# What is PHP?

PHP is a general purpose scripting language.

Stands for PHP: Hypertext Preprocessor

…it uh also once stood for Personal Home Page (gross, thats why they changed it)

* PHP is built in C, so the syntax is going to be familiar to anyone whose used C or a C variant (like Java) before.

## Comments

// inline  
  
/\*  
multiline  
\*/

## Output

How do we display something to the screen?

echo "hello";  
print 'hello';  
var\_dump('hello');

## String Literals

echo 'this is a string literal'; // this will literally print whatever is in here  
  
echo "this is also a string, but allows for interpolation";

### Interpolation?

Think of interpolation as “variable value injection.” If you put a variable into a string with double quotes, PHP will swap the variable for the value.

There are a couple of different ways you can do this:

$x = 'pizza';  
  
echo "I love $x"; // I love pizza  
echo "I love {$x}"; // I love pizza the {} make it easier to read  
  
echo 'I love $x'; // not interpolated so this reads I love $x

Interpolation *only subs variables nothing else* these cases don’t work:

echo "this isn't a value: {1+3}"; // returns: this is a value {1+3}  
echo "this just fails: {$i++}"; // error

## Variables

All variables in PHP start with the $

//valid names...  
$a = 'foo';  
$\_a = 'foo';  
$a\_123 = 'foo';  
$another\_type = 'foo';  
  
//invalid names...  
$one-type = 'foo';  
$32 = 'nope';

### Constants

You can set constants in PHP too, no $ is needed, by convention, constants should be in ALL CAPS.

//constants  
define('CENTS', 100);  
  
// usage  
echo 500 / CENTS; // 5

*Note that when we define a constant, we wrap it in either single or double quotes, but when we use it…no quotes*

### Case sensitive variables

Generally speaking PHP is **case sensitive**

$foo = 1;  
$Foo = 2;  
$FOO = 3;  
$foO = 4;  
  
// on Linux and Mac these are 4 different values  
var\_dump($foo); // int(1)  
var\_dump($Foo); // int(2)  
var\_dump($FOO); // int(3)  
var\_dump($foO); // int(4)

This is **NOT** the case on Windows. Windows is a *case insensitive* operating system.

Be careful, this can lead to a whole suite of bugs that are hard to find and replicate.

#### Best Practices for Variable Names

1. The name should make sense in context
2. All variables should be lowercase
3. Pick a style and stick with it: separate vars with either an underscore or use camel case: $under\_score\_style vs $camelCaseStyle – there’s no “right way” to handle this, just pick one way and *EVERYONE CODE TO THAT STANDARD*

## Types

PHP is loosely typed. It will figure out what type the variable is for you.

$a = 42; // integer  
$b = 4.2; // float  
$c = 'forty two' // string  
$d = true; // boolean  
$e = []; // array  
$f = ['one' => 1, 'two' => 2, 'three' => 3] // associative array  
  
var\_dump($a);  
var\_dump($b);  
var\_dump($c);  
var\_dump($d);  
var\_dump($e);  
var\_dump($f);

## Operations

All your standard math operations are here too and PHP follows the order of operations when doing a calculation.

echo 4 + 4 / 2; // = 6   
echo (4 + 4) / 2; // = 4

## Incrementing/Decrementing Values

$i = 0;  
$i++;  
echo $i; // = 1  
$i+=1;  
echo $i; // = 2  
$i = $i + 1;  
echo $i; // = 3  
  
// decrementing   
$i--;  
echo $i; // = 2  
echo $i++; // = 2...wait what?  
echo $i; // = 3...huh??

think of the ++ as a second instruction to the PHP interpreter. In the above example we said, “echo out the value of $i…then increment it”

If you want to increment *first* you can do that too:

echo ++$i // = 4

this says, “echo out the value of $i, after you increment it”

### String Operations

$greeting = 'Hello';  
echo $greeting . 'World'; // Hello World  
  
$location = 'World!';  
  
$full = $greeting . $location;  
echo $full; // Hello World!  
  
$big = 'Now this is the story'  
 . ' all about how'  
 . ' my life got twist-turn upside down <br>'  
 . ' i\'d like to take a minute just sit right there'  
 . ' and I\'ll tell you how I became the prince of bell air';  
  
echo $big;  
  
$bigger = "hello world <br> this is a test";  
echo $bigger;  
  
$all = "Story time! <br> {$big} well? what did you think? {$bigger}";  
  
echo $all;

## Conditionals

PHP has the same conditionals as every other OOP language out there.

