$var Variables

Variables

Variables are used as "containers" in which we store information.
A PHP variable starts with a dollar sign ($), which is followed by the name of the variable.

```
$variable_name = value;
```

Rules for PHP variables:
- A variable name must start with a letter or an underscore
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($name and $NAME would be two different variables)

For example:

```
<?php
  $name = 'John';
  $age = 25;
  echo $name;

 // Outputs 'John'
?>
```

Try It Yourself

In the example above, notice that we did not have to tell PHP which data type the variable is.
PHP automatically converts the variable to the correct data type, depending on its value.

> Unlike other programming languages, PHP has no command for declaring a variable. It is created the moment you first assign a value to it.

Constants

Constants are similar to variables except that they cannot be changed or undefined after they've been defined.
Begin the name of your constant with a letter or an underscore.
To create a constant, use the **define()** function:

```
define(name, value, case-insensitive)
```

Parameters:
name: Specifies the name of the constant;
value: Specifies the value of the constant;
case-insensitive: Specifies whether the constant name should be case-insensitive. Default is **false;**

The example below creates a constant with a **case-sensitive** name:

```
<?php
 define("MSG", "Hi SoloLearners!");
 echo MSG;
```

```
  // Outputs "Hi SoloLearners!"
?>
```

Try It Yourself

The example below creates a constant with a **case-insensitive** name:

```
<?php
 define("MSG", " Hi SoloLearners!", true);
 echo msg;

 // Outputs "Hi SoloLearners!"
?>
```

Try It Yourself

No dollar sign ($) is necessary before the constant name.

Data Types

Variables can store a variety of data types.
Data types supported by PHP: **String**, **Integer**, **Float**, **Boolean**, **Array**, **Object**, NULL, Resource.

PHP String

A **string** is a sequence of characters, like "Hello world!"
A string can be any text within a set of single or double **quotes**.

```
<?php
 $string1 = "Hello world!"; //double quotes
 $string2 = 'Hello world!';  //single quotes
?>
```

You can join two strings together using the dot (.) concatenation operator.
For example: **echo $s1 . $s2**

PHP Integer

An **integer** is a whole number (without decimals) that must fit the following criteria:
- It cannot contain commas or blanks
- It must not have a decimal point
- It can be either positive or negative

```
<?php
 $int1 = 42; // positive number
 $int2 = -42; // negative number
?>
```

Variables can store a variety of data types.

PHP Float

A **float**, or floating point number, is a number that includes a decimal point.

```php
<?php
  $x = 42.168;
?>
```

PHP Boolean

A **Boolean** represents two possible states: TRUE or FALSE.

```php
<?php
  $x = true; $y = false;
?>
```

Booleans are often used in conditional testing, which will be covered later on in the course.

Most of the data types can be used in combination with one another. In this example, **string** and **integer** are put together to determine the sum of two numbers.

```php
<?php
  $str = "10";
  $int = 20;
  $sum = $str + $int;
  echo ($sum);

  // Outputs 30
?>
```

Try It Yourself

PHP automatically converts each variable to the correct data type, according to its value. This is why the variable **$str** is treated as a number in the addition.

Variables Scope

PHP variables can be declared anywhere in the script.
The **scope** of a variable is the part of the script in which the variable can be referenced or used.

PHP's most used variable scopes are **local**, **global**.
A variable declared outside a function has a **global scope**.
A variable declared within a function has a **local scope**, and can only be accessed within that function.

Consider the following example.

```php
<?php
  $name = 'David';
  function getName() {
    echo $name;
  }
  getName();

  // Error: Undefined variable: name
?>
```

Try It Yourself

This script will produce an error, as the **$name** variable has a **global** scope, and is not accessible within the **getName()** function. Tap continue to see how functions can access global variables.

> **Functions** will be discussed in the coming lessons.

The global Keyword

The **global** keyword is used to access a global variable from within a function.
To do this, use the **global** keyword within the function, prior to the variables.

```
<?php
 $name = 'David';
 function getName() {
  global $name;
  echo $name;
 }
 getName();

 //Outputs 'David'
?>
```

Try It Yourself

> Tap **Try It Yourself** to play around with the code!

Variable Variables

With PHP, you can use one variable to specify another variable's name.
So, a **variable variable** treats the value of another variable as its name.

For example:

```
<?php
 $a = 'hello';
 $hello = "Hi!";
 echo $$a;

 // Outputs 'Hi!'
?>
```

Try It Yourself

> **$$a** is a variable that is using the value of another variable, **$a**, as its name. The value of **$a** is equal to "hello". The resulting variable is **$hello**, which holds the value "Hi!".

End.