Note also the estimated file size and download time for each image. The trick now is to choose the correct balance of quality and image size.



4 **Experiment with the different GIF settings, dither, number of colours and note their effect on the preview image and the file size.**

5 **Select one of the** GIF **versions by clicking in its panel and change its format to** PNG-8**.**

6 **Vary the colours and dither settings and note the file size/image quality results.**

7 **Return the format to** GIF.

8 **Examine each of the GIF images carefully and critically, use the Move and Magnify tools, modify the attributes. Which combination of settings do you think should be used in this case? That is, which combination of settings provide the best quality/file size compromise?**

Number of Colours: _____

Dither / Amount: _____

9 **When you are happy with your choice ensure that version is selected by clicking its panel and click Save.**

10 **Fill in the dialogue box and save the image as** polly.gif**.**

11 **Preview the image by opening it in a browser.**

Activity 20.3 Measuring

Before doing any measuring in Photoshop it is important to set the zero point, the place from which measurements are taken. Its default location is the top left-hand corner of an image but it can be moved, you may wish to measure the distance between two parts of an image for example, or you may want to know something's distance from the centre.

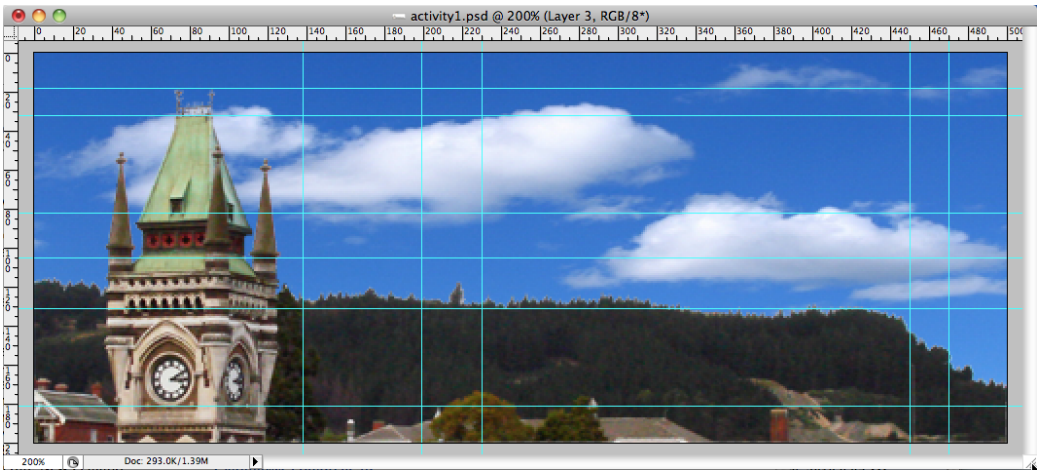
Watch the movie Photoshop_Measuring.mov**.**

1 **Open** activity1Sml.psd**.**

2 **Place horizontal guides at 18, 32, 82, 105,131 and 181 pixels.**

3 **Place vertical guides at 138, 199, 230, 450 and 470 pixels.**

4 **Lock the guides.**



Activity 20.4 Working with Shapes and Opacity.

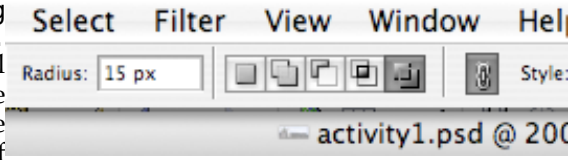
Watch the movie Photoshop_shapes_and_opacity.mov.

1 **Lock the** Unilogo **layer.**

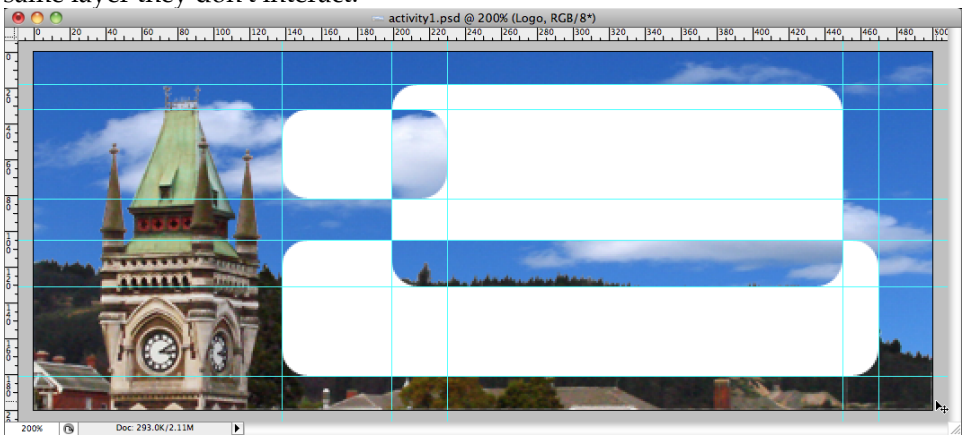
2 **Lock the** Polly **layer also.**
Locking the layers will prevent unintentional editing of these layers.

3 **Add a white rectangle from (200, 18) to (450, 131).**

4 **Ensure** Exclude overlapping shape areas **button is selected.**
This option isn't visible until there is already one shape on the layer; you can't exclude overlapping parts of a shape if there is no shape to overlap.



5 **Drag out two more rectangles using the guides, as pictured below.**
This effect only works if the shapes are on the same layer. If they are not on the same layer they don't interact.



6 **When you are satisfied save your file.**
In the image above the shape layer is on top of the unilogo and Polly, obscuring them.

7 **Move the** Shape 1 **layer below** Polly **and** unilogo**.**

8 **Reduce the** Opacity **of the** Shape 1 **layer to** 70%**.**

9 **Add a** Drop Shadow **to** Shape 1 **with an** opacity **of** 75%**, a** light angle **of** -155**, a** distance **of** 8 **and a** size **of** 2**.**

10 **Lock** Shape 1**.**

11

12 **Unlock the** Polly **and** unilogo **Layers.**

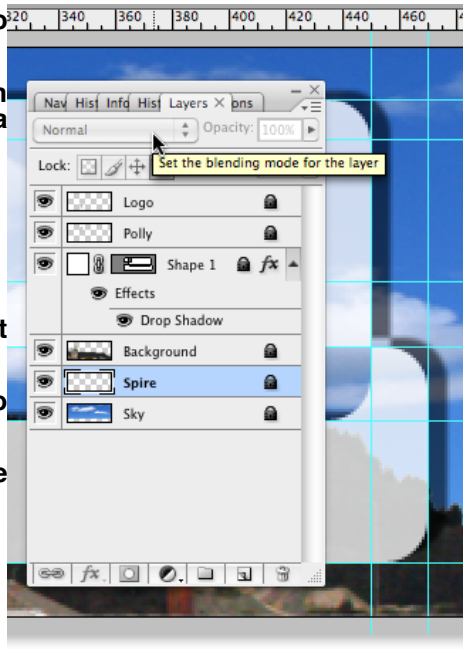
13 **Move them within the guides as shown at right.**

14 **Select the** unilogo **layer and then link it to the** Polly **layer.**

15 **Align the horizontal centres of the penguin and the University's logo.**

16 **Lock all layers.**

17 **Save** activity1Sml.psd**.**



Activity 20.5 Adding Text

You will now add the text. You will make the two phrases on two layers so they can be positioned individually.

Watch the movie Photoshop_Text.mov**.**

1 **Add the text:**
It's Time For A Change.
Computer Science
@ Otago University
The Key To Your Future.

The font is Minion Pro**, the** size **is** 16pt **and the style is** Semibold**.**

2 **Add the URL text (the size is** 18pt**).**

3 **Lock both Text layers.**

4 **Save your file as activity1Sml.psd.Add a Lens Flare**

We now need to add a Lens Flare effect, this will look like a bright sun. Before we add this effect we need to Flatten your multi-layered image. Flattening your image discards all the layer information and renders your image as a single bitmap. We do not of course want to do this to out original image, just in case we want to do other things with it later.

1 **Open activity1Sml.psd and immediately save it as lensFlare.psd.**
So now we are working on a copy of activity1Sml.psd.

2 **Select Layer> Flatten Image.**
Note in the Layers palette there is now only 1 layer.

3 **Select Filter> Render > Lens Flare.**

Experiment with the settings until you find something suitable and then click OK.

Activity 20.6 Save your image for use on the web

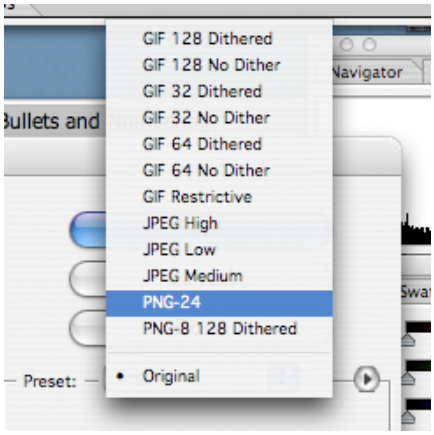
Currently your file is saved as a .psd file. This is a very useful format to keep the file in for editing in Photoshop. It keeps all layer information as well as being lossless. The .psd format however is not suitable for publishing images in, the files are large and are not readable other than by Photoshop and a few other specialist applications.

1 **Select** File > Save for Web**.**
Remember that this image will be published on a web page so not only is quality important but keeping the file size small is a factor. As activity1 is a photographic or continuous tone image format that allows for a wide range of colours is indicated, either JPG or PNG 24 offer compressed high colour images.

2 **Select the** 2-up **tab.**
The 2-up view is very useful for comparing the visual quality and file sizes of an image using different formats, compression settings and colour palettes.

3 **Select the upper image by clicking on it once, it is outlined in light blue to indicate it is the active pane.**

4 **Pull down the** Presets **menu next to the pane and select** PNG 24**.**
Note the file size is displayed under the image pane.



5 **Select the lower pane and select** JPEG Low**.**
Notice the compression artefacts especially around the edge of text and on Polly.

6 **Experiment with other formats and settings.**
It is still worth looking at the effects of GIF and PNG-8 formats on photographic images, the results can be surprisingly good.

