CSY1018
Web Development
Topic 6

Tom Butler
thomas.butler@northampton.ac.uk

Topic 6

- Version control with Git
- What is version control?
- Basic usage
- Branches
- Merging

Workflow

- Development on a live website is a bad idea
- Although it’s possible to edit the HTML/CSS *live* on the website, users will see your changes as you make them
- If something breaks, real people using the website will see it. This is incredibly unprofessional!

Workflow

- Instead, you should run a copy of the live website on your own computer
- This has a lot of benefits:
  - Speed: You don’t need to upload files to a website to see the changes, just save the file and refresh the page
  - You can break things and it doesn’t matter, if a page stops working, it doesn’t matter, nobody other than you will see it
  - You can test different approaches

Workflow

- Once you have made all the changes you want, you can then upload them to the real website
- Using this methodology you have copies of the code in two different places

Workflow

- Each time you make a change to your local copy it needs uploading to the server
  - More on this later!

Version Control

- When developing software you need to frequently make changes to it
- Sometimes you want to completely replace functionality in the code
-

Version Control

- One technique for managing this is that you can make a backup of the original file e.g.
- Back up index.html by copying it to index2.html
- Work on index.html
- If needed restore index2.html

Version Control

- On larger projects this can be very difficult to manage
- Especially when multiple people want to work on the code

Versions

- During development you'll usually have two versions of any given code:
  - The last known “good configuration” where everything works as intended and is currently running online with real visitors
  - The “in-development” version that may contain missing code and things that need to be fixed and is running on your computer only

Versions

- In real projects you end up with different scale tasks:
  - Quick updates (e.g. changing some text on the website)
  - New developments (E.g. adding new features, e.g.a shopping cart)

Workflow

- In industry, clients will understand the difference between these requests
  - Some big developments can take weeks or months and will be deployed (made live!) when it’s ready
  - Some changes need to be done quickly

Making changes

Live version

Development version

```
<main>
<p>Welcome to our website.</p>

</main>
```

```
<main>
<p>Welcome to our website.</p>

<p><a href="newproduct.html">Click here to see our new product launch!</a></p>

</main>
```

Making changes

- Here the development version has some unfinished code that's not ready go go live (the new product page is not ready yet!)
- However, the client might want you to urgently add a notice to the home page

Making changes

Client Request

```
<main>
<p>Welcome to our website.</p>


</main>
<aside>
<h2>Closed today!</h2>
<p>Please note: Due to heavy snow we will be unable to open today.
You can still contact us by email if you have any enquiries</p>
</aside>
```

Development version

```
<main>
<p>Welcome to our website.</p>

<p><a href="newproduct.html">Click here to see our new product launch!</a></p>

</main>
```

Making changes

- To make this change you need to:
  - Back up your changes to the file e.g. rename it index2.html
  - Download the original index.html
  - Make the quick change to the file
  - Upload the new file
  - Restore your changes (put index2.html back to index.html in your development version)
  - Make the same change to your development version

Making changes

- This can be incredibly difficult to manage in real projects, especially when multiple people are working on the same code
  - Which happens with most software!
- You end up with several versions of the file and when you're ready to 'go live' with a feature manually applying the changes to the most recent version of the file
- This is both time consuming and difficult
- The more frequently things change, the bigger problem this becomes

Version control software

- To overcome these problems, software companies use version control software to solve these problems
- There are several different tools which do this job
- Subversion (SVN) was popular in the early 2000s however it's use has heavily diminished

Version Control Software

- In recent years most companies now use a program called Git
- Git was developed by the developer of the Linux Kernel to solve version related issues where hundreds of developers were working on the same code at any one time
- Its more advanced features can be overkill for small projects, however it is still incredibly useful even on tiny projects

GitHub

- Knowing how to use Git is increasingly more important for software developers due to GitHub
- GitHub is a code hosting platform that allows you to manage code via Git
- The popularity of GitHub has helped Git become the standard Version Control Software in use today
- Most open source projects are hosted on GitHub
- If you want to work on an open source project or release your own code on GitHub you'll need to understand how to use Git

