CSYM019
Internet Programming

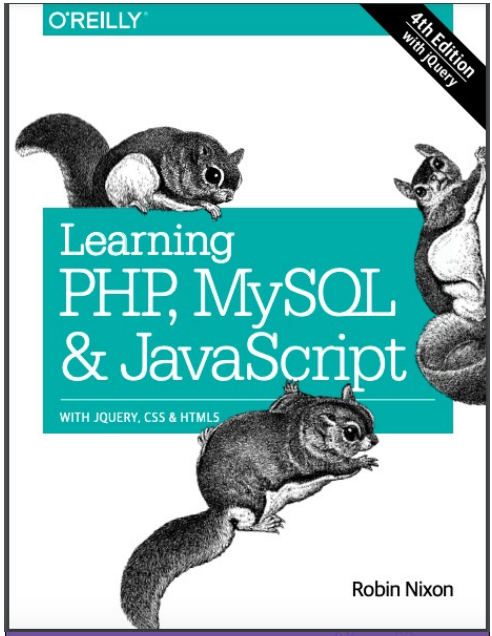
Muawya Eldaw
Muawya.eldaw@northampton.ac.uk

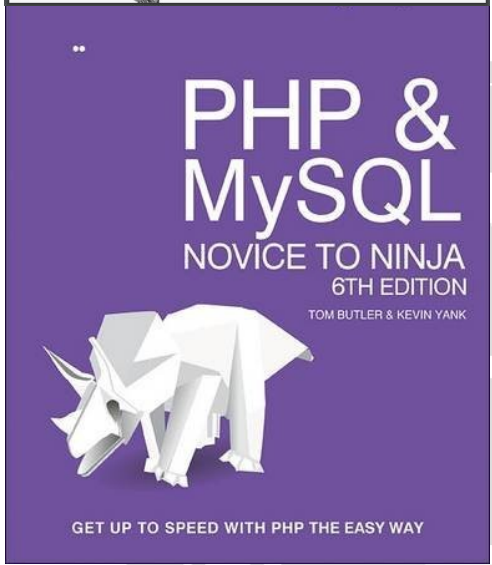
Week 1 - Introduction

- About me
- How the sessions work
- Introduction to HTML

About me

- Dr Muawya Eldaw
- Received PhD from Birkbeck, University of London, 2019
- Worked as a software developer since 1999
- Been teaching for 7 years
- Publish frequently in peer-reviewed conference and journals:
  - Google Scholar Webpage: https://scholar.google.co.uk/citations?...





CSYM019 Sessions

- 2 Hour sessions
- **1 hour optional drop-in support session**
- Will stop for exercises frequently so you can run the code for yourself
- Exercises will take between 5 minutes and 30 minutes
- We may not cover a whole topic in a week, some topics will last more than one week
- Each week we will continue where we left off the week before

Internet Programming

- Internet PROGRAMMING requires coding
- There are a lot different technologies which are used on the web
- We will cover 4 of them:
  - HTML (the basis for all web pages)
  - CSS (used to describe how pages look)
  - JavaScript (used to add interactivity to pages)
  - PHP (Server side programming e.g. posing messages to a website, placing orders, uploading images to a server)

General Advice for programmers

It Takes 10,000 Hours to Become An Expert

- The science behind this statement is questionable but the sentiment is correct
- However, in programming a little knowledge goes a long way
- The only way to get good at something is practice
- You won’t learn how to code overnight (Including the night before the assignment hand in)

Resist the temptation to rush ahead

- Programming concepts build on one another
- If you get stuck, it’s often because you don’t fully understand an earlier concept
- It’s usually quicker to go back and try the exercises from the previous week

The Concorde Fallacy

- In the 1970s the British and French governments spend billions on the Concorde aircraft
  - They kept spending money despite huge financial losses
  - Their reasoning was that they would lose all their investment if they pulled out
  - In the end, they lost even more money because they kept funding the project
- **Never be afraid to scrap everything and start over! It’s often quicker in the long run**