7 **Select a format/quality that you think provides the best balance of quality and file size.**
Hint: My file ended up at 31Kb whilst still retaining sufficiently good quality.

8 **Save you file in your chosen format.**

My chosen file format/setting: _____
My file size: _____

Lab Completion (1%)

Have a demonstrator check your work now.

✔ Activity completed and saved in appropriate format.

Demonstrator: _____
Date: _____

GRAPHICS II (1%)

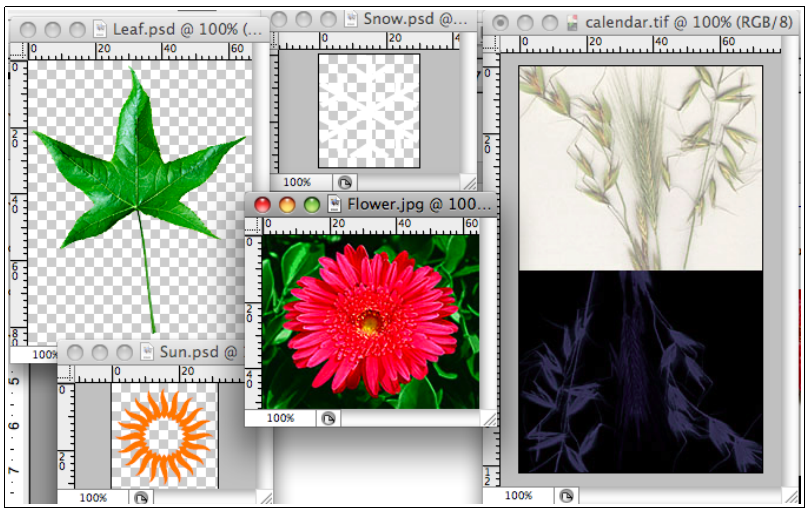
Lab 21 Graphics III (1%)

Introduction

In this set of labs we have only been able to scratch the surface of what can be achieved with Photoshop. The Photoshop Help is quite useful and there are a range of other tips and tutorials under the **Help > How To..** links. There are also many good resources on the WWW that contain step-by-step instructions on how to achieve particular effects.

Activity 21.1 Making a Composite Image II

In this first activity you will make an image similar to that below.



There are no step-by-step instructions or instructional videos for this lab but here are some tips:

1. **Get a copy of the lab files and open the file Flower.jpg.**
2. **Separate it from it's background by using the Magic Wand tool.**
   - Use a tolerance of about 120.
   - Click on the flower and then shift-click to select additional areas.
   - When the whole flower is selected copy it and paste it into a new Photoshop file.
   - Save the new file as Flower1.psd.
3. **Open the other files used in this image, Seasons.psd (this is the background image), Sun.psd, Snow.psd and Leaf.psd.**
4. **Naming the layers with sensible names will make your task much easier.**
5. **The sun should be at least 50% transparent.**
6. **Rotate the leaf using Edit>Free Transform.**
7. **Resize the snowflake by using Edit>Free Transform. Emboss it using Filter>Stylise>Emboss.**
8. **Add a shadow to the flower by duplicating the layer (Layer>Duplicate layer). Name the layer Flower Shadow and work on that layer as follows:**
   - Make sure the foreground colour is set to black.
   - Select Edit>Fill. Select the Preserve Transparency option to apply the fill only where it is painted. Click OK. The flower shadow is now black.
   - Make the opacity of the shadow layer ~60%.
   - Select Filter>Blur>Gaussian Blur to blur the shadow.
   - Reposition the flower layer so that it looks like a shadow.
   - Link the flower layer and its shadow.
9. **Add the text, the font is 80pt.**
10. **Add the fading effect over the text by doing the following:**
    - Select the type layer.

- Select Layer>Layer Mask>Reveal All.
- Select the gradient tool.
- Drag the tool across the text. You may wish to experiment with the options in the option bar.

11 Save your file as Calendar.psd.

Checkpoint

Have a demonstrator check your work now

✔ Calendar.psd

Demonstrator:_____

Activity 21.2 Making a Composite Image III

1 Locate and open the files tour.psd (the background image), CD.psd and Horn.psd.
You will use these files to make an image similar to that below (a finished version is available as Concert.tif). As in the activity above step-by-step instructions will not be given.

- Use the Marquee tool to extract the CD from its background. Use the Move tool to move the selected CD to tour.psd. Use the Free Transform tool to resize and rotate it. Note that the CD is semi-transparent.
- Use the Magic Wand tool to select the horn from its background. You will need to experiment with the tolerance in the Option bar. Using Select>Inverse may be useful. You might also like to see the effect of un-checking the Contiguous checkbox in the Option bar. Note the scratch on the bell of the horn, remove it using the Clone Stamp Tool.

- Modify the background layer using either Filter>Distort>Twirl or Filter>Distort>Wave.
- Add the text as shown. Use Layer>Layer Style to apply effects to the text layer.
- Ensure all the layers are visible, tour.psd includes not only the background image but also a layer called notes. Drag the notes so they are just above the horn and make the 50% transparent.
- Add a gradient to the background layer. Set the opacity to 30% and the mode to Multiply.
- Flatten your image and add a vignette. To do this use the Rectangular Marquee tool to select most of the image (you will need to leave a thin border unselected) and then choose Vignette (selection) from the Actions palette. You will need to click the Play button at the bottom of the palette to apply the action.

2 Flatten your image again and save it as Concert.psd.

Checkpoint

Have a demonstrator check your work now

✔ Concert.psd

Demonstrator:_____

Lab Completion (1%)

Have a demonstrator check your work now

✔ Calendar.psd.

✔ Concert.psd.

Demonstrator: _____
Date: _____

GRAPHICS III (1%)

Lab 22 Project Submission

Introduction

You should use this lab time to submit your project. Remember to test everything after you have uploaded it to the server. That means test every link on every page, ensure everything validates and that any images load and display correctly.

Remember that you must not make any alterations whatsoever after the project submission date or your project will be marked as late.

Lab 23 Site Update (1%)

Reference

 http://www.w3schools.com/css/css3_intro.asp

 http://www.w3schools.com/html/html5_intro.asp

Introduction

Often web developers don't get to make a website from start to finish. Sometimes they are modifying or updating pages that someone else has made. In this lab you will finish a half finished site.

Overview

Today your task is to complete a small website that has been only partially written. The original developer got as far as marking up the content using valid HTML 4.01 Strict and started on the CSS but then abandoned the project and ran off with a sailor to Barbados! You can find what files exist in the coursework directory. In particular note the design pictures, this is what your finished site should look like.

The site specification has also been changed, the site should now be marked up as validated HTML5.

The pages should display 'properly' in

Mac OSX:

- Firefox
- Safari

WindowsXP:

- Firefox
- IE8

In this case 'properly' will mean usably: not all the features (especially CSS3) will work in all of the browsers specified and so you will need fall-back positions which allow your styling to '*degrade gracefully*'.

Activity 23.1 Retrieve the lab files

1. **Copy the relevant lab files from /coursework.**
2. **Open and examine the three design documents.**
   Note all the detail on the documents including colour codes, widths etc.

Stop! Before you start the next section read through all the following instructions.

1. **View html5.html with your favourite HTML editor.**
2. **Read through the html.**
   You don't need to read every word of the content. What you are trying to do is gain a good understanding of two things:
   - The gross structure of the page as described by the big block-level elements.

   In particular take note of how the divs are slicing the content up into chunks.
   - How the content is described, the fine grained semantic elements.

   Do you know what all the elements are? Do you see the pattern of how and why they are being used?

   When you think you understand how the page works proceed to the next step.
3. **Open html5.html using Firefox.**
4. **Use the tools on the Web Developer Toolbar (e.g. Outline → Current Element) to further explore the structure of the page.**
   You might find it useful to have the style sheet source open at the same time. This time you need to concentrate on seeing how the current styling works.
   You might like to make small changes to the style sheet to see if the layout reacts as you think it will.
   The point of this is to properly understand how the page styling works.
5. **Alternatively or additionally you might like to view html5.html with Safari and then using the Develop → Web Inspector.**
   The Web Inspector provides a *lot* more information than Firefox's Web Developer Toolbar.
6. **Do the same for css_3.html.**

7 **When you are sure you are sufficiently familiar with the pages you can proceed to the next step: updating the HTML markup to HTML5.**

Activity 23.2 Mark-up using HTML5

1 **Re-markup the pages as valid HTML5.**
To start with you need to modify the Doctype declaration so that it identifies the page as being HTML5 (and not HTML 4.01 for instance).

2 **Change the doctype declarations of html5.html and css_3.html to the HTML5 style.**

3 **Re-markup the content of html5.html and css_3.html as valid HTML5.**
You may need to use the following HTML5 tags:
- <header>
- <footer>
- <section>
- <article>
- <aside>
- <nav>

As well as the other more familiar tags like <p>, <ul> etc.
See http://www.w3schools.com/html5/html5_new_elements.asp for a full list and brief description of their usage.
Changing element names will of course destroy the formatting as the CSS selectors will be different. You might like to change the selector names in style.css as you go.

4 **Ensure your HTML 'validates' as HTML5 and that your have used the best possible elements to describe the content.**

Checkpoint

Demonstrator:_____

Activity 23.3 Styling the page with CSS3

Now that you have valid marked-up content you can style it using CSS3. Your finished page should look exactly like (well very very similar to) those in the design pictures supplied along with the lab files.

1 **Modify style.css so that html5.html and css_3.html conform to the design specifications.**
Check the design pictures.

@font-face

One of the most exciting features of CSS 3 is the widespread implementation of @font-face. No longer are designers tied to the boring web-safe fonts. Now you are able to specify any font you want, as long as you can supply the particular font file. The form of the selector is:

```
@font-face{
  font-family: myFunkyFont;
  src: url(/path/to/font/FunkyFont .ttf) format("truetype");
```

```
}
```

Then you can use your new fontName in your CSS just like you would usually.

```
h1{
  font-family: myFunkyFont, verdana, sans-serif;
}
```

1 **Open style.css and add the @font-face selector.**
Use the fonts specified in the design.

