WEB 306 - PHP: Databases and Framework

Day 1 – Intro to Servers and PHP

1. Intro to Text Editors

1.1 Intro to VS Code

Visual Studio Code, or VS Code for short, is a customizable, modular, free and open-source text-editor created by **Microsoft**.

> **VS Code Resources**
>
> 1. **You can download it at code.visualstudio.com**
> 2. **View VS Code documentation at code.visualstudio.com/docs**
> 3. **View VS Code setup guide**
> 4. **If you have questions about VS Code then you can read the FAQ.**

1.1.1 Visual Studio Code Vs. Visual Studio IDE

Microsoft makes two different text editors with very similar names. One is called ***Visual Studio*** and the other is called ***Visual Studio Code***. Despite the similarity in names, the two editors are very different.

Technically, ***Visual Studio*** is considered an Integrated Development Environment, or IDE, which provides many advanced tools for debugging code and compiling programs. You could use Visual Studio for writing PHP code, but I would not recommend it. ***Visual Studio Code*** is a much more lightweight text editor which doesn't have the same number and complexity of features as Visual Studio but can be customized to add more features onto it. It is much more suitable for writing PHP code, in my opinion.

Microsoft explains the difference the following way:

> **Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.**
>
> **— From Visual Stuido Code Frequently Asked Questions**

1.1.2 Opening Projects and Creating Documents

To open a project in VS Code go to **File > Open Folder** or press **ctrl+K** and then **ctrl+O** and select a folder. To create a new file press **ctrl+N** and press **ctrl+shift+S** to save it as a specific file type. To reopen the last file you closed press **ctrl+shift+T**.

1.1.3 User Interface

When you open a new project the file Explorer will appear to the left of the document window. If the file Explorer is hidden, click on the first icon in the left-hand menu or press **ctrl+shift+E** to show the file Explorer. You can split the screen horizontally or vertically. To split the screen go to **View > Editor Layout**. VS Code opens files in tabs, just like your web browser. Press **ctrl+tab** and use the up and down arrows to cycle through your open file tabs. Press **alt+1** to **alt+9** to view the corresponding tab.

Press **ctrl+shift+P** to open the Command Palette. The Command Palette is used to enter a wide variety of special commands into VS Code to perform actions. These would be farily specific and you would look up a required command depending on the task you are trying to accomplish.

Editing Shortcuts

1. **Undo: ctrl+Z**
2. **Redo: ctrl+shift+Z OR ctrl+Y**
3. **Indent/outdent line: ctrl+] and ctrl+[**
4. **Move line up/down: alt+↑ and alt+↓**
5. **Copy line up/down: shift+alt+↑ and shift+alt+↓**
6. **Place multiple cursors: alt+click**
7. **Toggle line comment: ctrl+/**
8. **Toggle block comment: shift+alt+A**
9. **Find/Replace: ctrl+F**
10. **Toggle word wrap: alt+Z**

1.1.4 Extensions

Extensions can be installed to add extra features to VS Code. To view currently installed extensions and to install new extensions, click on the fifth icon in the left-hand menu or press **ctrl+shift+X**. Recommended VS Code extensions include `indent-rainbow` and `prettier`. The Emmet code abbreviation tool is installed in VS Code by default and does not require a separate extension. Read more about extensions here.

1.1.5 Themes

Themes can be used to change the look of VS Code. There are different types of themes which change different aspects of the design. Color Themes change the colors of the user interface and code highlighting while File Icon Themes change the look of the icons which are used to identify file types. To view currently installed Color Themes and to change your Color Theme go to **File > Preferences > Color Theme (Code > Preferences > Color Theme on macOS)**. To install new themes, search for and install a theme as an extension from the extension menu by clicking on the fifth icon in the left-hand menu or pressing **ctrl+shift+X**. Feel free to find a theme that you like, which matches your personal style! Read more about VS Code themes here.

1.2 Intro to Atom

Atom is a customizable, modular, free and open-source text-editor created by **GitHub** (more on them later).

1.2.1 Opening Projects and Creating Documents

To open a project in Atom go to **File > Open Folder** or press **ctrl+shift+O** and select a folder. To create a new file press **ctrl+N** and press **ctrl+shift+S** to save it as a specific file type. To

reopen the last file you closed press **ctrl+shift+T**.

