Edinburgh Napier University

SET08101 Web Tech

Lab 1 - Introduction & Learning Environment

Workbook developed by Dr Simon Wells

1 Aims

During the practical portion of this topic you will:

- Choose and run an editor (in which you'll create your HTML, CSS, & JS files during the rest of the trimester.
- Write a simple HTML document; your first web page.
- Use the Chrome developer tools to support your web development journey.
- Use Git to store the files that make up your website.
- Use GitHub pages to publish your web pages as a real web sites for all the world to see.

This is an introductory lab that is designed to ease ourselves back into things after the Winter break. It's also an important lab for getting ourselves set up for the rest of the trimester. We'll create our learning environment, comprising a text editor, Chrome, Documentation, & Git, and we'll use that to create a webpage, first on our local computer, and then, via GitHub, on the Web as a published, albeit simple, website. You might have done some web development before, in which case this week might seem quite straightforward, but this lab is designed to achieve two goals, firstly, to get everyone starting in the same place, and secondly, to get you to build and publish your first simple web site in the first week of the trimester. After this, we're mostly extending and refining what we've done this week. So it is important not to skip any steps from this week's lab.

Remember that the lab sheet is a starting place. Most lab sheets will cover some core skills but they are meant to form the starting place for your week's worth of self-directed study, not just 2 hours of work during the timetabled lab. You can probably get through all of the core activities for a given lab sheet in 2 hours, but you are unlikely to have completed all of the challenges because they are designed to be more open ended and lead both to reflection on learning, but also to opportunities to flesh-out and practise your skills.

2 Activities

2.1 Get some tools: A Text Editor & a Web Browser

All we really need for front-end development is a decent text editor and a web browser. Nearly all programming involves working with text, and web programming is no exception. So the most important tool we can add to our toolbox is a good text editor. If you make a good choice then your text editor will be the place that you spend a lot of time. Obviously as things get more complicated we will extend our toolbox, but an editor is a good place to start.

If you already have a favourite text editor (not a word processor) then feel free to use that. As a developer you will write and edit a lot of text so your editor, whether stand alone or part of an integrated development environment, is an important tool. It's a good idea to become familiar with the features that your chosen editor offers and to practise using them.

If you don't already have a favourite editor (and you are on Windows) then Notepad++[1] is a good place to start[2]. See what is in Apps Anywhere if you're on a lab machine. Many of these editors are also either multi-platform or cross-platform so you should be able to run it on your operating system of choice.

Whichever you choose, download it, install it, and run it. Note that NotePad++ can even run from a USB stick so that you can take your development environment around with you to different machines.

[1]https://notepad-plus-plus.org/

[2]Note that there are many decent editors available, Vim is an oldie but goodie (and my personal choice although quite idiosyncratic and complex) but more recently Atom https://atom.io/, Sublime http://www.sublimetext.com/, and VisualStudioCode have gained in popularity as well

Lab machines should already have the Chrome browser installed and ready to run. However, if you're on your own laptop then you might need to install it. This way we can all have a consistent browser experience across the class. This is quite important when it comes to marking of assignments as there can be differences between the various browsers and, at least for this module, we probably don't want to spend too long exploring the thorny problem of cross-browser compatibility.

2.2 Hello Web

Using your text editor, write a short HTML document called **index.html**. The contents of your HTML document should be as follows (but you can alter it and experiment if you desire to see what happens). e.g.

```
<!DOCTYPE html>
<html>
    <head>
        <title>SET08101 - Web Tech</title>
    </head>
    <body>
        <h1>Hello Web Tech</h1>
    </body>
</html>
```

You can open this page in a web browser, like Chrome, by double clicking the .html file which will then open in your machine's default browser. On lab machines the default may be Edge. Alternatively, if you have a browser window open, then you can drag and drop the .html file onto an open browser tab.

It can also be useful to open your web page in a variety of browsers (whichever you have installed on the machine you are using). Have a look to see what browsers are installed on the lab machines and compare any differences in the page that is rendered. Consider why different browsers might display things differently.

Think about where this .html file is located, and why, although it is using "Web" technology, HTML, it is not yet really a Web Site.

2.3 Developer Tools

You will be using the developer tools built into Chrome a lot in this module, often to find out why things aren't running properly (or why the browser isn't doing what you're sure you told it to do). Partly this is because you will make mistakes. Partly because getting lots of different web technologies to work together is tricky. Things don't fit together just right and we need to work out why. We'll use Chrome to do this because:

- It runs on nearly every common platform
- It has developer tools built in
- It's a fairly stable platform to work with
- There is a lot of supporting documentation

Open your hello.html file in Chrome and view it using the developer tools. Try mangling your source file, e.g. removing some tags and compare your input file to the view in the developer tools. The browser seems to fix issues, why do you think this is? How mangled can your file be before it isn't displayed?

The developer tools have a lot of features and functionality. Visit the Chrome developer tools documentation[3] and read up on what it can do for you.

[3]https://developer.chrome.com/devtools and https://developers.google.com/web/tools/chrome-devtools/

2.4 Background Reading

Use your browser to visit http://link.springer.com. If you are on the university network you shouldn't have to log in, but otherwise your university credentials will work with the Shibboleth log in option on the site. Use the search option with the following query: "Algorithms and Data Structures", you might want to narrow the result by selecting the "Books" option in the menu on the lefthand side.