@font-face and IE

As ever though it isn't quite that easy. IE uses the proprietary .eot font file type while all other browsers don't. The other browsers support a variety of different formats but all other browsers do support .otf font files. Getting the right browsers to see the right font file is entirely straightforward. First you need a src that points to the IE friendly font file and then another url for a font file that all other browsers understand.

```
@font-face{
  font-family: fontName;
  src: url(./path/to/font/FontFile.eot), /*for IE*/
  url(./path/to/font/FontFile.ttf)format("truetype");
}
```

1 **Modify your CSS as above and check that it works using cs-ts to check using IE.**
The .eot versions of the fonts are in the coursefiles directory.

See http://sixrevisions.com/css/font-face-guide/ or http://randsco.com/index.php/2009/07/04/p680 or Google for more information on using @font-face.

Gradients

Prior to CSS3 if you wanted a gradient background you had to use an image in some fashion. CSS3 however has the gradient property which means the gradients are rendered by the browsers and therefore don't need to be included as images. This property is only an experimental feature in Firefox and Safari. IE6/7/8, Opera, earlier Safari, and Firefox 3 cannot render CSS 3 gradients. In addition there are currently different ways of specifying gradients for different browsers <sigh>.

See one of the following (or Google your own) for more information on using gradients:

http://css-tricks.com/examples/CSS3Gradient/

http://gradients.glrzad.com/

Don't worry about trying to get gradients working in IE for your pages, just specify a fall-back background colour. The gradient should work in all other major browsers though on both Mac and Windows.

Because gradients are not properly implemented yet they will generate errors when you validate your CSS. Errors generated by the use of gradients are acceptable.

Drop Shadows

Box shadows could hardly be easier, Google them.

Rounded Corners

Giving elements rounded corners comes in two basic flavors, CSS 2.x and CSS 3. You will only have to use the CSS 2.x method if you are trying to get rounded corners on IE8 or earlier or early versions of other browsers. In this lab we will use the CSS 3 method and live with the consequence of non-validating CSS.

Rounded corners in CSS3 could, again, hardly be easier.

```
selector{border-radius: <number>px; }
```

e.g.

```
{ p: border-radius: 10px; }
```

See http://www.css3.info/preview/rounded-border/ or Google for more information.

Lab Completion (1%)

Have a demonstrator check your work now.

- ✔ Semantically rich and valid HTML5
- ✔ Valid CSS3
- ✔ Meets design specifications.

Demonstrator: _____
Date: _____

SITE UPDATE (1%)

Lab 24 Responsive Web Pages and Frameworks (1%)

Reference

 Lecture ??

Introduction

> *"**Responsive web design** (**RWD**) is an approach to web design aimed at crafting sites to provide an optimal viewing and interaction experience—easy reading and navigation with a minimum of resizing, panning, and scrolling—across a wide range of devices (from desktop computer monitors to mobile phones)"*[1]

There are three elements to RWD:

- Fluid layouts
- Flexible images
- Media queries

In this lab you will look at media queries to selectively apply CSS rules. Later in the lab you will look at using frameworks to ease the creation of fluid layouts. Finding out about flexible images is left to you but http://alistapart.com/topic/responsive-design is a good place to start.

Frameworks are collections of prewritten CSS rules that allow you to easily implement various layouts without all that tiresome fiddling around. In this lab you will look at a couple of frameworks.

1 https://en.wikipedia.org/wiki/Responsive_web_design

Activity 24.1 Media Queries Example

A media query allows access to device attributes allowing different CSS rules to be applied to an HTML page depending on the device being used to view it. The most obvious examples are using different style sheets for pages when they are viewed on small/mobile device screens or for printing.

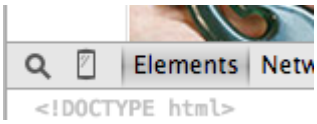
In this first activity you will see how a web page that uses media queries reacts to being viewed with different devices.

1. **Open Chrome and navigate to www.otago.ac.nz.**
   We will be using Chrome for this lab as there is a nice device simulator. Firefox has something similar, but not as nice.

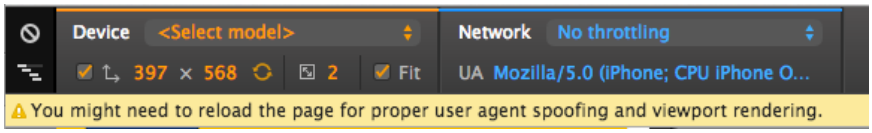
2. **Right-click anywhere on the page and select Inspect.**
   The viewport will be split into a number of frames and tabs that supply a lot of information about the page that is currently being displayed. This is like the Web Developers' Toolbar you have been using but much more powerful. It is an exceptionally useful tool when building web pages.

3. **Spend a few minutes experimenting with the various tabs to see what services it provides.**

4. **When you are ready click on the button that looks like a cell-phone, just next to the Elements button.**
   The page view will change to simulate a small screen device.



5. **Notice at the top of the page the new controls. Experiment by selecting different devices. The simulated viewport will resize itself. You will be prompted to reload the page. This allows the browser to download a device specific version of the page (if there is one).**



6. **Set the device to Amazon Kindle Fire and reload www.otago.ac.nz.**
   Note how the page looks. In particular the blue menubar on the left hand side.

7. **Grab the handle on the right hand side of the simulated viewport and gradually make the page narrower. Keep your eye on the width readout in the menu.**
   When it gets to 700 the page will change, the blue menu will disappear. This is due to different style sheets being applied as the size of the viewport crosses a certain threshold.

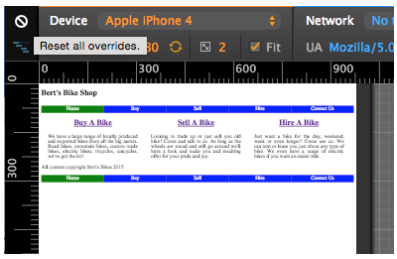
8. **Click on the  icon to see a representation of what styles are shown at what screen sizes.**
   Otago's home page is quite complicated and there are multiple style sheets and multiple media queries controlling exactly what rules are applied at any given width / height / media.

Activity 24.2 An Example Page

1. **Open MQExample.html in Taco and also Chrome. It is in the supplied lab files.**
   This is a very basic 'web site'; the links don't go anywhere and there are no images. However it will serve as a basis to see how media queries work.

2. **Have a look at the HTML and CSS source code. Experiment with the page in Chrome especially how it reacts to browser window size changes.**

This page is written to be responsive. Most of the major sizes are given in % and this works well for a wide range of screen sizes and resolutions.



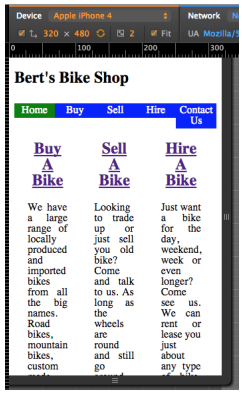
3 **With the page open in Chrome right-click and select Inspect and then click on the cell-phone button as you did in the previous activity. Select iPhone4 as the device and reload the page.**
After the page reloads it should look like the picture. The fluid layout has done its job but the result is not very usable.

4 **Add the following line in the <head> section of the HTML document.**

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This line gives the browser rendering the page instructions on page sizing and scaling.


http://www.w3schools.com/css/css_rwd_viewport.asp



5 **Save the HTML and reload the page in Chrome.**
It's different but not necessarily better. In the next activity you will write a style sheet for small screen devices and then use a media query to get the browser to select the appropriate style sheet at render time.

Activity 24.3 Adding a Media Query

Media queries allow the for inclusion of a block of CSS rules only if some condition is true. The media query can either be in the CSS or the HTML.

The `@media` Method

```
body {
        background-color: red;

        color: blue;
}

@media(max-width: 500px) {
    body {
        background-color: blue;

        color: red;
    }
}
```

1 **Start a new HTML page and save it as Mqcss.html.**
An `<h1> Hello</h1>` is all the content it needs.

2 **Add a link to a style sheet and add the CSS above to it.**

3 **View the page in a browser and make the window wider and narrower.**

If everything has gone well you should see the background colour change when the window width is less than 500px. Using the CSS method you can specify blocks of rules that are selectively applied.

The `<link media=...>` Method

The `<link>` method achieves exactly the same ends as the CSS method but does so by selectively applying whole style sheets rather than just a block of rules.

1. **Open MQExample.html in Taco if it isn't already open and add the following tag to the header section.**

```
<link rel="stylesheet" media="(max-width: 500px)" href="MQSmall.css" />
```

2. **Now make a style sheet called MQSmall.css and add a rule to it setting the background colour to red.**
   This is just a check to see if the link is working correctly.
3. **View the page in Chrome and test the rule is being triggered when you make the viewport narrower than 500px.**

It would be nicer for MQExample.html is that when it is viewed in a small screen device that the three sections containing the Buy, Sell, Hire headings were displayed one after the other rather than side-by-side.

4. **Look at the CSS, what is making the sections behave like that?**
   It is flagged in the CSS comments.
5. **Replace the background colour change rule with a rule so that the sections elements stack rather than sit side-by-side.**
6. **Test MQExample.html in Chrome by selecting the iPhone4 as the device.**

You could achieve the same effect of course using the @media syntax instead of linking to a style sheet that contains only one or two rules. Whether to use the @media or <link> syntax is quite a fraught topic but as you can imagine it depends on many factors such as: how many different rules there are, how likely you are to want to add new things, how reusable or portable the CSS ends up being and so on and so on.

1. **Modify MQExample.html (and its associated style sheets) so that it now uses the @media syntax to achieve an identical effect.**

Checkpoint

Have a demonstrator check your work now.

✔ MQExample uses @media media query to remove 3 columns.