1.2.2 User Interface

When you open a new project a file tree will appear to the left of the document window. Press **ctrl+** to hide and show the file tree. You can use panes to split the screen horizontally or vertically. To split a new pane go to **View > Panes**. Atom opens files in tabs, just like your web browser. Press **ctrl+pagedown** to view the next file tab and **ctrl+pageup** to view the previous one. Press **alt+1** to **alt+9** to view the corresponding tab.

1.2.3 Packages

Packages can be installed to add extra features to Atom. To view currently installed packages go to **File > Settings** and select **Packages**. To install new packages go to **File > Settings** and select **Install**. Recommended Atom packages include `indent-guide-improved`, `atom-beautify` and `language-batch`. The Emmet code abbreviation tool is not installed by default and requires the separate `emmet` package. Read more about packages here.

1.2.4 Themes

Themes can be used to change the look of Atom. To view currently installed themes go to **File > Settings** and select **Themes**. To install new themes go to **File > Settings** and select **Install**. Feel free to find a theme that you like, which matches your personal style!

Atom Resources

1. **You can download it at atom.io**
2. **View Atom documentation at atom.io/docs**
3. **You can read the Atom Flight Manual at flight-manual.atom.io**
4. **View Atom installation guide**
5. **Why use Atom? Here's an explanation**

Editing Shortcuts

1. **Undo: ctrl+Z**
2. **Redo: ctrl+shift+Z OR ctrl+Y**
3. **Indent/outdent line: ctrl+] and ctrl+[**
4. **Duplicate line: ctrl+shift+D**
5. **Place multiple cursors: ctrl+click**
6. **Toggle line comment: ctrl+/**
7. **Find/Replace: ctrl+F**

2. HTML

HTML stands for Hypertext Markup Language. HTML is a markup language. Markup languages are used to help computers and devices such as screen readers define and derive meaning from blocks of text.

> **A markup language is a computer language that uses tags to define elements within a document. It is human-readable, meaning markup files contain standard words, rather than typical programming syntax. While several markup languages exist, the two most popular are HTML and XML.**
>
> **— From Markup Language, TechTerms**

HTML specifically is used to define the structure and meaning of elements in a web page.

2.1 Is HTML a Programming Language?

People who are new to learning how to code frequently assume that HTML is a programming language if they do not have experience with any other coding languages. This is a reasonable assumption. However, there is a lot of disagreement about whether or not HTML, as a markup langauge, should be classified as a programming language as well.

Many more experienced programmers argue that from a technical perspective, it is ***not*** a programming language because it does not share the same qualities as most programming languages. People don't universally agree on what these qualities are. They range features such as variables, functions and flow control to being turing complete. However, some even more experienced programmers, including ones who work on HTML interpreters in browsers such as Firefox, argue that from a highly technical perspective, HTML ***is*** a domain-specific, declarative programming language. Watch this video for an explanation of this argument.

Ultimately, because nobody agrees on the precise definition of what a programming language is, this classification is somewhat subjective. What is important to understand is that, as a markup language, which is lacking features such as variables, functions and flow control, HTML is fundamentally different from an average programming language. PHP was created specifically to add these features to HTML.

HTML Resources

1. W3Schools
2. MDN
3. CSS Tricks
4. Scotch.io
5. Stack Overflow
6. W3C
7. W3C Validator

2.2 HTML Files

To create an HTML file, save the file with the extension `.html` or the less commonly used `.htm`. Regardless of which file extension you use, make sure you are contsistent.

HTML files which are saved with the name `index.html` and have been uploaded to a server will automatically be loaded as the home page when a domain name is entered. The same is true of files named `index.htm`. However, be careful about how you name your files as files named `index.htm` will take precedence over files named `index.html`.