You’re not learning JavaScript/PHP/Python/Etc

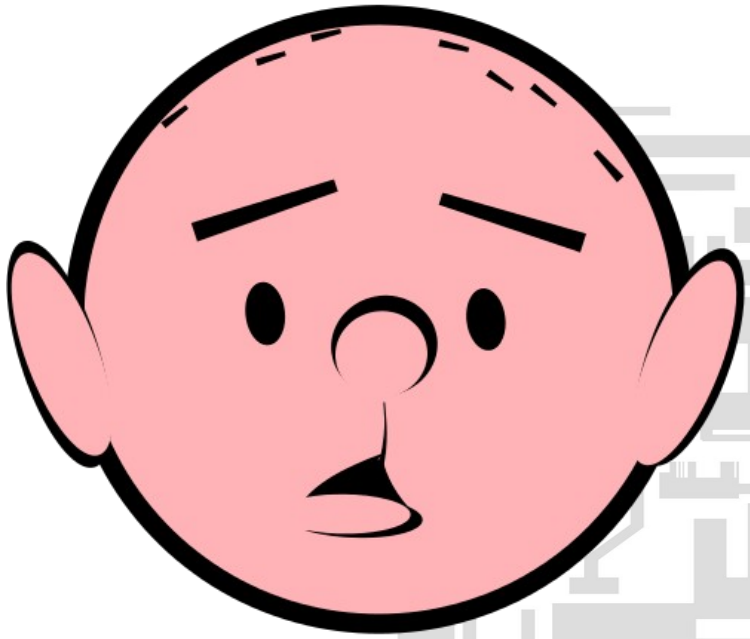
- Most languages share the same concepts
- The concepts will translate to most other programming languages
- Don’t fall into the trap of thinking “I am learning JavaScript” instead think “I am learning to code”
- Once you are proficient in one programming language, you can get to a reasonable standard in another within a few days!
- **The concepts are more important than the syntax**

Getting Braces and Semicolons in the Right Place Is the Easy Part

- At first you'll struggle to get braces, semicolons, brackets, etc in the right place
- The difficult part is getting the logic right
  - The computer will tell you if the syntax is wrong. It's either right or wrong. It works or it doesn't.
  - The computer can't tell you whether you've given it the correct instructions to solve the problem at hand

“You won’t get anything done by planning”

- Karl Pilkington



There are three fundamental steps you should perform when you have a program to write:

1. Define the output and data flows.
2. Develop the logic to get to that output.
3. Write the program.

Notice that writing the program is the last step in writing the program. This is not as silly as it sounds. Remember that physically building the house is the last stage of building the house; proper planning is critical before any actual building can start. You will find that actually writing and typing in the lines of the program is one of the easiest parts of the programming process. If your design is well thought out, the program practically writes itself; typing it in becomes almost an afterthought to the whole process.

Sam’s Teach Yourself Beginning Programming in 24 Hours

How these sessions will work

- Each week there will be around 1 hour of lecture and 1 hour of practical time
- I will give a short introduction and we will stop to do exercises at various points

Learning tips

- If you miss a session make sure you read the notes and try the exercises
- You'll learn more by trying the exercises than just reading the notes
- Everything we cover will be useful for the assignment
- Focus on learning the concepts being taught, not the individual exercises
- Make sure you complete the exercises each week
- No solutions will be available but everything you need to complete the exercises is covered in the notes
- Put your hand up and ask questions at any time

What I expect you to already know

- Basic computer usage:
  - Navigating around the web
  - Using your operating system: Installing programs, navigating around
  - Using your file manager to locate/manage files
  - Saving files (and finding them again!)
  - Opening zip files
  - How to open the same file in different programs
  - Looking things up
    - Don't ask me questions you can type directly into google! "How do I do a loop in javascript?"

HTML

- HTML stands for:
  - Hypertext Markup Language
- It is made up of HTML *tags*
- A tag is a description of some contained data
- Each tag uses angle brackets: < and >
- Most HTML tags have a start and end tag

Browsers

- HTML code is run in a *web browser*
- You’re all familiar with web browsers:
  - Google Chrome
  - Mozilla Firefox
  - Edge
  - Internet Explorer

HTML

- HTML looks something like this:

HTML basics

- The web browser uses the tags to determine how to handle the containing data
- Only the data between the tags is displayed to the person who views the page

```
<tag-type>Contents</tag-type>
```

HTML tags

- Different tags have different meaning to the browser
- Example tags:
- <p> For a paragraph
- <h1> for a top level heading
- List item

HTML

- A HTML file can look like this:

```
<h1>Page heading</h1>
<p>Page content</p>
```

HTML

- HTML is a *nested* structure.
- You can put one tag inside another
- For example, a paragraph inside a list item:

```
<li>
  <p>Page content</p>
</li>
```

HTML

- Most tags can be *nested* inside other tags
- Every page should be wrapped in a <html> tag
- A <html> tag should contain two distinct sections:
  - <head> - Metadata used to tell the browser about the page
  - <body> - The contents of the page (things the user sees)
- Every HTML file should start with a DOCTYPE

HTML

- A basic valid HTML file looks like this:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>My Web Page!</title>
    <meta charset="UTF-8" />
  </head>