Demonstrator:_____

Activity 24.4 More Complex Media Queries

So far we have only used very simple media queries, `media="(max-width: 500px)"` from the link tag above for example. There are however many possible values that the media property can take and what is more they composed (joined together) using logical operators.
The values the media property can take are one of two types, *media types* or *media features.*

Media Types

Media types describe the output type and may be one of: *all, print, screen, speech*.
Print as a media type (`media="print"`) is used to provide CSS rules for web pages that are

printed. They are used typically to remove navigation elements and advertising sidebars and change the font families.
We won't be looking at *speech*. *Screen* is for screens, computers, phones, tablets, projection etc. *All* is self explanatory and if no media type is given *all* is assumed. That is why above we were allowed to use `media="(max-width: 500px)"`. That media query doesn't specify a media type, only a media feature. In that case the rule would apply to any media.

Media Features

The list of media feature that may be queried is far longer than the list of media types. Media features are some property of the device a particular page is being viewed on. They include such things as width (as we have seen), height, aspect ratio, orientation (portrait or landscape). The reference below contains a full list and description.

 http://www.w3schools.com/cssref/css3_pr_mediaquery.asp

Media Query Composition

The elements of a media query maybe composed along the following lines:

<link rel="stylesheet" media="[*not* | *only*] <mediatype> [*and* | *not* | *only*] (<media feature>)" href="style.css">
And so CSS such as the following is quite proper.

```
@media only screen and (max-width: 500px) and (orientation : portrait)

{
    body{
        background-image: url("smallBg.gif");
    }
}
```

1. **Retrieve MQExercise.html and its associated MQExStyle.css from the course files and open them in Taco.**
   These files are an extended version of the Bob's Bikes site you saw earlier.
2. **Using the reference below as a guide make styles for MQExercise.html for the following media type / media feature combinations. Use the `<link>` method for the first one and the @media method for the others.**

- **Media type: Print**
  - ✔ **the navigation shouldn't be displayed.**
  - ✔ **the default font family should be set to a serif font.**
  - ✔ **the ads shouldn't be displayed.**
- **All media types, screens narrower than 800px.**
  - ✔ **The ads (in the .col-4 sections) should be displayed two to a row.**
- **All media types narrower than 600px.**
  - ✔ **All sections should be stacked in a single column.**
- **Screen only narrower than 600px.**

✔ **different colour backgrounds for different orientations**

Checkpoint

Have a demonstrator check your work now.

✔ Four different media queries implemented appropriately.

✔ **Demonstrator:**____________________________

Activity 24.5 A Simple Framework

A CSS Framework is a pre-written collection of CSS rules in one or more style sheets that you can use to get all sorts of layout functionality without having to write a line of CSS yourself. The huge advantage of this is that you don't have to spend lots of time writing fiddly CSS yourself. CSS that scales well and doesn't have annoying quirks that you couldn't quite fix. Frameworks can speed up prototyping and development enormously. The downsides are that there is a learning curve (although that can be quite gentle) and you may end up with a lot more CSS than you actually need, which is inefficient. However, used carefully frameworks are fantastic. In the following activities you will firstly look at an incredibly basic framework and then at a more substantial one.

1 **Retrieve SimpleCols.html and its associated CSS file, SimpleCols.css from the lab files. Open both in Taco and preview the html file in a browser.**
SimpleCols.css is the actual framework. It is just a set of CSS rules that allow for easy columnisation of a layout. SimpleCols.html describes how to use the SimpleCols framework and shows examples.

2 **Open mySimpleCols.html in Taco, it contains the majority of the HTML you need. Rearrange the HTML blocks and add appropriate classes to recreate the layout shown below.**

Activity 24.6 Other CSS Frameworks

There are very many CSS frameworks available. Some are simple, some more complicated, some better, some worse. They are all however variations on a theme and so once you understand how a few work then you can pretty much use them all. In this next activity you don't need to write any actual code. Just read a bit about a couple of frameworks and be prepared to tell a demonstrator what you have discovered.

1 Choose one of the following frameworks and read enough of its documentation so that you can explain to a demonstrator what it is intended for and how to use it.

- **http://purecss.io/**
- **http://getskeleton.com/**
- **http://www.amazium.co.uk/**
- **http://jslegers.github.io/cascadeframeworklight/**
- **http://www.cascade-framework.com/**

Lab Completion (1%)

Have a demonstrator check your work now

- ✔ Media Queries
- ✔ Simple Framework
- ✔ Other Frameworks

Demonstrator: _____

RESPONSIVE WEB PAGES AND FRAMEWORKS (1%)

Lab 25 PHP (1%)

Reference

 Lecture 24

 www.php.net

Introduction

In this lab you will be introduced to PHP. PHP is a server-side scripting language which can generate HTML for the web server to pass back to the browser. PHP is widely supported and is used by many organisations including Wikipedia and FaceBook. One of PHP's great assets is how easily it makes information submitted via a form available to the programmer.

Like the JavaScript lab earlier we can give you only a tiny taste of what PHP is like.

Activity 25.1 Hello World!

The first thing you will do is make sure you can get some very basic PHP working. This will prove that the basic infrastructure is in place and that you are naming and saving files correctly. This is often referred to a the "Hello World" program. You did something very similar in the JavaScript lab as well.

1. **Start a new document in Taco.**
2. **Leave the default HTML content in place but save the file as test.php**
   Files that contain PHP need a .php suffix otherwise the server will not know that it should pass them through the PHP processor.
3. **In the `body` of the document add the following 'php island':**

```
<body>
  <?php
     echo("It Works!");
  ?>
</body>
```

   The `<?php` and `?>` open and close the php segment. They tell they web server that it should pass that segment to the PHP module and wait for whatever it returns.
   The `echo` statement is executed by the PHP module and it produces the output "PHP Test Page" which it returns to the web server, where it forms part of the html stream being sent to the browser. At least that's the theory, lets see if you can get it to work.
4. **Save the page and upload it to a new directory in your csnet account.**
   You can't check PHP pages locally unless your local machine has the PHP interpreter installed and Taco is configured to use it. On your lab machines this may or may not be done. I will assume it has not, so upload your file to the server.
5. **View your test.php in a browser.**
   Hopefully you should see "It Works!" in the browser window. If you don't something went wrong. Go back and check for typos and get it working before moving on to the next step.
6. **In the browser click on View Source.**
   Notice that there is no PHP code in the document the browser receives, only what was 'echoed'. Notice also that the string isn't marked up with html tags.
7. **Add tags to the echo statement:**

```
echo("<h1>It Works!</h1>");
```

8. **Save the file, upload it and view it in a browser and then View--> Source.**
   Now the browser gets properly marked up content which will validate.

Activity 25.2 Comments

Comments in PHP are the same as in JavaScript. Remember to comment your code sensibly.

```
// Single line comments

/* Comments that are so long that they need more than one line to record
the fullness of their message */
```

Activity 25.3 Variables

Recall that variables are the containers that hold your data and that you refer to them by the variable name. In PHP variables are very simple. You don't need to specifically create variables (as you do in JavaScript with the var statement), you just start using them and they are created automatically.

A variable name always starts with a '$' and then a letter, and then a combination of letters, digits and underscores. So:

- $a
- $a_
- $index
- $my_Salary
- $TheTop_10
- $t_hhh____298176253

Are all legal variable names, although I suggest that the last one is best avoided as a variable name.

The following are not legal variable names:

- input
- $_input
- $99percent
- $super*

1 Add the following code (inside the php markers) to test.php.

```
$quote = "Life well spent is long."

echo($quote);
```

The first line creates a new variable and assigns to it (sets its value to) the text string "Life well spent is long" (without the quotes).
When echo encounters a variable name (instead of something in quotes"") it gets the value of the variable and echoes (prints) that.

2 Upload your file and view the page with a web browser.
The page won't validate (try it).

3 Modify the php code so that appropriate html tags are produced as well.

4 Modify the code so that a new variable is created which contains the author of the quote, Leonardo da Vinci,.

5 Now modify the code so that first the quotation is output (surrounded by appropriate html tags and then the citation, again in proper tags.

6 Save and view your page to check.

Checkpoint

Demonstrator:_____

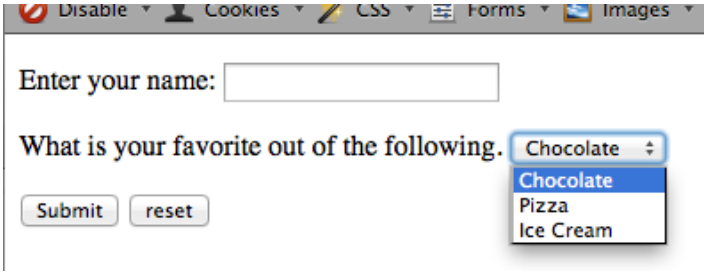
Activity 25.4 Responding to forms

PHP is a server-side language. It is generally used to respond to or process some user input usually gathered via an HTML form. In this first activity you will make a simple form and write PHP to respond to what is entered into the form.

Write the HTML

1 Start a new html document and save it as form.html.

2 Add code to it so that you have a simple form with two inputs.



3 **Ensure you do the following:**
The form *action* should be processForm.php (you will write this next).
The form *method* should be "get".
The name attribute of the text field should be 'name'.
The name attribute of the select should be 'fave'.
The values for the <select> options should be: "choc", "pizza" and "ice cream".

Write the PHP

1 **Start a new, blank document and save it as processForm.php.**
In this document you will put the PHP code to be executed when a user pushes the Submit button on the form.
Firstly, as usual, we'll do something incredibly simple, just to make sure everything is pointing to the right place.

2 **Add a line of PHP to processForm.php just to check...**

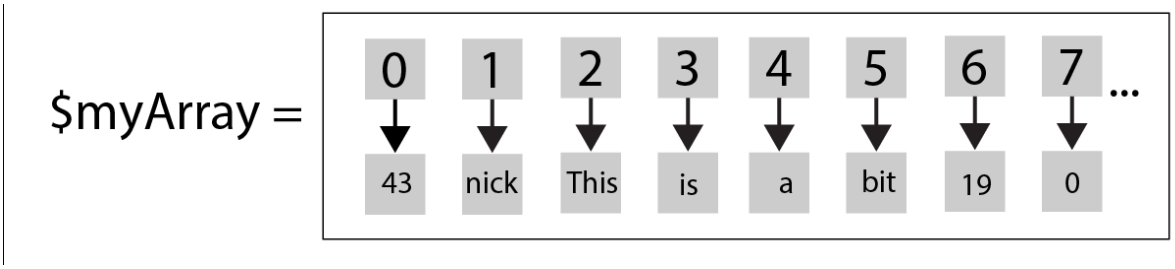
```
<?php
   echo("<h1>It Works!</h1>");
?>
```

3 **Save form.html and processForm.php and upload both files to csnet.**

4 **Try viewing form.html in a browser and clicking the submit button**
All things being well you should see a page with "**It Works!**" in big black type. If you don't see that something has gone wrong. Check for typos, both in the HTML and the PHP. Get this bit working before moving on to the next step.