2.3 Elements and Tags

An element is a chunk of text content which is grouped and defined for a specific purpose. All elements in an HTML page are defined by tags. Most elements have opening and closing tags which would be used to define whatever text content is in between them. Opening tags are defined by opening and closing angular brackets, with a tag name which describes the tag in between them. Closing tags are written the same way as opening tags except they have a forward slash before the word.

```
<article>This text would be defined as an article!</article>
<p>This text would be defined as a paragraph!</p>
```

Some tags, such as the line break tag, are self-closing tags and do not require closing tags.

```
<br>
```

2.4 Attributes

Attributes provide additional information about HTML elements. They are written inside of HTML tags, after the tag name but before the closing angular bracket.

```
<article lang="en-us">This article is in English!</article>

<article lang="fr">This article is in French!</article>
```

2.5 Comments

Comments are used to write text which will not be read as code to explain what code does and is for. HTML comments are defined by an angular bracket, exclaimation point and two hyphens to start the comment and two hyphens and an angular bracket to close the comment.

```
<!-- This is an HTML comment! -->

<!--
Comments can take
up multiple lines
-->
```

2.6 Universal Template

2.6.1 Document Type Declaration

Every HTML page uses the same basic universal template. This template starts with a document type declaration which identifies the document as an HTML page and defines the HTML version. The generic document type declaration will define the page as being the latest version of HTML which is HTML5. The document type declaration is defined by one self-closing tag:

```
<!DOCTYPE html>
```

2.6.2 HTML Tag

After the document declaration the rest of the HTML page is wrapped with the opening and closing `<html>` tags which define the start and end of the document.

```
<!DOCTYPE html><!-- Defines document type -->
<html><!-- HTML page starts -->
</html><!-- HTML page ends -->
```

2.6.3 Head Tag

The first tag which goes inside of the `<html>` tags is the `<head>` tag. The `<head>` tag defines ***information about*** the page. It is like the brain of the web page. It is also used to import external files.

```
<!DOCTYPE html><!-- Defines document type -->
<html><!-- HTML page starts -->
  <head><!-- Information about page starts -->
  </head><!-- Information about page ends -->
</html><!-- HTML page ends -->
```

2.6.4 Meta Tags

The word “meta” means something which refers to itself. Meta tags are tags that go in the <head> and provide information about the page. Every HTML page needs to have a meta tag which defines the character set at the beginning of the <head> tag. Meta tags are self-closing tags.

```
<!DOCTYPE html><!-- Defines document type -->
<html><!-- HTML page starts -->
  <head><!-- Information about page starts -->

      <meta charset=”UTF-8”><!-- Defines character set -->

  </head><!-- Information about page ends -->
</html><!-- HTML page ends -->
```

2.6.5 Title Tag

Every HTML page should also have a <title> tag inside of the <head> tag which defines the title of the web page that shows up in places such as the user’s browser tab and in Google searches.

```
<!DOCTYPE html><!-- Defines document type -->
<html><!-- HTML page starts -->
  <head><!-- Information about page starts -->
      <meta charset=”UTF-8”><!-- Defines character set -->

      <title>My Cool HTML Page</title><!-- Defines title -->

  </head><!-- Information about page ends -->
</html><!-- HTML page ends -->
```

2.6.6 Body Tag

After the <head> tag, every HTML page must have a <body> tag. The <body> tag is where all of the structural content of the page goes. of the page goes. If the <head> is the brain of the page, the <body> is the physical body of the page.

```
<!DOCTYPE html><!-- Defines document type -->
<html><!-- HTML page starts -->
  <head><!-- Information about page starts -->
      <meta charset=”UTF-8”><!-- Defines character set -->
      <title>My Cool HTML Page</title><!-- Defines title -->
  </head><!-- Information about page ends -->

  <body><!-- Structure of page starts -->
```

```
    <!-- Structural elements and content go in here -->

    </body><!-- Structure of page ends -->
</html><!-- HTML page ends -->
```

2.7 Inline vs. Block Elements

Every HTML element has a default “display” property which determines how that element takes up space and is positioned on the web page.

Inline elements start on the current line of text and only take up the space of whatever content is inside them. Block elements start on a new line and take up 100% of the width of the screen.

Examples of inline elements include: `<span>`, `<a>`, `<em>`, `<strong>`, `<i>`, `<b>`, `<u>`, `<img>`. Most other tags are considered block elements.

