Oliver Peacock

PHP Notes

# Using PHP

<?php and ?> tags.

# Printing

Use echo command and text afterwards.

\” – allows “ to be used in text

\n – new line

echo “1” . “2” – used to chain echo (concatenated).

# Commenting

/\* text in between \*/ or // followed by test

# Variables

$\*\*\* = “pen” – String variable

$nervar = $\*\*\*; - a ner variable gets same text as $\*\*\*

$nervar .= ", boranges"; adds more text to end of old variable and makes It new variable

$nervar =& $\*\*\* nerver variable will always be same as \*\*\* if \*\*\* ever changes

Echo “Hello” .$\*\*\*.”sir” – allows variable in echo

Echo “hi ${\*\*\*}” – parsing doesn’t need concatenated

# Numbers

Allows two number data types: Int and Float

+ add – subtract \* multiply / divide \*\* Power of % modular

+= -= \*= /= short syntax

$num++ adds 1 came with other operations

# Functions

Function x(){ } basic function.

x(); calls function

return "blastoff!"; returns the value, stops the function so put at end of function

if no return in function it was return as NULL;

$value = x(); calls and runs the x function, the returned value is put in $value

echo first(), second(), third(); use comma for splitting functions

**sending data through functions**

function sayCustomHello($name)

{

echo "Hello, $name!";

};

sayCustomHello("Aisle Nevertell"); // Prints: Hello, Aisle Nevertell!

sayCustomHello("Codecademy learner"); // Prints: Hello, Codecademy Learner!

Func(12, 3)

Function Fun($one, $two)

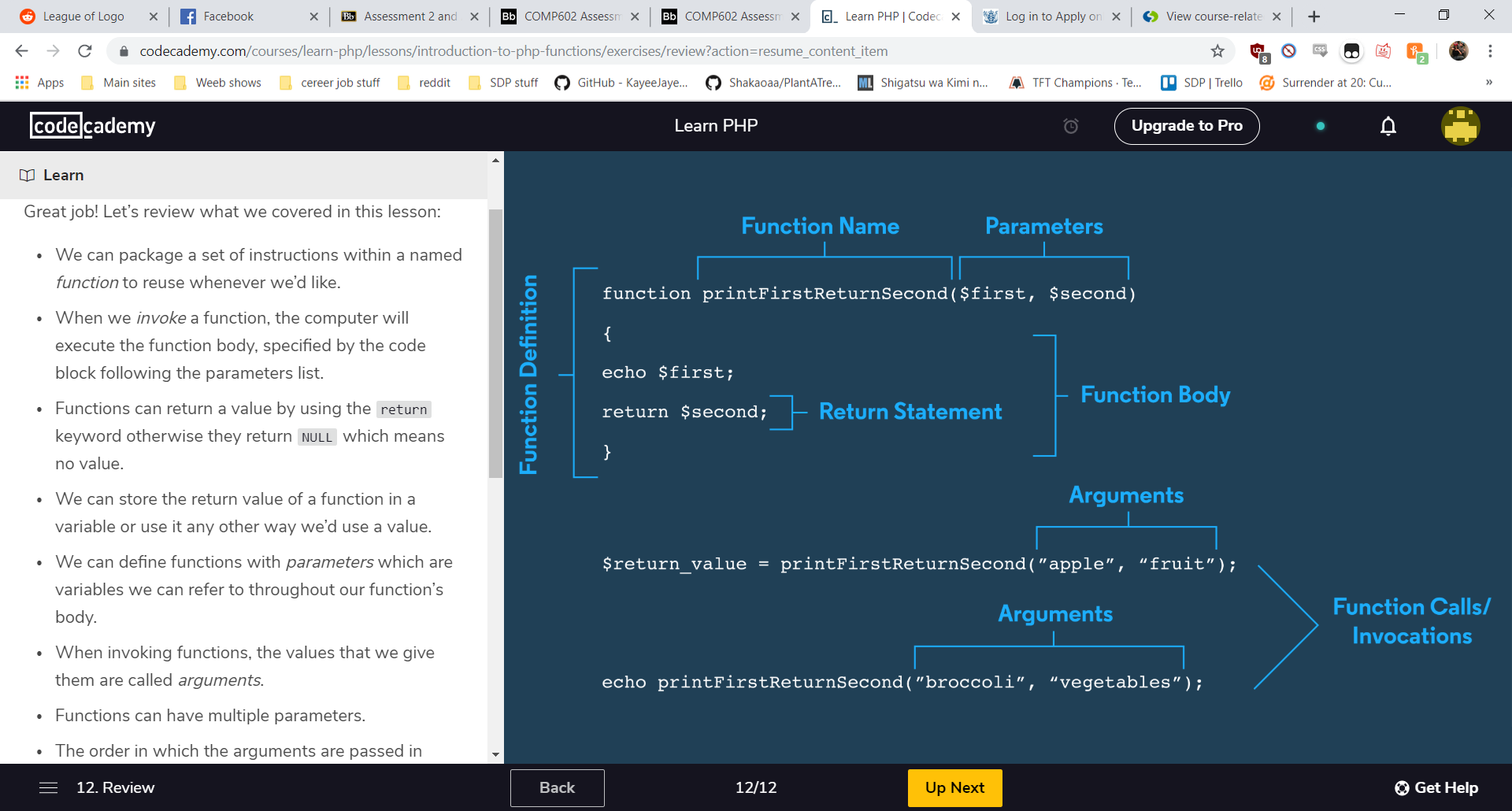
{

Return $one/$two;