### If/Else

$x = 1;  
if ($x == 1) {  
 echo "ONE";  
} else {  
 echo "TWO";  
}

### If/ElseIf

$x = 3;  
if ($x == 1) {  
 echo "ONE";  
} elseif ($x > 1) {  
 echo "> 1";  
} else {  
 echo "gotta be < 1";  
}

### Switchs

switch ($x) {  
 case 1:  
 echo "ONE SWITCH";  
 break;  
  
 default:  
 echo "DEFAULT";  
 break;  
}

*Note:* Switch statements seem like a good idea when you’re starting out programming. They have this nice logical flow to them and you don’t need to keep rewriting all these if statements. In practical terms, they’re a mess and they usually represent a “code smell.” A code smell means there’s a problem with your logic or a fundamental issue with what you’re doing. The truth is, 95% of the time, you don’t *need* a switch, you need to think about how and what you’re writing in a different way.

### Ternary Operations

A ternary operation is used as a shortcut to an if statement. It follows the pattern:

condition ? expr1 : expr2

The above says, “if the condition is met, return the value for expr1 otherwise return the value for expr2”

$i = true;  
echo ($i == true) ? "true" : "false";

## Type Checking

Again PHP is a loosely typed language. (did I say this is important?) Remember that we didn’t need to tell PHP that a variable was a specific type and that PHP will just figure it out for us. This has some advantages, but also some pitfalls.

Consider the following:

$i = 1;  
  
var\_dump($i == 1); // true  
var\_dump($i == '1'); // true  
var\_dump($i == '1lsdkfjlsdkfjkjsdf'); // true  
var\_dump($i == true); // true

What just happened? Since PHP is loosely typed, it’s not checking types in the above examples, only values.

**Great, but what if I need to make sure the value AND the type are equal?**

We can use the === or !==

var $i = 1;  
  
var\_dump($i === 1); // true  
var\_dump($i === '1'); // false  
var\_dump($i === '1lsdkfjlsdkfjkjsdf'); // false  
var\_dump($i === '1'); // false  
var\_dump($i === true); // false

*It is recommended that you do this type of check for most, if not all your conditionals*

### Type Casting

$x = '1';   
$y = (int) $x;   
  
var\_dump($x); // string  
var\_dump($y); // integer  
  
var\_dump(intval($x)); // integer  
  
$z = 4.2;  
var\_dump(intval($z)); // 4  
var\_dump( (int) $z); // 4

### Money

Let’s talk about money while we’re here. There are a lot of different ways to store money in a database. The simplest way is to store all your prices/money as an **INTEGER** and as **cents**.

For example, if you have a product that is $5.00 you’ll store that in the database as 500. When you need to display it on the front end you just divide by 100. Many payment gateways (like Stripe) will expect you to send them values as cents too so get in this habit.

### More Conversion Stuff…

$x = "100" + 20;  
var\_dump($x); //(int) 120  
  
$x = 30 + "feet";  
var\_dump($x); // ERROR  
  
$x = "30" + "feet";  
var\_dump($x); // ERROR  
  
$x = "30" . "feet";  
var\_dump($x); // (string) 30feet

## Loops

PHP has your standard looping structures

for ($i=0; $i < 10; $i++) {   
 // ...  
}  
  
do {  
 // ...  
} while ( $x <= 10);  
  
while ($x <= 10) {  
 // ...  
}

### Looping Arrays

$a = [1, 2, 3, 4, 5];  
  
for ($i=0; $i < count($a) ; $i++) {  
 echo $a[$i] . "<br>";  
}  
  
$a2 = [  
 'one' => 1,  
 'two' => 2,  
 'three' => 3,  
 'four' => 4  
];  
  
foreach ($a2 as $key => $value) {  
 echo "{$key} --> {$value} <br>";  
}  
  
foreach ($a2 as $value) {  
 echo "{$value} <br>";  
}

## FizzBuzz

This is a little problem that you’ll (probably) be asked 100 times to code if you’re going into software development.

*Write a program that prints the numbers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”*

### Solution

for ($i = 1; $i <= 100; $i++) {  
 $s = '';  
  
 if ($i % 3 == 0) {  
 $s .= 'Fizz';  
 }  
  
 if ($i % 5 == 0) {  
 $s .= "Buzz";  
 }  
  
 echo ($s !== '') ? "{$s}<br>" : "{$i}<br>";  
}

## Functions and Classes in php

function hello() {  
 return "hello";  
}  
  
echo hello();

class Hello {  
  
 private $x;  
  
 public function \_\_construct()  
 {  
 $this->x = true;  
 }  
  
 public function getX()  
 {  
 return $this->x;  
 }  
  
 public function setX(bool $newX)  
 {  
 $this->x = $newX;  
 }  
}  
  
$h = new Hello();  
var\_dump($h->getX());  
  
$h->setX('food'); // do you care about this? maybe?  
var\_dump($h->getX());

If you want to force strict types you can, but only on a *per file* basis

<?php declare(strict\_types=1);