2.8 Heading Tags

Heading tags are used to define headings in a web page. Headings are used by browsers, screen readers and search engines to create an outline of the page by defining the the importance of headings using numbered tags which range from `<h1>` to `<h6>`. `<h1>` would be the most important heading and `<h6>` would be the least important heading.

```
    <h1>Heading 1</h1>
    <h2>Heading 2</h2>
    <h3>Heading 3</h3>
    <h4>Heading 4</h4>
    <h5>Heading 5</h5>
    <h6>Heading 6</h6>
```

You ***MUST*** start with an `<h1>` tag, use the headings in numerical order and ***MUST NOT*** skip from one level to the next. For instance, immediately after using an `<h1>` tag, you should use an `<h2>` tag and should not use `<h3>` to `<h6>` tags until you have. You could use another `<h2>` tag after using an `<h2>` tag though.

Historically, there was a rule that you could only have one `<h1>` tag per HTML page but this rule was changed with HTML5. However, you may now only use one `<h1>` tag per sectioning content element (see ***2.10 Sectioning Content***).

2.9 Semantic Markup

Semantic tags are tags which clearly describe the meaning and purpose of the content inside them to the browser, screen readers, search engines and developers. These are some examples of semantic tags and non-semantic tags.

2.9.1 Header and Footer Tags

The `<header>` and `<footer>` tags define the header and footer of either the overall HTML document or a sectioning content element. You may have multiple `<header>` or `<footer>` tags

per page but should only have one of each per sectioning content element (see ***2.10 Sectioning Content***).

2.9.2 Nav Tag

The `<nav>` tag is used to define a set of links which are used to navigate throughout a site. Not all links or groups of links should be inside of `<nav>` tags. Only major blocks of navigation.

2.9.3 Main Tag

The `<main>` tag specifies the main content of a page. It should only be used once and should contain content which is unique to the page.

2.9.4 Non-Semantic Tags

There are two generic HTML tags which are devoid of any semantic meaning. The only difference between them is that one is an inline tag and the other is a block tag (see ***2.7 Inline vs. Block Elements***).

The `<span>` tag is an inline tag with no semantic meaning. The `<div>` tag is a block tag with no semantic meaning. These tags should be used whenever an element does not meet the definition of any existing semantic tags. They are typically used more often than other tags but are overused as many people use them in places where they could, and should, be using more semantic tags.

2.10 Sectioning Content

Sectioning content elements are elements which are used to define sections where the numbering of heading tags restarts. This means that every sectioning content element can have its own `<h1>` tag. However, every sectioning content element may only have one `<h1>` tag inside of it until another sectioning content element is created inside of it.

Sectioning content elements are also used to group `<header>` and `<footer>` elements. You may only have one of each per sectioning content element.

The following tags are the sectioning content elements: `<article>`, `<aside>`, `<nav>`, `<section>`.

2.11 Classes and IDs

Classes are attributes which are used for classifying groups of HTML tags.

```
<div class="product"></div>
```

You may use a class as many times as you want, and a tag may have as many classes as you want.

```
<div class="product clearance"></div>
<div class="product featured"></div>
```

In contrast, IDs are used to identify one specific HTML tag. You may only use an ID once and you may only have one ID per HTML tag.

```
<article id="biography">This is my life story.</article>
```

Generally classes and IDs are used to target elements using languages such as CSS and JavaScript.

However, you can also link to IDs to link to specific parts of a web page.

3. CSS

CSS stands for Cascading Style Sheets. It is a stylesheet language. CSS controls the visual design and layout of the web page.

3.1 Is CSS a Programming Language?

There is even more debate over whether CSS is a programming language than there is over whether HTML is a programming language. Like HTML, CSS is fundamentally different from an average programming language, as a stylesheet language. It serves a very specific purpose and lacks many common features of programming languages such as standard flow control structures.

However, CSS does have a form of variables, which are called “custom properties.” There are also specific types of conditional statements called media queries. It is highly likely that full flow control features will be added to CSS in the future. While it is still somewhat subjective whether CSS is a programming language or not, it is becoming more and more like a standard, domain-specific programming language.

