# Everlytic Developer Assessment

|  |  |
| --- | --- |
| Name | Stephan Van Zyl |
| Date | 26 April 2020 |

Please complete the answers to the questions below. The assessment should take roughly 30 minutes.

### What is the difference between public, protected and private in a class definition?

|  |
| --- |
| This defines the class variable visibility |
| public can be accessed from anywhere |
| protected can only be accessed from within the class where it is defined, or any derived classes  private can only be accessed from within the class where it is defined |

### Given this code: function doSomething(&$foo) { $bar = $foo; $foo += 1; return $foo; } $value = 3; $result = doSomething($value); echo "value: $value, result: $result"; What will be output to screen and why?

|  |
| --- |
| Output will be “value: 4, results: 4” |
| In the function we are passing an argument by reference. This means we are actually passing the memory address of the variable into the function and changing the variable contents directly |
| As opposed to passing a variable by value where we actually pass a copy of the variable into the function and not altering the original variable directly. |

### What is wrong with this query: "SELECT \* FROM table WHERE id = $\_POST[ 'id' ]"?

|  |
| --- |
| 1.) We should sanitize user input before using it. |
| 2.) Select \* will select everything from the given table, this might not always be desirable and we should probably be more specific in the fields we want. |
| 3.) We are already inside double quotes., we should not be adding quotes around the array key. |

### What is the cause of this warning: 'Warning: Cannot modify header information - headers already sent', and what is a good practice to prevent it?

|  |
| --- |
| This error occurs when anything is sent before you send HTTP headers, such as accidental white space at the beginning or end of a file, as such you can omit the closing php tag as it’s not really required. |
|  |
|  |

### What is wrong with this code: class Foo { protected $bar; public function \_\_construct() { $this->bar = 1; } public static function doSomething() { return $this->bar; } }

|  |
| --- |
| $this Can only be used when creating non-static methods in a class, as $this refers the instance of the class. |
| Static methods do not have access to instance variables. |
|  |

### 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".

<?php

for ($i = 1; $i <= 100; $i++) {

$multipleOfThree = (($i % 3) === 0);

$multipleOfFive = (($i % 5) === 0);

if (!$multipleOfThree && !$multipleOfFive) {

print $i . "<br/>";

// No need to evaluate further

continue;

}

if ($multipleOfThree) {

print('Fizz');

}

If ($multipleOfFive) {

print('Buzz');

}

print("<br/>");

}

### **What does the following code do? Explain what’s going on there.**

$date = '08/26/2003';

print preg\_replace('/([0-9]+)\/([0-9]+)\/([0-9]+)/'‚ '$2/$1/$3', $date);

|  |
| --- |
| It will swap the day and the month around (26/08/2003) |
| In the regular expression we have three groups all looking for numbers from 0-9. preg\_replace takes three arguments – the pattern, the replacement and the subject. Our pattern as explained, looks for groups of numbers, the replacement says replace position 1 with group matched in position 2, replace potion 2 with group matched at position 1 and leave position 3 as is. |

### **Given a line of text $string, how would you write a regular expression to strip all the HTML tags from it?**

|  |
| --- |
| /<[^>]\*>/g |
| Start by matching a “<” character. Then we look for anything but the closing tag “[^>]\*” for an unlimited number of characters. And then we look for the closing “>” tag. |
|  |

### A palindrome is a word that reads the same backward or forward. Write a function that checks is a given word is a palindrome. Characters case should be ignored. EG. Deleveled is a palindrome and should return true as character case is ignored.

### <?php class Palindrome

### {

### public static function isPalindrome($word)

{

$reversed = $word;

// Do in place replacement of string

for ($i = strlen($reversed) - 1, $j = 0; $j < $i; $i--, $j++) {

$temp = $reversed[$i];

$reversed[$i] = $reversed[$j];

$reversed[$j] = $temp;

}

// Could also have used the built in php function

// $reversed = strrev($word);

return strtolower($word) === strtolower($reversed);

}

### } echo Palindrome::isPalindrome('Deleveled');

### What security issue is prevalent in the code below and how would you fix it? <?php $messageStmt = $db->query('select message\_text from messages where message\_id = 1'); $messageStmt->execute(); $message = $messageStmt->fetch(PDO::FETCH\_OBJ); ?> <div><?= $message->message\_text; ?></div> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

There is nothing wrong with the above statement, because we are not including any variables in the select, so the use of query() is fine (query will execute when it’s called and there is no need to call execute()). However, if we were to introduce variables in the query, it is best to use a prepared statement instead of a query statement. This will prepare the query statement with placeholder marks, and then execute it, sending variables separately.

### Write an inner join for the following tables

|  |
| --- |
| SELECT  u.UserKey,  u.FirstName,  u.LastName,  a.AddressKey,  a.Address1,  a.Address2,  a.City,  a.State,  a.ZipCode  FROM User AS u  INNER JOIN Address AS a ON a.UsrKey = u.UserKey |
|  |

### 