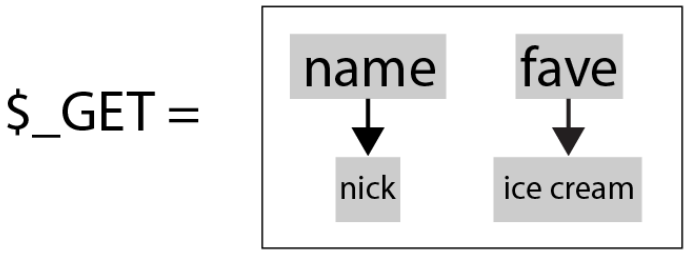
Getting hold of user input

Getting hold of user input supplied through HTML forms and processing it is what PHP is all about and it makes getting hold of the actual user input (comparatively) very easy. I say comparatively because it involves the use of an 'array'. At its simplest an array is a collection of data values stored in numbered (indexed) array elements, just like in the earlier JavaScript lab.



Thus $myArray[1] holds the value 'nick' and $myArray[6] has the value 19. Simple.
In PHP the name of an array of values from a form that used the GET method is $_GET; and rather

than using numbers to index each element of the array the 'name' (from the name attribute) of the HTML element is used. So if I filled in the form with the name "nick" and selected "Ice Cream" as my favourite then I would expect to have the following array available to me:



And in my program I could refer elements $_GET['name'] and $_GET['fave']

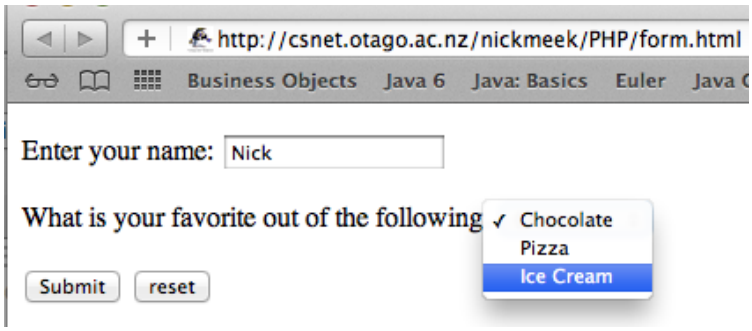
1 **Remove the previous line and add the following to processForm.php.**

```
print_r($_GET);
```

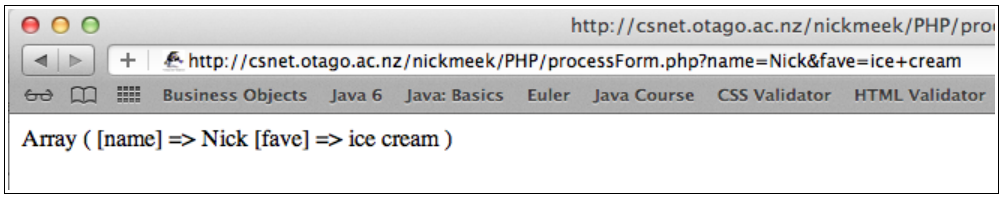
The print_r() function prints out the contents of any array and as we now know $_GET is an array.

2 **Save and upload processForm.php.**
Remember you will have to re-upload the file each time you make a change.

3 **View form.html in a browser, fill in the two inputs.**



4 **And click submit.**



The print_r() function tells you that $_GET is (indeed) an array with two elements. The first element is called $_GET[name] and it contains the text "nick". The other element is called $_GET[fave] and it contains the text "ice cream".
Now all you have to do is get PHP to echo something sensible back to the screen.

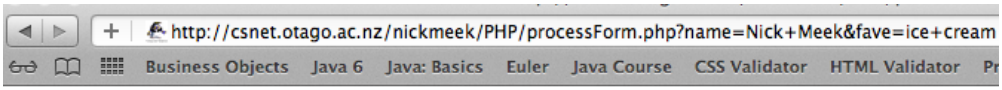
Generating the response

1 **Remove the previous line of PHP and replace it with:**

```
echo("Hello $_GET[name]");
```

2 **Save and upload your page and test it by filling in the form and clicking submit.**

3 **Modify the line so that its output is similar to the following.**

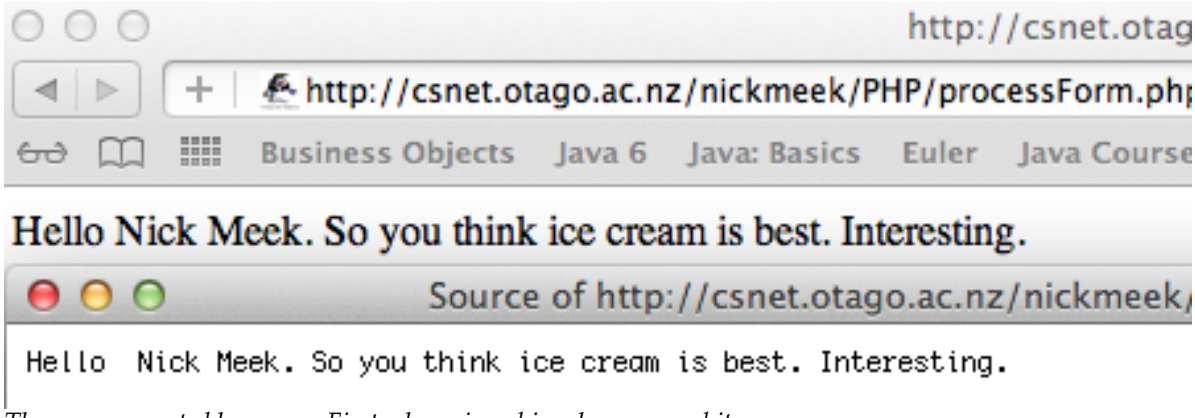


Hello Nick Meek. So you think ice cream is best. Interesting.

Notice the URL has been modified, the data entered in the form is tacked on the end.

Adding the HTML

If you look at the source that processForm.php generates and returns to the browser it is pretty sparse, it certainly isn't an HTML document, it doesn't have a doctype for a start.



The page generated by processFirst when viewed in a browser and its source.

1 **Start a new HTML document in Taco.**

2 **Select and copy all the HTML that Taco inserts automatically**
This is what is copied from your template file and should be the bare outline of an HTML document.

3 **Close the new document without saving it.**

4 **Paste the text you have on the clipboard into processForm.php.**

5 **Move the PHP block of code to the start of the HTML <body>.**

6 **Add and tags around the PHP block. Save, upload and test your page. Both form.html and processForm.php should validate.**

Checkpoint

Demonstrator:_____

Activity 25.5 Choose your own style

In this next set of activities you will construct a form which allows users to select a different style sheet for a page depending on their preference.

1 **Look at the movie changeDemo.mov in the coursework files so you know what you will be making.**

Make the HTML form

2 **In the lab files for this lab you will find change.html and three style sheets, style1.css, style2.css and style3.css. Copy them to a directory in your home drive.**

3 **Using Taco open change.html**
This is a very unexciting page that contains paragraphs of fake text.

4 **Add HTML markup above the h1 element to add a form with three radio buttons.**
Make the *action* of the form "change.php.
Make the *method* of the form "get".
Name the three radio buttons the same but give them different values, I used "1", "2" and "3" for the values.

Write the PHP

We are going to change the name of change.html to change.php so we can add php code to it.

1 **Save change.html as change.php and delete change.html.**

2 **Add a line of PHP to change.php to check everything is pointing to the right thing, namely itself in this case.**

```
<?php

   echo("<h1>It Works!</h1>");

?>
```

3 **Save change.php and upload it to csnet, try viewing it with a browser.**
All things being well you should see a page with "**It Works!**" in big black type. If you don't see that, check for typos, every character is vital.

Activity 25.6 Getting hold of the input

Now you need to write the code that gets the input from the form and writes the relevant HTML.

As we have seen PHP makes it really easy to access data input by users via forms. The HTML form which calls this PHP page uses the "get" method. That means that PHP will automatically make the information the user put in the input on the form available through an array called $_GET[<*input element name*>].

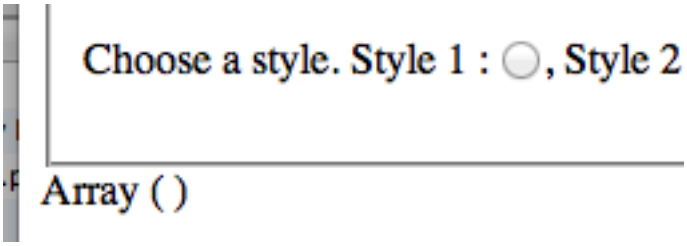
1 **Add the following lines to the body of change.php immediately below the form:**

```
<?php
  print_r($_GET);
?>
```

Remember the print_r() function prints out the contents of an array.

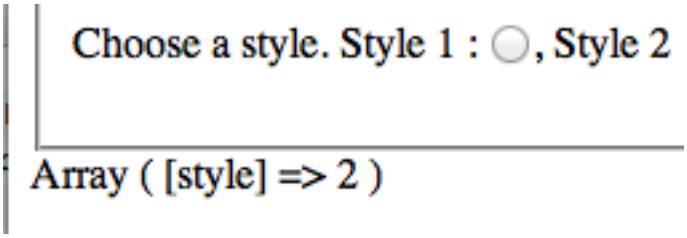
2 **Save and upload change.php.**

3 **View change.php on csnet using a browser. You should see the following.**



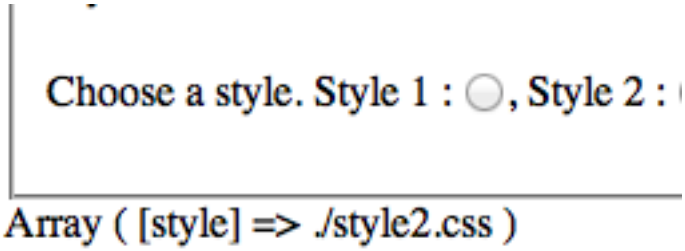
The first time the page is loaded none of the form elements had a value and so the $_GET array was empty.