3.2 Style Tags and CSS Files

There are multiple ways to apply CSS to HTML. One way is using `<style>` tags. Any code written between `<style>` tags is read as CSS code. Generally, `<style>` tags go in the `<head>` tag.

```
<head>
  <meta charset=”UTF-8”>

  <title>My Cool HTML Page</title>

  <style>
  /* CSS code goes in here */
  </style>
</head>
```

CSS Resources

1. W3Schools
2. MDN
3. CSS Tricks
4. Scotch.io
5. Stack Overflow
6. W3C
7. W3C Validator

You can alternatively create an external CSS file by saving it with the file extension `.css` and linking to it from the `<head>` tag using a `<link>` tag. Common names for external CSS files include `style.css` and `main.css`. It is common convention to keep external CSS files in a folder called **css**.

```
<head>
  <meta charset=”UTF-8”>

  <title>My Cool HTML Page</title>

  <link rel=”stylesheet” href=”css/main.css”>
</head>
```

3.3 CSS Selectors and Targeting

CSS targets HTML elements using selectors. CSS selectors could be the name of an HTML tag, a class, an ID or even an attribute. To target an HTML tag, just use the name of the tag and follow it with curly braces where the CSS styles will be specified.

```
body {}
```

To target a class use a period and then the class name.

```
.some-class {}
```

To target an ID use a number sign (or hashtag symbol) followed by the name of the ID.

```
#some-id {}
```

To target a tag with an attribute use square braces with the attribute name and optionally the value inside.

```
[name] {}
[name=”email”] {}
```

You can target multiple CSS selectors by using a comma separated list.

```
h1, h2, h3, h4, h5, h6 {}
```

You can target nested HTML elements by using a space between selectors, without a comma between them.

```
nav ul li a {}
```

To target ONLY direct children of an element, use the greater than symbol.

```
nav > ul > li > a {}
```

To target ALL HTML elements use the asterisk.

```
* {}
```

3.4 Properties and Values

Once you have targeted an HTML element using a CSS selector, you can change the styles of that element by changing the values of CSS properties. Properties are the names of the types of styles you are changing such as color, background, width and font-family. Values are the options which properties are set to such as red, 100% and Arial.

```
body {
   margin: 0;
   font-family: Arial, sans-serif;
}

header {
   background: darkorchid;
   color: white;
   padding: 50px;
```

```
}
```

4. Intro to Servers

4.1 What is a server?

> **"A server is a computer program that provides a service to another computer programs (and its user). In a data center, the physical computer that a server program runs in is also frequently referred to as a server. That machine may be a dedicated server or it may be used for other purposes as well."**
>
> **— From Server, Techopedia**

So basically it's a computer program which stores and sends, or serves, data to the Front-End, being the web browser.

4.2 The Request/Response Process

1. **The client sends a request to the server.**
2. **In response, the server processes a server-side language like PHP, Python or Ruby on the back end and returns HTML, CSS and JavaScript code to display a web page on the front end**

4.3 What is a Back-End Developer?

> **"A back-end developer is a type of programmer who creates the logical back-end and core computational logic of a website, software or information system. The developer creates components and features that are indirectly accessed by a user through a front-end application or system."**
>
> **— From Back-end Developer, Techopedia**

Basically, back-end development is the data processing and logic which can not be seen or directly accessed by the user. This includes languages such as C, C++, Java, Ruby, Python, Perl, PHP and SQL.

4.3 XAMPP

XAMPP is a collection of software which allows us to run server and database software in our local environment.
XAMPP stands for X (your operating system), Apache, MySQL/MariaDB, PHP. This forms what is known as your development "stack".

4.3.1 The LAMP Stack

The traditional "stack" which is used by developers is known as the LAMP stack. The LAMP Stack

consists of:

1. **Linux is used on most servers but XAMPP allows us to use Windows or Mac OS as well**
2. **Apache is the server software**
3. **MySQL or MariaDB which are database software**
4. **PHP is our back-end scripting language**

If an employer asks for LAMP developers or asks you what "your stack" is, this is what they are referring to. The second "P" in XAMPP actually stands for Perl, but we won't be using that. Python developers also call their stack a LAMP stack. Other stacks include the LAMR stack (Ruby developers) and the MEAN stack (NodeJS developers).

