Beginners PHP Programming

About the course

Welcome to the course! This course aims to teach attendees basic programming concepts, using the widely available PHP language and MySQL database server. Programming can be used for a lot of things, but we'll focus just on one: building applications for the web. The skills you will learn will teach you to think like a programmer, and so will be very useful if you decide to learn another language.

The course will be hands-on, so bring a laptop (netbooks are okay but might be impractical for the amount of typing we'll be doing). The software we will install will run on pretty much any operating system (if you wish to install before your first lesson, WAMP for Windows or MAMP for Macs is suitable – Linux users probably have everything they need already). You may need to ensure you have administrator rights on the laptop (if it is a work computer, it may be locked down so software cannot be installed).

About the format

We'll have a number of objectives to try each week, which we'll work on together. The material will roughly run in order, so we can later look at complex topics that draw earlier, simple ones together. However, occasionally we might need to make use of something we've not yet learnt; this is okay, as we'll cover everything eventually.

If you miss a week, you may find it helpful to refer to the notes and do some internet research to cover the objectives for that week. We can always use some time after class for catch-up sessions if necessary. Get in touch by email throughout the week if you would like to see how far we got in a particular week.

Attendees are welcome to ask questions at any time, and helping each other is encouraged. If some feature of the course could be changed to make it easier for you to learn, let me know. While these notes will form a rough plan for each week, if attendees wish to cover other material, we'll try to cover that also.

The main aim of this course is that it will be interesting, fun and encourage you to experiment with technology. As with many things, there is no end point where one can stop learning – if you can find the time for additional hacking, there's plenty of tutorials on the web.

Code examples

There are a lot of exercises in this course, as the process of typing can really aid the memorisation of the material. In each case, a suitable file name is suggested, so that items are easy to look up. This will be helpful on the occasions when one exercise builds upon another one.

Introduction

Objectives:

- Appreciation of the limitless, exciting creativity of building for the web
- Be introduced to the permanent learning experience of being a programmer
- Understanding the client-server model used by the web
- Know a little about PHP, in particular that: it is easy to learn, and so is suitable for beginners; it is somewhat "designed by committee" and so has a number of inconsistencies and flaws; it is arguably one of the easiest languages to deploy; there is a large amount of learning material available for it
- Be aware that many languages are used on web servers, and that they all have their strengths and weaknesses

Resources:

- The official manual is at: php.net/manual
- Good history and introduction at Wikipedia: https://en.wikipedia.org/wiki/PHP

Installation

Objectives:

- Install a web server (Apache) and a database server (MySQL), perhaps using an all-in-one installer (e.g. WAMP for Windows)
- Know how to start and stop servers
- Ensure PHP is available on the console, by obtaining the version (-v) and examining the loaded modules (-m)
- Check PHP is working in the browser, by running a `phpinfo()` script

Resources:

- Download this and install if you're on Windows: wampserver.com/en/
- Download the free product here if you're on a Mac: mamp.info/en/

Introduction to code

Objectives:

- Understand that PHP programs can be written in a simple text-editor, but that word-processors and desktop publishing packages are generally unsuitable
- Know that programs are executed line by line, in order, except where statements are used to change the order
- Understand the purpose of the open tag `<?php`, the close tag `?>`, and the semi-colon line separator
- Look at a sample of PHP code from the internet, and see that braces `{}` are used a lot

- Be aware that indentation and commenting are valuable aids to readability, and that getting this right from the start of a project will save time later on

Resources:

- Here's some publicly available PHP code written by Facebook engineers: https://github.com/facebook/facebook-php-sdk/blob/master/src/facebook.php

The console

Objectives:

- Write a simple program that runs on the command line. The program can be as simple as:

```
<?php
echo "Hello world!\n";
?>
```

- Be aware that console programs can be run on a timer
- Learn that a server console is usually only available to server administrators
- Modify the above program so that it also outputs the number 42, followed by a newline
- Appreciate the difference between echoing strings and numbers

The web server

Objectives:

- Understand the purpose of the web server, in basic terms
- Know there are several different "brands" of server, that Apache httpd is by far the most popular, powering around 52% of the web, followed by Microsoft IIS at 20%
- Understanding that HTML documents are rendered by the browser
- Create a simple HTML document (`simple.html`) and load it into the browser via the file system
- Load the HTML document via the web server (and see that it looks the same)
- Create a simple PHP script (`simple.php`) and load it via the file system (and understand why raw PHP code is seen)
- Load the PHP script via the web server (and see that it has been processed into HTML)
- Understand that the most dynamic web pages are rendered using a process similar to this, regardless of the programming language used

Simple HTML

Objectives:

- Understand that HTML is a document description language, rather than a programming language, and that it is often referred to as "markup" rather than "code"

- Be aware that HTML documents have a doctype, and that all new applications should generally use the HTML5 type (i.e. `<!DOCTYPE html>`)
- Be conversant with some simple types: `<html>`, `<head>`, `<title>`, `<body>`, `<h1>`, `<p>`, `<a href>` and `<br />`.
- Appreciate HTML as being comprised of blocks nested inside each other
- Write an HTML document that makes use of all of these tags, and display it in a web browser

Resources:

- Here's a resource for learning HTML5: html5iseasy.com/

Variables and types

Objectives:

- Understand the purpose of a variable in a computer program
- Be aware what characters are considered valid for a variable name
- Be aware that there is a string type and two numeric types, integer and float, for whole and fractional numbers respectively
- Know that most types in PHP are not explicitly stated, and that this is known as “weakly typed” (in contrast to “strongly typed” languages)
- Understand how to join strings together using the dot operator (“concatenation”)
- Write a program (`variables_hello.php`) to store your name and render the following string in an HTML document or a console program:

  ```
  Hello <name>, how are you?
  ```

- Write a program (`variables_calc.php`) to store three numbers (either integers or floats) in variables, store the result of a mathematical operation of your choice that involves them all, output all the numbers, and the result. Example:

  ```
  My calculation is 1.5 + 4 / 2 and the result is 3.5
  ```

- Modify `variables_calc.php` so that, rather than using the dot operator, you use some inline variables

The array type

Objectives:

- Understand the purpose of an array as being a list of variables, and that it can be used for a broad range of list types, such as menu entries, shopping cart contents, database results etc.
- Be able to explain the difference between an associative and an indexed array
- Write a program (`array_set.php`) to insert a set of items into specific array positions (e.g. four kinds of fruit, five types of weather, or whatever set you choose)
- Modify the last program to use the [] operator, to add items to the end of an array

- Understand the use of debugging commands `print_r()` and `var_dump()` to see what is inside an array
- Write a program (`array_assoc.php`) to insert a set of key-value pairs into an associative array (e.g. the attributes of a bicycle, such as [`colour => silver, frame => mountain bike, brakes => hydraulic, wheels => 2`], or whatever topic you like)

Further reading:

- Reference manual for the array type: php.net/array

Conditionals

Objectives:

- Understand how `if` constructs are used to help the computer make decisions
- Write a web program (`if.php`) to read a query string called "name" via the `$_GET` array (we'll learn much more about these later on)
- Modify the program to test whether the "name" parameter is equal to a particular string value (such as your own name), using the equality `==` operator
- Understand that string comparisons are case-sensitive
- Modify the program to accept a number value, such as "number_of_cats_owned", and perform a numeric equality comparison
- Add in an "else" clause to one or both of your `if` statements

For loops

Objectives:

- Understand how to loop through a section of code, and that each loop run is known as an "iteration"
- Write a web-based program (`for_magic.php`) to echo a magic word of your choosing three times, using a for loop, from one to three
- Modify your program so that the magic word indicates which iteration count it is on, for example:

```
Hocus pocus number 1
```

- Be aware of the tradition of using loop variables starting from `$i`, but don't use them if a clearer name would be better

More HTML

Objectives:

- Understand the difference between a tag and an attribute
- Learn the purpose of `<strong>`, `<em>`, `<ul>`, `<ol>` and `<li>`
- Take your `for_magic.php` program, and modify it so that it outputs your magic words in a bullet-pointed list

- Understand that all static and dynamically generated HTML documents should pass W3C validation, but that in practice programmers don't often check!
- Submit your web-based program and paste it into the official W3C validator, and fix any problems that you find. You can copy and paste it, or use the upload feature

Resources:

- Here's the official online validator validator.w3.org
- Another HTML5 resource: html5doctor.com