4 **Click one of the radio buttons and click Submit, you should see something like this:**



The array has a single element called 'style' with the value "2"

5 Modify the values of the radio buttons so that they return strings that look like CSS file references.



Now all you need to do is get PHP to write that string in the head section of the HTML document.

6 Delete the previous PHP block and modify the head section of change.php as shown.

```
<link rel="stylesheet" type="text/css" href="<?php echo($_GET[style]) ?
>">
```

7 Save, upload and test change.php. Does it work as it should?

There is one last thing to do. Currently if none of the radio buttons are selected then no style sheet is selected, view the source generated by change.php when none of the options is selected.

8 Add a line of HTML to change.php so that a default style sheet is used if no option is selected.
Hint: Think about the way style sheets cascade.

Lab Completion (1%)

Have a demonstrator check your work now.

- ✔ Activity Basic form exercise completed.
- ✔ Activity Style Sheet exercise completed.

Demonstrator: _____
Date: _____

PHP (1%)

Lab 26 Catch-up

Introduction

This is an opportunity to catch-up on any work you may have missed so far. Remember, you can have labs marked late.

Appendix A Javascript Form Validation

Introduction

Earlier we looked at using HTML5 to validate form input. This is great and takes a lot of rather tedious work away from the web developer and embeds it in the browser. However, what if the page is being viewed in a browser that doesn't support HTML5? In that case you can use JavaScript to achieve a similar result. In this lab you will use JavaScript to validate a form.

JavaScript is often used for client-side form validation. In this activity you will add some JavaScript to the Toyger Order Form so it is checked for (some) errors before it is submitted to the server. The JavaScript will be triggered when the Submit button is clicked and will check the following:

- ✔ There is something in both of the name fields.
- ✔ There is a number in the phone number field.
- ✔ There is a valid email address in the email field.
- ✔ One of the radio buttons are selected.

If the form isn't correctly filled in the JavaScript will stop the form being submitted and display a helpful error message for each detected error.

It is important to note that you will be starting with a working form and adding something to it. The form doesn't *require* the JavaScript, it works perfectly well without it. If the JavaScript fails for some reason (disabled by the user for instance) then the form will still work.

1. **Copy ToygeyComplete.zip to your home directory and unzip it.**
   It contains a completed version of the Toyger Order form.
2. **Watch the very short video (FormDemo.mov) that shows a finished form in action.**
3. **Open order-form.html in a browser and click on the Submit button.**
   Notice you are directed to a Success page rather than the form being submitted to the server. This is just faster and easier for this type of development. Later on (before the pages are properly deployed) you would need to change the action of the form to the correct server-side script. Notice also that the Success page redirects you to order-form.html after 2 seconds. Again this is just faster and easier during development. The following meta tag in the head section of Success.html controls this.

```
<meta http-equiv="refresh" content="2;url=./order-form.html">
```

4. **Open order-form.html in Taco and look through the markup.**
5. **Add the following to the `<head>` section of orderForm .html**
   JavaScript files can be embedded in the html document (as above) or linked. It is generally considered better to link JavaScript files.

```
<script src="./validateForm.js" type=”text/javascript”></script>
```

   This tells the browser where to find the JavaScript file which you will create next.

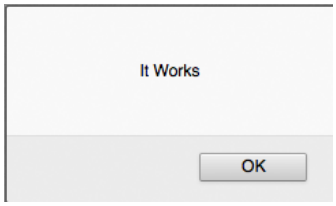
6. **Start a new, empty text file and save it as *validateForm.js* to the same directory as the other files.**
   This is the file where you will write JavaScript code.

You should now check that this link is working correctly. To do this you will place some code in the JavaScript file that will display an alert box when the JavaScript is run, just like in the previous JavaScript lab.

7. **Add the following to validateForm.js:**

```
alert(“It Works”);
```

8. **Open order-form.html in a browser.**
   You should see an alert box. Clicking the Ok button should make the alert box disappear and the page finish loading.



Activity 27.1 Using `onsubmit`

The `<form>` tag has many possible attributes only some of which we have covered. See http://www.w3schools.com/jsref/dom_obj_form.asp for a more complete list. We are going to use the `onsubmit` event. The form of the `onsubmit` attribute is:

```
onsubmit="<some JavaScript>".
```

1 **Modify the form tag in order-form.html to include the `onsubmit` attribute.**

```
<form ... action="success.html" onsubmit="validate()">
```

Now, when the Submit button is clicked the validate() function is called (you will write this function next).

2 **Add the following code to validateForm.js and resave the file.**

```
alert("It works");

function validate(){

  alert("It Still Works");

}
```

3 **Reload order-form.html in a browser.**
You should get an "It Works!" alert box. Click Ok.

4 **Click the Submit Form button.**
You should see an "It Still Works" alert box.

5 **Delete the first line (`alert("It works");`) so that you are just left with just the function and test your JavaScript by reloading order-form.html.**

Notice however that when you submit the form and click on OK in the alert box that you are still sent to the Success page. You need to modify things somehow so that after the function does some stuff (checking the form) then the form either is or isn't submitted depending on whether the form validates or not.

1 **Modify the forms onsubmit attribute as follows:**

```
onsubmit="return validate()"
```

Now if the return value of the function is 'true' the browser will submit the form, if the return value is 'false' it will halt the submission process.

2 **Modify the validate() function in validateForm.js to return false.**

```
function validate(){

  alert("It Still Works");

  return false;

}
```

3 **Save the files and reload order-form.html in a browser. What happens when you submit the form?**
Feel free to get rid of the `alert(); when you begin to find it annoying.`

4 **Change the return value in validate() to "true", what happens now?**

Now all that is needed is code in validate() that returns 'true' or 'false' depending on whether the form fields are filled in correctly or not.

Activity 27.2 Form validation algorithm

Over the next few activities you will add code to validateForm.js. The code will check each field in the form and see if it has the some 'legal' value in it; you get to decide what legal means.

If all this sounds a little daunting don't panic. You will be supplied with all the code you need. This lab is just about showing you the sorts of things that are done with JavaScript. You are not expected to learn it and you will not be quizzed on it later.

This form checking function needs to do the following:

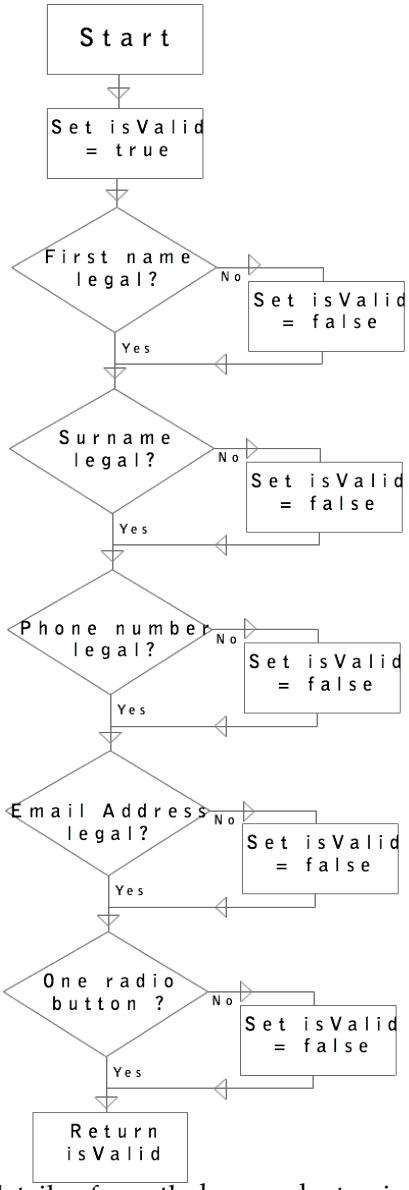
If all the fields in the form have legal values then the function should return 'true', but, if even one of the fields does not contain a legal value then the function should return 'false'. It's a bit like the sorts of checks you might do before leaving home in the morning. "Phone? Check. Wallet? Check. Keys? Check. Trousers? Check. But if any one of these checks fails you shouldn't proceed until it is corrected (especially the trouser one).

The following are, very broadly, the steps or operations that need to be carried out in order to accomplish this task. This sequence of steps is called an 'algorithm'.

We will use a single value to keep track of whether the form is correctly filled out. We will call this value 'isValid'. It can have one of two values, true if the form is correctly filled out, false otherwise.

1. To start with set isValid to 'true'. That is, assume the form **is** filled out correctly. isValid is the value the function will return when it is finished.
2. Check what value the text field firstname has, if it not a legal value set isValid to false and proceed to the next step otherwise just proceed to the next step.
3. Check what value the text field lastame has, if it not a legal value set isValid to false and proceed to the next step otherwise just proceed to the next step.
4. Check what value the text field phone has, if it not a legal value set isValid to false and proceed to the next step otherwise just proceed to the next step.
5. Check one of the radio buttons are selected, if it not a legal value set isValid to false and proceed to the next step otherwise just proceed to the next step.
6. Check that the user has entered a valid email address, if it not a legal value set isValid to false and proceed to the next step otherwise just proceed to the next step.
7. Return the value stored in isValid.

The same algorithm may be expressed as a flowchart.



The next task is to fill out the details of exactly how each step in the process is actually done and express the algorithm as program code that the computer can execute.

Activity 27.3 Variables (recap)

You can think of variables as named containers (little pieces of computer memory) in which you can store values (data or results) while your program is running. Often the things stored are simple things like a numeric value, 7 or 22.91 for instance or the thing stored might be some text, "my name". It might be a boolean value, a boolean value is either true or false. Sometimes the things stored in variables are much more abstract, we won't worry about them.