1. **Download XAMPP at the Apache Friends website.**
2. **Once you have XAMPP installed turn on Apache and MySQL/MariaDB.**
3. **Skype uses port 80, the same port number that Apache uses so quit Skype if it's open.**
4. **Remember to save all projects which use technologies that are dependant on server software such as PHP and AJAX in C:/xampp/htdocs so that the server can find them.**
5. **In order to view these files using the server it is necessary to go to the address "localhost" in your browser which is an alias for C:/xampp/htdocs**

5. Intro to PHP

5.1 History of PHP

PHP was created in 1995 by by Rasmus Lerdorf, a Danish-Canadian programmer who grew up in King City and graduated from the University of Waterloo. PHP initially stood for Personal Home Page as Rasmus just created PHP so that he could make a personal website.

It was originally intended to act as a server-side templating language for HTML that would connect to a server-side programming language like C which would handle most of the data processing. As PHP gained in popularity, its functionality expanded beyond its original intended purpose and it is now a full-fledged programming language which can be used as both a templating language for HTML as well as for complex data processing.

The mascot for PHP is an elephant called the ElePHPant because one of the developers thought the acronym PHP looked like an elephant.

5.2 About PHP

As PHP expanded in complexity it was renamed and the acronym now stands for PHP: Hypertext Preprocessor. A preprocessor is a program which processes input data in order to produce output which is used as input data in another program. In the case of PHP which is a preprocessor for HTML, this means that PHP code is preprocessed to produce HTML code which is processed to build a final web page.

Because PHP is preprocessed on the server, PHP code cannot be seen on the client in the HTML source code. This keeps passwords and other sensitive information which is included in the code private. It also means the code is private.

PHP Resources

1. PHP Website
2. XAMPP
3. Documentation
4. W3Schools
5. CSS Tricks
6. Scotch.io
7. Stack Overflow

5.3 PHP Files and Tags

5.3.1 PHP Files

To use PHP with any HTML page, just resave that page as a `.php` file. Because PHP was designed as a templating language for HTML, PHP files can read ***both*** HTML and PHP code. Like HTML, files named `index.php` will load automatically when the domain is entered.

5.3.2 PHP Tags

The biggest advantage of using PHP over other server-side languages is that we can mix PHP code with our HTML code. To use PHP we need to use special PHP tags in our HTML. PHP tags look like this:

```
<?php ?>
```

These tags don't work like link tags for CSS. There is no external file to link to. We just insert them wherever we want our PHP in our HTML code. You can write PHP code wherever you want! PHP can be used in the title tag. PHP can be used in meta tags. You can even use PHP in style tags.

5.4 PHP Comments

Comments are used to write text which will not be read as code to explain what code does and is for. There are two types of comments in PHP.

5.4.1 Inline Comments

Inline comments go in one line. Inline comments are defined using two forward slashes.

```
// Inline PHP comments look like this and take up a single line
```

5.4.2 Block Comments

Block comments can take up multiple lines. Block comments are created by using a forward slash and an asterisk at the beginning of the comment and another forward slash and askterisk at the end.

```
/*
Block PHP comments
look like this and can
take up multiple lines
*/
```

5.5 Variables

We can also store data in “variables” to use later. Variables hold data which could change — or vary. Think of solving for x in algebra. Alternatively you can think of a variable as a box which holds an item. This item could be removed and replaced with a different item.

To create a variable, just write a dollar sign immediately before the name for the variable, followed by an equals symbol.

```
$title = “Intro to PHP”;
```

It is necessary to include the dollar sign whenever you reference a variable in PHP.

There are rules and conventions to naming variables in PHP:

1. **Variable names must start with a letter or an underscore, not a number**
2. **After the first character, characters can be either letters, numbers or underscores**
3. **Variable names can only contain alpha-numeric characters and underscores (A-z, 0-9 and _)**
4. **Names are case-sensitive: `$some_variable`, `$SOME_VARIABLE` and `$Some_Variable` are all different variables**
5. **When naming variables in PHP there are two naming conventions which people tend to follow:**
   Lower snake case: `$snake_case_looks_like_this`
   Lower camel case: `$camelCaseLooksLikeThis`

Snake case is most popular in PHP so that is what we will use.