  <body>
    <h1>Page heading</h1>
    <p>Page content</p>
  </body>
</html>
```

Nothing in <head>
appears on the page, but the info is used elsewhere

HTML

- HTML is forgiving
  - You can make mistakes and the browser will try to fix it
- But you shouldn't rely on this behaviour!
  - Try to always write valid HTML
  - You can (and should!) check your HTML for errors using the W3C HTML Validator http://validator.w3.org/
    - Alternatively, most editors (e.g. Sublime) have plugins to do this inside the program which makes it trivial to do
  - If your page isn't displaying as you expect, it's probably has an error and the validator will help you find out how to fix it by telling you where to look (e.g. tag name or line number)

Before we start

- HTML and Javascript are text based languages
- They can be edited with *any* text editor
- It's better to use a text editor with features specifically for writing code
  - Do not use notepad!
- Notepad++ is installed but it's old and clunky

Editors

- Recommended text editors for HTML/JavaScript development:
  - Visual Studio Code (https://code.visualstudio.com/)
  - Sublime Text 3 (http://www.sublimetext.com/3)
  - Atom (https://atom.io/)
- All work on Windows/Linux/OSX
- **You can install Atom on the university machines without admin rights!**

HTML

- HTML is forgiving
  - You can make mistakes and the browser will try to fix it
- But you shouldn't rely on this behaviour!
  - Try to always write valid HTML
- Make sure you save your files with a .html extension! e.g. week1.html

Validating your HTML

- **You can (and should!) check your HTML for errors using the W3C HTML Validator http://validator.w3.org/**
  - Alternatively, most editors (e.g. Sublime) have plugins to do this inside the program which makes it trivial to do
- If your page isn't displaying as you expect, it's probably has an error and the validator will help you find out how to fix it by telling you where to look (e.g. tag name or line number)

Doctypes

- The Doctype is a special instruction at the top of the HTML page which starts with:

```
<!DOCTYPE
```

- There are lots of doctypes you can choose from:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
```

Doctypes

- However, the only one you should use is:

```
<!DOCTYPE html>
```

- This is simpler, understood by all browsers and the standard for HTML5
- If you come across any example code that uses a different doctype (mentioning XHTML or HTML4), the article you are looking at is probably over ten years old, find a better more recent example!

HTML has changed

- Web browsers change all the time, new features are supported, things can be achieved in different ways
- Old code will work but will usually be more complicated!
- When looking for information on HTML, CSS and Javascript, make sure you find information that is up to date
- **Look at article dates, anything older than 4-5 years should be avoided, find something newer**

Finding Help

- Do not use W3Schools!
- W3Schools is **not affiliated with the w3c although they like to pretend they are**
- W3Schools often has outdated or plain wrong information

Finding help

- Better resources include:
  - https://developer.mozilla.org/en-US/ (Run by the people who make the Firefox Web Browser)
  - http://eloquentjavascript.net/ (For Javascript – Very in depth explanations of the basic concepts)
  - https://www.codecademy.com/ (Has interactive demos and tests)

Exercises

- 1. Use one of the installed editors (Atom or VS Code) to create a HTML file with some basic information:
  - Heading: CSYM019 – Internet Programming
  - Paragraph:
  - This module focuses on creating client and server side software as well as web applications for the World Wide Web us. It concentrates on the technologies used to allow such software to be designed implemented and deployed.
  - List:
    - Field: Computing
    - Level: MSc
    - Lecturer: Muawya Eldaw
    - Day: Tuesday
    - Time: 11:00
- **Note: You will need to research how to create a list, you need more than li tags**
- **Remember to set a relevant <title>**
- **Remember to save your files with a .html extension!**

- 2. Open your page in a browser and check it looks ok
- 3. Validate the page using validator.w3.org if it’s not valid, tweak the code until it is

  Remember to validate the page after you have fixed the issues it highlights!

- 3. Create a second page for one of your other modules (Details available here: https://www.northampton.ac.uk/awards/modules/COMPUTER-SYSTEMS/7/
- 4. Add a third module and link it from your other pages
  - Remember to validate all your pages!
- 5. Create a list of links at the top of the page to act as navigation.
  - You will need to research how to link from one page to another
  - Put the links in an *unordered list* ( )

- 6. Test out different HTML tags
  - A good resource is https://developer.mozilla.org/en-US/docs/Web/HTML
  - Click “HTML Elements”
  - Play around with some of the tags to see what they do
- 7. Use your new tags to change the look of your web pages.
- 8. Compare your results to others in the class and see how different your designs are
- 9. Try adding different pages on your own topics e.g. about you, your favourite team, film, etc