You can, and should, download PDF (or EPUB) copies of the following books which will supplement your reading throughout the module.

- Introducing Web Development
- Practical Web Design for Absolute Beginners
- Moving to Responsive Web Design
- Beginning Responsive Web Design with HTML 5 and CSS3
- Sustainable Web Ecosystem Design

Pointers to other texts, chapters, and papers will be suggested throughout the module, but these should be enough to get started and provide a solid underpinning for your learning. The main idea with the background reading is to give you a set of resources that you can use to expand your knowledge and fill in the gaps. You'll find that many books will cover the same ideas but because the different authors say things in different ways, the authors have different voices, you might respond more positively, or get more from one author as opposed to another.

2.5 Use Git

Git is a source control system that enables you to keep track of your source code, its history and any changes you make. Git can be used to track any file but is most efficient and best suited when used only with textual files. Because Git is a *distributed* source control system it works very well to enable groups of people to work on the same source code as well as supporting experimenting with your code, trying out lots of different ideas in separate *branches* (which are a bit like a copy of your code but with tools to help manage that copy and support re-integrating it with your main source tree if you want to), and being able to roll back to an earlier version if you decide you have take a wrong turn.

I'm not going to give detailed instrutions for setting up Git on the lab machines, that's why this is a bit of a *challenge*. However, a good starting place is my Git Quickstart document: https://www.dropbox.com/s/2kz34u0zb4qajvd/getting.started.pdf?dl=1

IMPORTANT Git will be a requirement for submission of the coursework hand-in so should spend some time getting familiar with it as soon as possible.

A good place to start to build your knowledge of Git is by dipping into the Git SCM book[4].

There are also numerous interactive tutorials and resources to help you get started with Git:

1. Github's Learn Git 15 minute tutorial: https://try.github.io/levels/1/challenges/1
2. Learn Git Branching http://pcottle.github.io/learnGitBranching/
3. Git Immersion http://gitimmersion.com/lab_01.html
4. A web search for "Introductory Git Tutorial" will lead you to hundreds of tutorials that go over the basics of Git usage.

[4]https://git-scm.com/book/en/v2

You should also create a Github account[5]. Once your account is created, you should create a repository within your new account called 'set08101'. You can then push all of your code throughout the module into this repository. I'd suggest adding a folder for today's lab and committing your hello.html file.

Using Git counts as practising a professional skill. The advantage of this approach is that at any point, if you need help with your code, then you have a copy that can be shared remotely, e.g. with Simon or another member of the teaching team. However, this only works if you keep adding your code to your repository. That means whenever you make changes you need to (1) add them, (2) commit them with an explanatory message, and (3) push the changes from your local repository to the shared one on Github or Bitbucket.

2.6 GitHub Pages

One of the most valuable advantages that Git and GitHub give us in this module is the GitHub pages feature. This was designed to enable open-source software projects to easily create new websites for this project pages and documentation. However, this is also a very straightforward way to get free and reliable Website hosting as it is basically a traditional, static, web server; just what we need for this module. GitHub pages can be enabled for any Git repository stored on GitHub. All that is needed is for the repository to contain at least one .html page, of which one must be named **index.html** (all lowercase). This is the default file, the **home** page or **index** page, that GitHub pages will serve to any browser that requests the site. Then there is a single setting that needs to be enabled in the settings of the individual repository. Each repository can have a a different website associated with it, or none. The pages that make up the site are stored inside the repository.

I have an example, simple "Choose Your Own Adventure" style game website[6] and the source-files for the site are in GitHub[7] I've also recorded a short screencast to show the process of enabling GitHub pages for a given site[8].

Github Pages isn't the only way to publish a website to the Web. It just happens to be the method that we'll use for this module. We'll consider this a little more in the deployment topic later in the module. However, for now, be aware that there are many Web **hosting** providers, many of which offer free tier usage for small, low-traffic, static sites. There are also hosting providers to suit almost any budget. These will all enable you to upload the pages that make up your site to their Web server. The specific mechanism for transferrence of the site to the server can vary though. Git is an increasingly popular tool, but secure copy (SCP) and file transfer protocol (FTP) tools are also used frequently. If you're comfortable administrating a server then you can also run your own, physical or virtual, Web server, rather than using a third party hosting solution. Running your own server gives you maximum control over features and functionality, but requires a hefty investment of your own knowledge and skills to do well.

2.7 Challenge

If you've successfully completed all the sections of this lab sheet then you should have a live site running on the Web that is hosted by GitHub pages. **The challenge is merely to email the URL for your deployed Web page to your module leader (Rachel Salzano[9]).**

Having done this once, you should see how straightforward it is to create and publish a web page, so you should be able to do this as needed in future, and each time should be easier. We'll exploit this in earnest for the assignment hand-in, but for now, bask in the knowledge that you've just added a new, albeit possibly not very interesting, site to the Web.

[5]https://github.com/
[6]https://siwells.github.io/cyoag/
[7]https://github.com/siwells/cyoag
[8]https://napier.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=9bcf640e-326c-4249-92fc-acd101390572
9r.salzano@napier.ac.uk