6. Data Types

There are 10 primitive data types in PHP.

6.1 String

6.1.1 Double Quoted Strings

A string is a chunk of plain text with no predefined meaning. Strings are defined using quotes.

```
$description = “A description for this page.”;
$body_margin = “0”;
$body_font_family = “Arial, sans-serif”;
$header_bg = “darkorchid”;
$header_color = “white”;
$header_padding = “50px”;
$text_class = “.text-center”;
```

6.1.2 Single Quoted Strings

You can also write strings with single quotes.

```
$name = 'Tim';
```

We can insert HTML into strings.

```
$some_html = '<p class="text-center">Hello World!</p>';
```

One difference between using double or single quotes is that you can use single quotes/apostrophes inside of double quoted strings and double quotes inside of single quoted strings. PHP developers usually use single quotes for strings because there are so many double quotes in HTML.

6.1.3 Escaping Characters

The backslash can be used to escape special characters in strings to ensure that they are read as part of the string instead of being read as code. The most common example of this would be when you need to use the same type of quotes inside of your quotes without ending a string.

```
$double_quotes = "This is how you ignore \"double quotes\"";
$single_quotes = 'That\'s how you\'d ignore single quotes';
```

6.2 Integer

Whole numbers are called integers. Numbers are only read as integers if they are outside of quotes. Otherwise they would be considered strings.

```
$age = 25;
```

6.3 Float

Numbers with decimals are their own data type called floats.

```
$average = 4.365;
```

6.4 Boolean

Booleans are just a way of creating switches that can turn settings on or off. They can only be either true or false.

```
$teaching = true;
$on_break = false;
```

6.5 Array

Arrays are just a way of storing a list of multiple pieces of data in one variable. They are written as comma separated lists. They are defined using square braces.

```
$data_list = [$age, $average, $teaching];
```

6.6 Objects

Objects are another way of storing mutiple pieces of data but they are very complex. We will spend a

whole day talking about them.

6.7 Null

A null is a totally empty and meaningless value. A null value either hasn't been set yet or is not applicable in a given context.

```
$nothing = null;
```

It is hard to imagine how these would be useful at first but nulls are very important in PHP.

6.8 Callable, Iterable and Resource

There are three more data types called callable, iterable and resource. These types don't come up very frequently but it is good to know about them. For more information about these types, view the PHP documentation.

7. Testing and Displaying Data

`echo` is a keyword which is used to insert data into HTML with PHP.

```
echo $name;
```

If we `echo` strings with HTML then it will be read as HTML.

```
echo '<p class="text-center">Hello World!</p>';
```

`var_dump()` is a function for testing variables.

```
var_dump($name);
```

8. Concatenation, Interpolation and Operators

8.1 Concatenation

"Concatentation" is the act of linking data together in a series. We can take strings and connect them together with other strings, variables and functions using "concatenation." When you think of CONcatenation think of CONnection. In PHP concatenation is done with the period.

```
echo '<p class="text-center">Hello ' . $name . '</p>';
```

8.2 Interpolation

"Interpolation" is the insertion of data into other data. It's like concatenation but the variable is inserted directly into a string instead of being linked to the beginning or end of it. When you think of INterpolation think of INsertion. You ***MUST*** use ***double quotes*** for strings in order to use interpolation in PHP. In order to "interpolate" variables into our strings we need to wrap them in curly braces.

```
echo “<p class=\”text-center\”>Hello {$name}</p>”;
```

8.3 Operators

8.3.1 Arithmatic Operators

Like every programming language, PHP can perform calculations using integers and floats.

```
echo 5 + 2; // addition
echo 5 - 2; // subtraction
echo 5 * 2; // multiplication
echo 5 / 2; // division
echo 5 % 2; // modulus: devides two numbers and gives you the
remainder
```

8.3.2 The Concatenation Operator

The concatenation operator is used to assign the value of a variable as whatever the variable was previously, with the new value concatenated onto the end.

```
$name = ‘Tim’;
$last_name = ‘ Lai’;
$name .= $last_name;
echo ‘<p> ‘ . $name . ‘ </p>’;
```

This code would produce the full name “Tim Lai.”