Understanding errors

Objectives:

- Recall that the PHP may complain about a possible problem (a warning)
- Know that some problems are serious enough that execution cannot continue (a fatal error)
- Create a console program (`console_error.php`) that runs `error_reporting(E_ALL)` so all errors can be seen
- Add to this program (1) the echo of an undefined variable; (2) the echo of an indexed array element where the array is defined but the index is not; (3) the echo of an associative array element where the key is not correctly supplied as a string
- Run your program to see that warnings are “non-fatal”
- Add to the start of your program a for loop that says “Hello” five times, but deliberately omit the line terminator from your echo statement
- Run your program to see that syntax errors (and other fatal errors) will stop execution immediately

Thinking about the web

Objectives:

- Develop an awareness of constantly thinking about what problems might be solved online. Is there an untapped niche, or could something be done better?
- Suggest in class, or write down, some ideas for how you can harness the web. Ideas may come from professional experiences, personal interests and hobbies, anything
- Ideas may turn into a business, solve a social problem, be for the common good or be just to tinker with a technology
- Know that sketching and “wire-framing” can help stimulate ideas
- Simple hosting costs a cup of coffee every month, so hobbyists can afford a “play box”

Foreach loops

Objectives:

- Understand `foreach` loops, in their simplest form, as a loop that iterates over an array

- Copy your `array_set.php` program as `foreach_set.php` and add a `foreach` loop to echo each value from the array
- Modify this program so that the index is displayed in each iteration
- Copy your `array_assoc.php` program as `foreach_assoc.php`, and add a `foreach` loop to each key-value pair from the array

Resources:

- PHP manual: php.net/foreach

Introduction to forms

Objectives:

- Learn how to accept user input in a web application
- Understand the purpose of the `<form>` and `<input>` tags
- Know the purpose of the form `action` and the `method`
- Write a web program (`form_name.php`) containing a form, a text input called “name” and a submit button. The action should go to the same page, and the method should be “get”
- Learn how to use `isset` to test whether items are present in the `$_GET` array
- Modify your program so that, before the form, a paragraph before the form says “What is your name?” if they have not already answered, or “Hello there <name>” if they have answered

Using an IDE

Objectives:

- Learn the difference between text editors, programmers' editors and Integrated Development Environments (IDEs)
- Install the NetBeans IDE and create a project that points at the root folder of your web server
- Be aware that there are many editors of different kinds, and what you use is mainly a matter of personal taste
- Look at examples of syntax colouration, code folding and auto-completion
- Choose between using your new IDE (lots of features) or sticking with a text editor (nothing new to learn)

Resources:

- Download and install the PHP Netbeans from here: https://netbeans.org/downloads

While loops

Objectives:

- Understand the purpose of a `while` loop

- In addition to the `==` test from the `if` exercise, learn new comparison expressions `>`, `<`, `>=`, `<=` and `!`
- Write a web program (`while_loop.php`) featuring a `while` loop to count backwards from 10 to 1
- Modify your program so that one half of the bullet points are italicised (you choose which half). Remember that `<em>` is an inline tag, so needs to go inside `<li>`. Hint: you'll need two `if` statements for this

Resources:

- PHP manual: php.net/while

Introduction to algorithms

Objectives:

- Understand that an algorithm is a plan for a program, independent of programming language
- Using a text editor (or pencil and paper) note down how you would determine the largest value from a randomly-ordered array of numbers. Don't write your answer in code to start with – the idea here is to work out the plan, not what to type
- Write this program, calling the file `algorithm_max.php`. Include at the start an array containing around eight numeric values in a random order

User-defined functions

Objectives:

- Understand that blocks of code can be grouped into functions, which may have input values (parameters) and an output (a return value)
- See that using functions helps keep code organised (modularity) and means we can easily call it again without writing it again (code reuse)
- Modify your `algorithm_max.php` program so that your maximum algorithm is written as a function
- Modify this program so the function is called on an additional numeric array of your choosing

Reusing HTML

Objectives:

- Learn how blocks of HTML can be included in another using `require`
- Create a static web page, and call it `include_main_1.php`. Put in a paragraph of text to indicate that this is “File 1”, and do the same again for `include_main_2.php`, with a paragraph labelling it as “File 2”
- Add a file containing a snippet of HTML, called `include_menu.php`. In here, add links with suitable names to the two above files