1 Add the following as the first line of code to the validate function.

```
function validate(){
  var isValid = true;
}
```

You have now declared a *variable* called 'isValid' and set its value to the logical value 'true', remember we are assuming the form is correctly filled in.

2 Now change the return line in the function to:

```
return isValid;
```

Now `return` will return whatever value isValid is storing.

3 Check that everything is working as it should by reloading the form in the browser and trying to Submit it.
It should submit fine (because the JavaScript function is returning 'true') and you should be directed to the Success page.

4 Try changing the value of isValid to 'false' in the variable declaration. Does the form still work as it should?

Activity 27.4 Validate the first name field

Now we are going to start checking that the form is filled in correctly. We will do this one field at a time starting with the First Name field. We will get this bit working before moving on to the next bit.

Firstly we need to 'grab hold' of the name input field so we can examine it to see if the user has entered an acceptable value. We will grab hold of it by using a (JavaScript) method of the document (Object) and the element's id attribute. We will store a reference to the input field in a variable.

If all this sounds completely incomprehensible don't worry, you have been dropped straight in at the deep-end of object oriented programming. Just follow along and try to get a sense of what is going on.

1 Modify your script to include a variable declaration and assign to it the element with the id="firstname".

```
function validate(){
  var isValid = true;
  var fName = document.getElementById('firstname');
  return isValid;
}
```

Notice also I have removed the alert box.

2 Now we can check to see what the value of the text field is and act accordingly. Add the following to your script:

```
function validate(){
  var isValid = false;
  var fName = document.getElementById('firstname');
  if(fName.value == ""){
     isValid = false;
  }
  return isValid;
}
```

Note the double equals sign (==) to check equality.
Note we are checking to see if the value (contents) of the input field is equal to "", the empty string, that is, nothing was entered into the field. If this is true isValid is set to false which will stop the form being submitted.
Actually even a space will pass this test (because a space {" "} isn't equal to {==} the empty string {""})so lets at least check for that as well.

3 Modify your code as shown:

```
if(fName.value == "" || fName.value == " "){
```

The two vertical bars (||) mean 'OR'. So the above condition is TRUE if the input is the empty string ("") or if the input is just a single space (" ").

Activity 27.5 Adding feedback

It would be nicer if we could let the user know why the form didn't submit. With what you know already you could easily add an alert box informing the user what the problem is. It would be nicer though to mimic the feedback provided by HTML5.

1 Add the following line to your code.

```
if(fName.value == ""){
    isValid = false;
    alert("You must supply a first name");
}
```

2 Check that your code works properly by testing the form in a browser.
Remember to test by putting both legal and illegal names in the First Name input.

Activity 27.6 Validate the surname field

1 Add another variable declaration just below the previous one and use it to reference the Surname text field.

```
var sName = document.getElementById('surname');
```

2 Add another `if` code block to ensure the Surname field is not empty or just a single space.
You can copy and paste the previous if block from `if(....` to `}` and just change the appropriate bits (italicised below).

```
if(change this.value == "" || change this.value == " "){
    isValid = false;
    alert("You must supply a change this name");
}
```

3 Test your form throughly again.

Activity 27.7 Nicer feedback

Giving all the feedback using alert boxes is ugly and clumsy, you can do a lot better than that. In this next activity you will modify the JavaScript so that the error messages appear on the web page itself.

You will notice, if you look carefully at the html markup, that there are some mysterious empty paragraph elements, `<p id="firstnameError" class="error"></p>` for instance. There is also styling for these elements in the CSS.

1 Locate the above element in the html and add some content to it.

```
<p id="firstnameError" class="error">This is a test</p>
```

2 View the page in Firefox.
What you need to do now is get JavaScript to put the error messages in these paragraphs and dispense with the alert boxes.
To modify the contents of an html element you need to set its '*innerHTML*'.

3 Create yet another variable to point to the paragraph with id="firstnameErr"

```
var fNameErr = document.getElementById('firstnameError');
```

4 Now replace the alert error message with:

```
fNameErr.innerHTML = "You must include a first name.";
```

5 Test your page to ensure it works.
You will notice that the error message doesn't go away when the user corrects their input. You need to add code to clear the content of the error paragraphs. You really need to add a step to the algorithm. Immediately after declaring the variables you need a new step
1a: Clear all error messages.

6 Just under where you declare your variables put the following:

```
fNameErr.innerHTML="";
```

So, every time the script is run, that is whenever the submit button is clicked, one of the first things that is done is to clear the error fields of any previous content.

7 Modify your code so that the Surname error messages work in a similar way.

Activity 27.8 Checking the phone number

Lets make sure the phone number input is sensible. We could go into a lot of depth checking the validity of a phone number. Let us just assume that the phone number is ok as long as it consists of only digits, spaces and hyphens.

There are a number of ways we could check that the user input was a 'valid' phone number, the best probably is through the use of *regular expressions*. A regular expression, according to Wikipedia "provides a concise and flexible means for "matching" (specifying and recognizing) strings of text, such as particular characters, words, or patterns of characters.". Regular expressions are extremely powerful tools. Creating them however is well beyond the scope of the course. In the following activities we will provide the regular expression and give a brief explanation of what they match. If you wish to know more about them there is plenty of information online.

1 Create a variable to point to the text input with id="phone" and to the phone error paragraph.

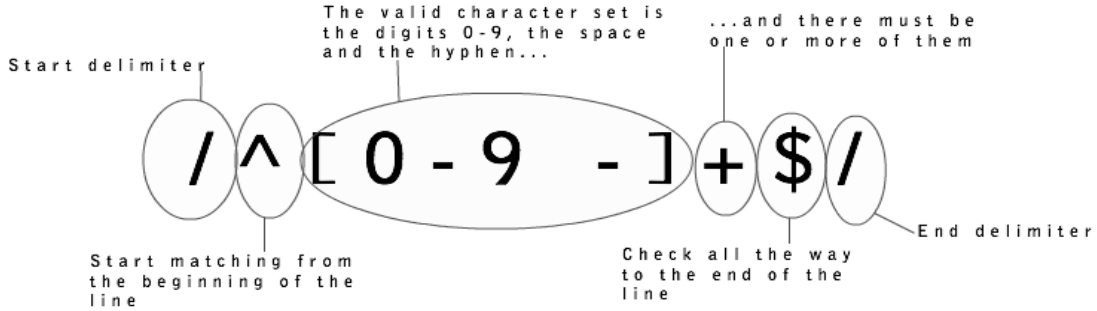
```
var phone = document.getElementById('phone');

var phoneErr = document.getElementById('phoneError');
```

2 Create a variable to hold the regular expression.
This is the regular expression we will use to check for 'valid' phone numbers:
/^[0-9 -]+$/

```
var phoneNum = /^[0-9 -]+$/;
```

This is a regular expression that will match complete lines of text that contain only one or more of the digits 0-9, spaces and hyphens.
It will match the following strings:
1
123
12 3 –56 -
-
It will not match:
<An empty line>
123a--
asde1
and so on.

Read the regular expression as follows:



3 Add a new block of code to check that the phone number is up to spec.

```
if(!phone.value.match(phoneNum)){

}
```

You can read this as follows;
If it is **not** the case (the '**!**' means not) that
the value of the input box 'phone ' (phone.value)
matches the regular expression signified by 'phoneNum' (.match(phoneNum))
Then do the following. .. The following will of course be to generate an error message and set isValid to 'false'.

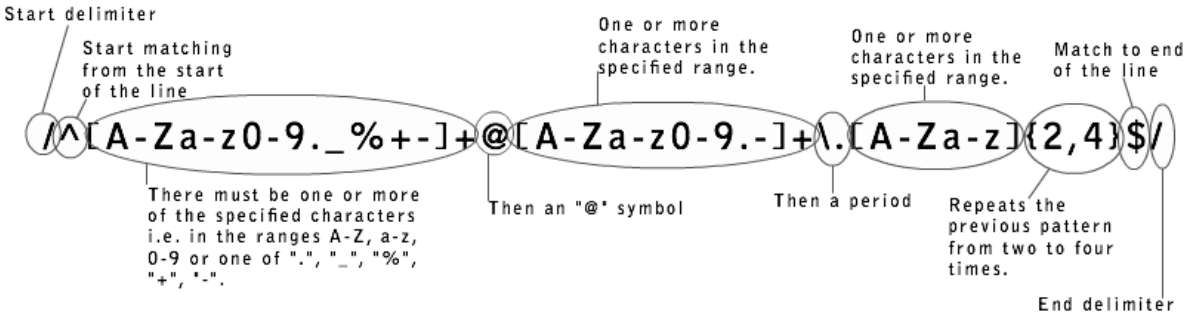
4 Add JavaScript in the body of the if statement to generate an appropriate error message and set isValid to false.

5 Test your script works as expected.
Remember you will also need to add code to clear the error message at the beginning of the script.

Activity 27.9 Checking the email address

Checking that a particular string of characters is a valid email address can be achieved in a number of ways. The most certain method is to send an email to that address and see if it bounces. Another way to check is to look at the address and see if it meets certain rules. An email address should start with a number of letters, digits, periods, underscores etc. Next it should have an @ sign and then some more characters and then a period etc. etc. So bill_bob.smith@sos.ds.cd might be a valid email address but @sos.ds.cd and bill_bob.smith.sos.ds.cd most definitely aren't. Again regular expressions can provide a solution. The regular expression below checks to see if a string of text might be an email address, that is that is has the general form of an email address. It does not guarantee that the string is an actual, in existence, email address.

```
/^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$/
```



1 **Add the code needed to check the email address.**
It will be very similar to the phone number code. You will need to:
   1. Create variables that point to the email input text box and the email error paragraph.
   2. Create a variable to hold the regular expression.
   3. Add code to reset the phone error message.
   4. Add an block of code to check if the input matches the regular expression and does appropriate things if necessary.

Activity 27.10 Checking the radio buttons

We need to ensure that at least one of the radio buttons is checked.

