# Everlytic Developer Assessment

|  |  |
| --- | --- |
| Name | Klaas Rikhotso |

|  |  |
| --- | --- |
| Date | 15/04/2023 |

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?

|  |
| --- |
| Public methods or properties are accessible everywhere by the class, subclasses and outside the class, protected method are accessible by the class and it’s sub classes and private methods or properties are accessible only by the class. |
|  |
|  |

### 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?

|  |
| --- |
| Value: 4, result: 4 |
| Because $value is passed by reference &, which means if $foo is modified inside the function, the original $value will be modified as well. |
|  |

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

|  |
| --- |
| SLQ Injection is possible, to prevent this one can use pdo prepared statements and parameterized queries |
| $stmt = $pdo->prepare("SELECT \* FROM table WHERE id = :id"); |
| $stmt->execute([ id => $id ]); |

### 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 is cause by sending output to the browser before the header function, you can prevent this by setting the header before sending output. |
|  |
|  |

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

|  |
| --- |
| The class will throw and error because it’s using $this on a static method, so if the class is a static class, it must either use self:: where there is $this-> or remove the static keyword on the function doSomething so instances of the object Foo could be created. |
|  |
|  |

### 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".

function fizzBuzz(){

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

if(($i % 3 == 0) && ($i % 5 != 0)){

echo "Fizz\n";

} else if (($i % 3 != 0) && ($i % 5 == 0)){

echo "Buzz\n";

} else if(($i % 3 == 0) && ($i % 5 == 0)){

echo "FizzBuzz\n";

} else {

echo $i."\n";

}

}

}

echo fizzBuzz();

### **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);  
This code uses the preg\_replace function to manipulate the date string ‘08/26/2003’ to match the format DD/MM/YYYY using the regular expression '/([0-9]+)\/([0-9]+)\/([0-9]+)/'.

|  |
| --- |
|  |
|  |

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

|  |
| --- |
| $string = “<h1>This is a heading</h1>”; |
| preg\_replace(‘/<[^>]\*>/’, $string); |
|  |

### 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)

{

$message = "The input string is not a palindrone.";

$palindrome = "";

for($i = (strlen($word) - 1); $i >= 0; $i--)

{

$palindrome .= $word[$i];

}  
if(strtolower($word) == strtolower($palindrome))

{

$message = "The input string is a palindrone.";

}

return $message;

}

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

### Considering message\_text stores a combination of html and text. 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><?php echo $message->message\_text; ?></div> The security risk here is Cross-Site-Scripting, this could be prevented by sanitizing the output to the browser by using the php function htmlspecialchars($message->message\_text, ENT\_QUOTES); \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Write an inner join for the following tables

|  |
| --- |
| SELECT u.\*, a.\*  FROM User as u  RIGHT INNER JOIN Address as a on a.UsrKey = u.UserKey |
|  |

### Complete the JS function below that validates the conditions of a password: 1. The password must be greater then 7 characters 2. The first character must be a capital letter 3. The password must contain at least one number

function isPasswordValid($password) {

var isValid = false;

// Write you logic here

let regex = /^(?=.\*[A-Z])(?=.\*\d).{7,}$/;

if(regex.test(password)){

isValid = true;

}

return isValid;

}