};

Function func($name = “Oliver”); uses Oliver as name value unless called using different value from calling line.

function convertToQuestion(&$param) the & is a reference and makes values share same memory



# Built-in PHP Functions

Gettype() – gets datatype of argument

Var\_dump() – gets datatype and length of variable

Strrev() – reverses string

Strtolower() – lower cases string

Str\_repeat(“hi”, 10) – takes a string and an int and echos string that many times.

Substr\_count($paragraph, “like”); - counts every time like is said in the variable.

Abs() – gives absolute value

Round() gives a rounded full number

getrandmax(); maximum random number allowed

rand(); - generates random number between set values

str\_pad($a,$b,$c,$d); - Pad a string to a certain length with another string

date() – current date

full list of inbuild functions: <https://www.php.net/manual/en/ref.strings.php>

math functions: <https://www.php.net/manual/en/ref.math.php>

# Arrays

$first\_array = array("hello", 1, "ss", 4, "wasf"); - makes array

$number\_array = [0, 1, 2]; - short syntax arrays

print\_r($number\_array); - prints all arrays values

echo implode(", ", $number\_array); - puts array into string output

$string\_array[] = "third element"; - adds new element in array

$string\_array[0] = “jaskjdjas” – changes specific array

array\_pop($array) – removes last array as a return

array\_push() – allows new elements in array, example below

$new\_array = ["eeny"];

$num\_added = array\_push($new\_array, "meeny", "miny", "moe");

array\_shift() – same as array pop but for first element.

array\_unshift() – same as push but for beginning.

nested\_arr = [[2, 4], [3, 9], [4, 16]]; - Nested array are arrays in arrays.

=> - associates a key to its value for arrays eg below

$about\_me = array(

"fullname" => "Aisle Nevertell",

"social" => 123456789

);

echo $my\_array["panda"]; - this grabs the value in the associate key

$my\_array["capybara"] = "cutest"; - adds new element to associate array

unset($nums["one"]); - removes a key

$animal\_rankings = $my\_array + $more\_rankings; - join arrays, cant have identical keys

function reallyChangeColor (&$arr) - passing array through function

example changing arrays through functions

## HTML notes for PHP

<?= - can be used in the html document as a quick echo for php

# Boolean

if (/\*some condition\*/) { - if statements

// Do something...

}

Else if(/\*some condition\*/){

//do something

}

else {

//Do this instead

}



Above example using if and functions array

Bollean makes condition to see if true or not

1 < 10; // Evaluates to: TRUE

11 < 3; // Evaluates to: FALSE

=== - identical

!== - not identical

Below example is uses swtiches as alternate for if === statements

switch ($letter\_grade){

case "A":

echo "Terrific";

break;

case "B":

echo "Good";

break;

default:

echo "Invalid grade";

}

$isClicked = FALSE;

$link\_color = $isClicked ? "purple" : "blue"; - shorthand if statements

if ("") {

echo "this will not print"; - if theres a string in if that’s classed as false it means falsy, non empty is truthy and will play string

} elseif (null) {

echo "this will not print";

} elseif ([]) {

echo "this will not print";

} elseif (0) {

echo "this will not print";

} elseif ("0") {

echo "this will not print";

} else {

echo "Finally!";

}

Statement below is a reading statement that allows user input

echo "Hi, I'm Aisle Nevertell. What's your name?\n";

$name = readline(">> ");

echo "\nNice to meet you, $name";

Nested loops (loops within a loop)

|| or or - if left or right is TRUE

&& or and - both left and right is true, otherwise is false

! – put before a statement asking if it isn’t true to statement

Xor – one statement but not both is true

include "one.php"; - use include to apply other php files when working in one

## looping

Below is example of usage of while loop

$count = 1;

while ($count < 11)

{

echo "The count is: " . $count . "\n";

$count += 1;

}

Below is usage of do while loop, a loop that keeps going until condition is met

$count = 1;

do {

echo "The count is: " . $count . "\n";

$count += 1;

} while ($count < 11);

For loop is another type of while loop with conditions

for ($count = 1; $count < 11; $count++)

{

echo "The count is: " . $count . "\n";

}

Foreach loops is array usage loops

$counting\_array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

foreach ($counting\_array as $count) {

echo "The count is: " . $count . "\n";

}

You can use break in a loop to get out of the loop

$count = 1;

while ($count < 11)

{

echo "The count is: " . $count . "\n";

if ($count === 5) {

break;

}

$count += 1;