1 **Add the following code:**

```
var radioBtns = document.getElementsByName('contactpreference');

var contactErr = document.getElementById('contactError');
```

Notice this is slightly different than the method you used before, this one returns **elements** by their name. As it happens there are more than one element with the name 'contactpreferences', in fact there are two. Rather than returning a single element it returns an *array* of two elements which you can reference like this: radioBtns[0] and radioBtns[1].

2 **Add the following code to check that at least one radio button is selected and supply an appropriate error message.**

```
if(radioBtns[0].checked == false && radioBtns[1].checked == false){
  isValid = false;
  contactErr.innerHTML="Please select one option.";
}
```

The "&&" operator means logical 'AND'.

3 **Check that your script works correctly.**
Remember you will also need to add code to clear contactErr at the beginning of the script.

You should note that the script developed here is a very naive solution; it could be improved in many ways. Such improvements are beyond the scope of this course and you should consider papers like COMP150 or COMP160 if you want to understand how to write better scripts.

Activity 27.11 Checking the checkboxes

Discussion

Checking that at least one of the checkboxes is checked is similar to checking if a radio button is clicked, and you could write the condition for the if statement in a very similar way:

```
if(checkBoxs[0].checked == false && checkBoxs[1].checked == false &&
checkBoxs[2].checked == false && checkBoxs[3].checked == false &&
checkBoxs[4].checked == false && checkBoxs[5].checked == false)
```

This is quite tedious however and isn't very scalable. What if there were 100 check boxes?, a 1000?

A more scalable solution would use the code below.

```
var checkBoxs = document.getElementsByName('interestedin');
var check = false; //We assume that no checkbox is checked
var interestErr = document.getElementById('interestError');
var i = 0;
interestErr.innerHTML="";
/*Loop through all checkboxes, only one needs to be selected
for the condition to be satisfied*/
for (i=0; i < checkBoxs.length; i++){
  if(checkBoxs[i].checked == true){
     check = true;
  }
}
/*Check to see the if a checkbox was checked and responding
appropriately*/
if(check == false){
```

```
  isValid = false;
  interestErr.innerHTML="Please select one option.";
}
```

4 Copy this code to your script and make sure it all works properly.

Appendix B Working Outside the Lab

Uploading files to your S:Drive with NetStorage

NetStorage enables you to transfer files to and from your S:Drive directory over the web. This means that you can access your S:Drive directory from wherever you have web access. This is a good way of making your work available for you to work on at home.

1. **You can only use this system to transport files. So, if you want to transport a directory or a number of files, make sure everything is in one directory and "zip" that directory up.**
   See lab one for instructions on how to zip a file or directory.
2. **Go to:** http://www.otago.ac.nz/netstorage**. You will need to enter your University username and password.**
3. **Double-click on the** SDrive@OUS **directory to access your S:Drive contents.**
4. **Select** File> Upload**, select browse to find the file you want to upload.**

Working on files outside of the COMP112 lab

All you need to write your HTML files is a text editor.
Standard installed text editors on Windows machines are Notepad and Wordpad, on a Mac OS 9 machine: SimpleText, and on a OS X machine: TextEdit (though you will have to change the file type from rich text to plain text in the preferences).

If you would like a text editor with a few features like Taco and you don't have OS X, then you will be able to find something comparable to download free.
In the Resources section of the COMP112 web site we provide some links to suitable applications.

The best environment for completing your COMP112 practical work is the COMP112 lab.
Working outside of the lab is valuable for practice and preparation, but you cannot get the help and guidance that we provide in lab. If you have trouble making it to your streamed lab times, see a Teaching Fellow about changing to a more suitable time.

Appendix C Text for Labs 11 & 13

A Guide to High Quality Web Development
The World Wide Web is populated by millions of web pages. It is possible for almost anyone to create and upload a web page in next to no time at all. But, as in all things, there is a world of difference between an ill-conceived amateur product and that created with professionalism, quality and usability in mind.

There is so much more to a good web site than markup quality. It must be easy to use and appealing in order to maintain an audience. Poor quality markup makes a site difficult to maintain and prone to exhibit problems as newer browsers are released. Poor interface makes a site unappealing to its audience. True quality is achieved when sensible standards-based markup is combined with an effective structure and design.

This guide endeavours to direct you through a process that will help you to produce quality results. There are three basic processes involved:

Planning
In all projects the planning phase is key. Without a well-considered plan the creation of your site will be fraught with difficulties and frustration and your end product risks being confused and inconsistent. Many questions must be asked and answered to establish the specific needs of each site you create.

Construction
Though it is tempting to leap in and construct your whole site in one go, it is more sensible to construct a single template page first. Once that page has been through the evaluation phase of the process you can return to construction and create the rest of your site, content in the knowledge that your foundation markup is sound.

Evaluation
Evaluation is a phase that you will repeat during the course of a project. Begin evaluation after you have created and validated. This phase will make sure that your site is usable and sensible. When your site is complete, evaluate it again to ensure that it works as a unit.

Your movement through these three phases is not strictly sequential. Planning is always your first port of call, but construction and evaluation are phases that you will alternate between during the course of a project.

Planning
Before you can design the structure of your site, you must know what the purpose of your site is and what information it needs to convey.

No web site will succeed if it does not consider the needs of its audience.

To start the planning process you need to answer some questions.

What is the purpose of your web site?
If you were creating a site for a business, the purpose of your site may be to provide information about products and services and a means for potential clients to make contact. Different sites will have different goals. It isn't enough to need a web site, you must know why.

Who is your target audience?
Considering your expected audience for a site may influence some design decisions. Your user is paramount. They should consider your site useful and easy to use or you risk losing them. It is important to keep your audience firmly in mind throughout the development of a site; their reaction will determine the success or failure of your project.

What will be the content of your site?
Once you have established what your site should achieve and whom your target audience is you need to establish what content you will need.

Solid content is vital. Your site can be as well structured as you please but if it is devoid of content it is no use to anyone. Gather together the resources you already have and consider what else you need to obtain.

Before you start to markup you should have text (edited and proofed) and images ready to go. There may well be tweaks to these later in the process but without the core of your content in existence you cannot effectively design your site.

How many pages (or sections) will your site contain?
Establishing the size and basic organisation of your site allows you to start to organise your content appropriately. Now that you know what your site will contain you can start to map out how that imformation will be presented in your site structure.

How will the information be split up?
The example below is for an imaginary business site. There are seven pages in total. The main page is understandably at the top of the structure. From that point information is divided into three sections: products, services, and request/contact information. The products section in turn hosts subsections of products (plastic-ware, metal-ware and all other products).

Image goes here

What is the site's navigation system?
Take the diagram you have just produced and consider what navigation should be provided on each page. This is a good time to establish a filename for each page.

At a bare minimum each page should have a link back to the "home" page. Ideally each page will have links to at least the other pages in their section. Users don't want to have to hop back and forth through your structure to navigate your site. Give them the cleanest most comprehensive navigation system possible.

So, for the site diagrammed above, there will be a link to the home page on every page and the following as indicated by blue arrows:

Image goes here

Now you will have a complete map of your site, outlining how it will all link together.

What will each page of your site look like?
Sketch an outline for your template page. Indicate where your navigation, title and main content will sit. Describe how the site will work. Contemplate your colour scheme: pick suitable colours for headings, text, background and links (all states: visited, unvisited etc.).

Finally, you are ready to start coding! All this preparation should make the next phase very straightforward.

Construction
With a solid plan beside you, you can start to markup a template (or prototype) page. This will be a single page in your site and will act as the basic template for your other pages. Establish the structure of the page and clearly comment where your navigation, main content, page heading, footer and so forth are marked up.

Use appropriate structural tags to describe your content. Put each

paragraph of text in <p> tags, put headings in the appropriate level of heading tag and so forth.

Ensure your page includes:
* an appropriate doctype
* character set information
* sensible, useful alt values

Now validate this template page.

When your markup is valid and you are satisfied with the general structure, you will evaluate your template. Return to construction once you have perfected this initial page.

Evaluation
Evaluation requires you to cast a critical eye over your work. Do this after completing your template page and before writing any more pages so that you don't duplicate problems through your entire site.

Ask yourself the following questions:
* Does everything work?
* Do my chosen colours contrast well? (Remember to check all link states.)
* Are the different link states distinguishable?
* Is all text easy to read? (Consider font, size, style and colour.)
* Does window resizing adversely affect the page layout?
* Does the page work as intended in the required browsers?
* Is the content accessible without images?
* Is the navigation easy to use and understand?
* Can the user determine where they are in the site structure?

Remember that your page is understandable to you because you created it but it isn't for you, it is for your users. Imagine you are a typical web user when evaluating your page. Be critical: consider how you can improve the experience of using your site.

A web site doesn't have to be 'pretty' to be useable. Focus on usability first and steer clear of 'party tricks' that you think will make your page look cool. That flashing, animated gif of a cutesy bunny won't enhance any users experience and is almost guaranteed to annoy them instead. Annoyed users don't stay long.

When you are satisfied with both the construction and design of your template page it is time to return to the construction phase to complete the rest of your web site.

Back to construction
Always validate your markup after any change. This is particulerly important before you duplicate your template otherwise you will copy any mistakes through every page in your site.

Duplicate your template page so you can produce the other pages for your site. Validate each one when you have finished adding its content to ensure you haven't introduced any mistakes.

Your markup should be easily maintainable. Appropriate use of comments and clear markup layout will assist this greatly.

When you have your completed web site, evaluate it again to consider it as a whole.

Evaluate again
The evaluation of the site as a whole should focus more on its behaviour as a unit than on the details of page layout, which you have already thoroughly considered when designing the template page.

Ensure that your site works as intended:
* Test every link.

```
* Check that each page's position in the site structure is obvious to the
user.
* Make sure all images display.
* Make sure all special characters display properly.
* Make sure alt values for all images are descriptive and sensible - view
your site without images to test this performance.
* Test the site in different browsers.
* Read your entire site to ensure you haven't introduced any mistakes.
* Test all user input aspects of your site.

Remember: you created your site so you understand it, but you aren't the
intended audience of your site. You must consider how a first time user
will cope with the experience you are providing in order for your site to
be successful.
```