- Add a suitable require PHP snippet prior to the paragraph in each “main” file, so that it all links together. You should now have a menu inserted at the top of each of your two pages

Introduction to CSS

Notes:

- Front-end design is not the aim of this course, so this section is deliberately simple. If you want to learn more about this, there's a great range of online and book resources

Objectives:

- Understand the purpose of CSS as “separating style from content” - content is easier to maintain, global style changes are trivial, and content is less cluttered with style attributes
- Learn how to create an inline stylesheet (using `style` with `type`)
- Discover the purpose of ids, classes, and CSS keywords `border`, `font-family`, `font-weight`, `font-style`, `color`, `background-color`, `margin` and `padding`
- Learn how to create an external stylesheet (using `link` with `href`) and understand why this is a good idea
- Modify `include_menu.php` so that it contains an inline stylesheet, and give the menu a red border, a pink background, and an emboldened font. Add some margin and padding too, so it is nicely spaced out
- Modify `include_menu.php` so that the inline stylesheet is converted to an external asset

Resources:

- A guide to stylesheets: htmlhelp.com/reference/css/

Style

Notes:

- Beginners should not worry too greatly about adhering to a style guide, but know that code consistency affects readability greatly in large projects

Objectives:

- Be aware that giving variables and functions a readable and meaningful name will make things easier as your programs get more complicated
- Remember to always comment code that needs explanation, but try to write your code so that it does not need explanation!
- Understand the different brace conventions (same line versus next)
- Understand the different indent conventions (2 versus 4 and tab versus space)
- There are several notable “style guides”, in particular PSR-1/2 and PEAR

Resources:

- The PSR guidelines: php-fig.org/psr/1/ and php-fig.org/psr/2/
- The PEAR guidelines: pear.php.net/manual/en/standards.php

Introduction to objects

Objectives:

- Know that classes contain functions (called “methods”) as well as variables (called “properties”), and that they help avoid “spaghetti programming”
- Be able to describe the difference between a class and an object
- Write a class `Animal` (`class_animal.php`) with public properties for these values:
    - what type of animal it is
    - the number of legs it has
    - its height in metres
- In a new file (`class_animal_demo.php`) instantiate the following, and give them realistic values:
    - `$fred` (a giraffe)
    - `$polly` (a human)
    - `$william` (a centipede)
- Be aware that properties may be set in an instance even if they are not declared, but that it is good practice to declare them
- Understand that for an instance to access its own methods and properties, we use `$this`
- Add to your class a method `sayHello` that says something like the following, with the placeholders replaced with instance values:

```
Hello! I am a <type>, I have <leg count> legs and I am
<height> metres tall
```

- Understand that there is no relationship between the variable name (e.g. `$polly`) and the instance contents, but that using meaningful names makes the code more readable

Introduction to MySQL

Objectives:

- Learn that the database server is like the web server – it just sits around waiting for remote commands
- Log onto your database console using your root credentials, thus (the command may vary from system to system):

```
mysql5 -u root -p
```

- Create a new user, a new database, and new privileges (substituting `your_name` and `your_db` with more suitable values):

```
CREATE USER your_name@localhost IDENTIFIED BY 'password';
CREATE DATABASE your_db;
GRANT ALL PRIVILEGES ON your_db.* TO your_name@localhost;
```

- Exit your console and log on again as your new user
- Run this code to see what databases you have access to:

```
SHOW DATABASES;
```

- Understand that data is stored in tables, which have columns and rows, like a spreadsheet
- Run this code to create a demonstration table and set up some data:

```
CREATE TABLE `test`.`animal` (
     `id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
     `name` VARCHAR( 100 ) NOT NULL,
     `type` VARCHAR( 50 ) NOT NULL
) ENGINE = InnoDB;
INSERT INTO animal (name, type) VALUES ('Fred', 'Giraffe');
INSERT INTO animal (name, type) VALUES ('Polly', 'Human');
INSERT INTO animal (name, type) VALUES ('William',
'Centipede');
```

- Finally, issue some commands to query the database:

```
SELECT * FROM animal;
SELECT name FROM animal WHERE type = 'Human';
SELECT name, type FROM animal WHERE type <> 'Human';
```