}

## html form example

<!DOCTYPE html>

<html lang="en" dir="ltr">

<body>

<h1>Welcome To This Form</h1>

<form action="" method="POST">

<label for="text">You can enter text here:</label>

<input type="text" name="text">

<hr>

<label for="num">You can enter a number here:</label>

<input type="number" name="num">

<hr>

<label for="slider">You can slide this:</label>

<br>

<span>Left</span>

<input type="range" name="slider" value="3" min="1" max="5">

<span>Right</span>

<hr>

<label for="boxes">You can check these:</label>

<input type="checkbox" name="boxes" value="first">

<label for="first">First</label>

<input type="checkbox" name="boxes" value="second">

<label for="second">Second</label>

<input type="checkbox" name="boxes" value="third">

<label for="third">Third</label>

<hr>

<label for="radio">You can select one of these:</label>

<input type="radio" name="radio" value="true">

<label for="true">TRUE</label>

<input type="radio" name="radio" value="false">

<label for="false">FALSE</label>

<hr>

<label for="dropdown">You can select one of these</label>

<select name="dropdown">

<option value="first">First</option>

<option value="second">Second</option>

<option value="third">Third</option>

</select>

<hr>

<input type="submit" value="Submit to Reset">

</form>

</body>

</html>

## Php form usage

<?php

function checkWord($input, $letter){

if ($\_SERVER["REQUEST\_METHOD"] === "POST" && strtolower($input[0]) !== $letter) {

return "\* This word must start with the letter ${letter}!";

} else {

return "";

}

}

?>

<h1>Time to Practice our ABCs</h1>

<form method="post" action="">

Enter a word that starts with the letter "a":

<br>

<input type="text" name="a-word" id="a-word" value=<?= $\_POST["a-word"];?>>

<br>

<p class="error" id="a-error"><?= checkWord($\_POST["a-word"], "a");?></p>

<br>

Enter a word that starts with the letter "b":

<br>

<input type="text" id="b-word" name="b-word" value=<?= $\_POST["b-word"];?>>

<br>

<p class="error" id="b-error"><?= checkWord($\_POST["b-word"], "b");?></p>

<br>

Enter a word that starts with the letter "c":

<br>

<input type="text" id="c-word" name="c-word" value=<?= $\_POST["c-word"];?>>

<br>

<p class="error" id="c-error"><?= checkWord($\_POST["c-word"], "c");?></p>

<br>

<input type="submit" value="Submit Words">

</form>

<div>

<h3>"a" is for: <?= $\_POST["a-word"];?><h3>

<h3>"b" is for: <?= $\_POST["b-word"];?><h3>

<h3>"c" is for: <?= $\_POST["c-word"];?><h3>

<div>

## Classes

Basic class

class Pet {

public $name, $color;

}

Usage of a class

class Beverage {

public $temperature, $color, $opacity;

}

$tea = new Beverage();

$tea->temperature = "hot";

echo $tea->temperature;

using function in a class in the main

class Beverage {

public $temperature, $color, $opacity;

function getInfo() {

return "This beverage is $this->temperature and $this->color.";

}

}

$soda = new Beverage();

$soda->color = "black";

$soda->temperature = "cold";

echo $soda->getInfo();

using constructer with function

<?php

class Beverage {

public $temperature, $color, $opacity;

function \_\_construct($temperature, $color) {

$this->temperature = $temperature;

$this->color = $color;

}

function getInfo() {

return "This beverage is $this->temperature and $this->color.";

}

}

$soda = new Beverage("cold", "black");

echo $soda->getInfo();

using inheritance and overloading that replaces a message

class Beverage {

public $temperature;

function getInfo() {

return "This beverage is $this->temperature.";

}

}

class Milk extends Beverage {

function \_\_construct() {

$this->temperature = "cold";

}

function getInfo(){

return parent::getInfo() . " I like my milk this way.";

}

}

$milk = new Milk();

echo $milk->getInfo();

private variables is used if we don’t want it to access other classes

class Beverage {

private $temperature, $color;

public $opacity;

function \_\_construct($temperature, $color) {

$this->temperature = $temperature;

$this->color = $color;

}

function getInfo() {

return "This beverage is $this->temperature and $this->color.";

}

}

class Milk extends Beverage {

function setOpacity($opacity) {

$this->opacity = $opacity;

}

}

Protected is used if we want someone to be allowed in child classes but not outside it

class Beverage {

private $temperature, $color;

protected $opacity;

function \_\_construct($temperature, $color) {

$this->temperature = $temperature;

$this->color = $color;

}

function getInfo() {

return "This beverage is $this->temperature and $this->color.";

}

}

class Milk extends Beverage {

function setOpacity($opacity) {

$this->opacity = $opacity;

}

}

Using static members are accessed using the Scope Resolution Operator (::).

class AdamsUtils {

public static $the\_answer = 42;

public static function addTowel($string) {

return $string . " and a towel.";

}

}

$items = "I brought apples";

echo AdamsUtils::$the\_answer;

echo "\n";

echo AdamsUtils::addTowel($items);