Git

- Git can be confusing at first
- Git is command line driven but there are GUIs available (Some are better than others!)
- It's useful to learn the command line to see what is actually happening behind the scenes but you can get a basic understanding through a GUI
- VS Code Contains a relatively simple GUI

Git - Basics

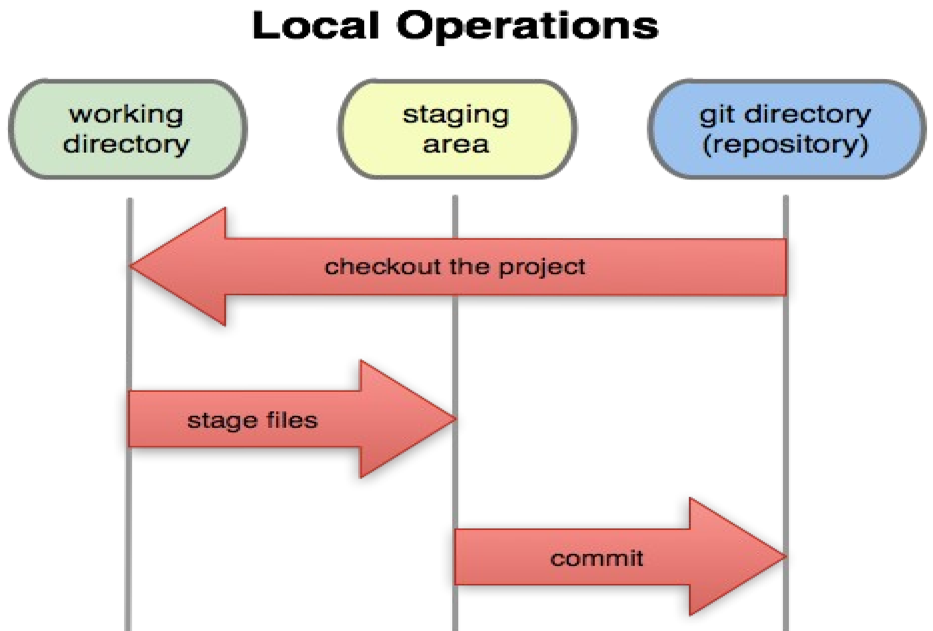
- Each set of code e.g. a website or an individual project is placed in a *git repository* (or `repo` for short)
- This is a directory that contains all the projects files
- In our case, the *repo* will be the folder that contains your HTML and CSS files

Git – Basics

- When you add files to a repository you must be identified as a user
- This is so when you look through a list of changes to the code, you can see who made the change
- Using powershell (in any location) You must run 2 commands before saving any files to the repository:
  - git config --global user.email “your@email.com“
  - git config --global user.name “Your name”
- With your name/email inside the quotes

Git statuses

- There are 3 types of file in git:
  - Unstaged files that have changed, but will not be committed
  - Staged (will be added to the repository)
  - Committed (Currently stored inside the repository)



Adding files

- “Committing” a file (or group of files!) saves a snapshot of those files as they are at the current point in time
- Each “commit” is a snapshot of how the project looks at a particular date/time
- You can then go back to different snapshots
  - (In theory… there is actually a better way to manage this)
- It is useful to be able to see file histories

Instructions for using git..

- Please see the accompanying video

Exercises

- 1. Install git (https://git-scm.com)
- 2. See slide 23. Open powershell and run the two commands to set your name and email address
- 3. Initialise a git repository in your website folder from topic 4
- 4. Commit your initial project
- 5. Add a new menu item and commit it
- 6. Try reverting the commit using `git revert` (you will need to open terminal using Ctrl+`)

Exercise 2

- 1. From the master branch create a new branch called “faqs menu item”
- 2. Add a new menu item called FAQs which links to faqs.html (you don’t have to create this file, just a link to it)
- 3. Commit the change
- 4. Switch between the master and “faqs” branches so you can see the different versions
-

Exercise 2

- 5. Switch to the master branch (without the FAQs link)
- 6. Create a new branch called “Footer”
- 7. Amend the text in the footer to include your name and a link to the contact.html (You don’t have to create the file)
- 8. Commit your changes
- 9. Switch between all three branches so you can see how it works
- 10. Switch back to master
- 11. Merge the changes from the other two branches into the master branch