1)What is the impact on performance in php7.4?

- PHP 7.4 Brings **Spread Operator** in Array Expression
- A significant advantage of **Spread operator** is that it supports any traversable objects, while the `array_merge` function only supports arrays.
- In PHP, anonymous functions are considered to be quite verbose and difficult to implement and maintain. This RFC proposes the introduction of the shorter and clearer syntax of the **arrow functions** (or short closures), that should allow us to clean up significantly our PHP code.

  Example:

```php
<?php
    $a = [1, 2, 3, 4, 5];
    $b = array_map(function ($n){
        return ($n * $n * $n);
    }, $a);
    print_r($b);

    echo '<br>With Arrow Function : <br>';
    $a = [1, 2, 3, 4, 5];
    $b = array_map(fn($n)=>($n * $n), $a);
    print_r($b);
?>
```

- **Typed Properties 2.0**

  In the new 7.4, PHP is able to support the following type list:
  `bool, int, float, string, array, object, iterable, self, parent`

2)write 4 different examples of spread operator?

-

```php
<?php
$animals = ['dog', 'cat'];
$animalkingdom = ['lion', 'elephant', ...$animals, 'giraffe'];
print_r($animalkingdom);
?>
```

```
Array
(
    [0] => lion
    [1] => elephant
    [2] => dog
    [3] => cat
    [4] => giraffe
)
```

-

```php
<?php
$num1 = [1, 2, 3];
$num2 = [...$num1]; // [1, 2, 3]
$num3 = [0, ...$num1]; // [0, 1, 2, 3]
$num4 = array(...$num1, ...$num2, 111); // [1, 2, 3, 1, 2, 3, 111]
$num5 = [...$num1, ...$num1]; // [1, 2, 3, 1, 2, 3]
print_r($num5);
?>
```

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 1
    [4] => 2
    [5] => 3
)
```

- **Not only that, but you can also use it in a function. Check out this example:**

```php
<?php
function getNum() {
  return ['a', 'b'];
}
$num6 = [...getNum(), 'c']; // ['a', 'b', 'c']


function arrGen() {
    for($i = 11; $i < 15; $i++) {
        yield $i;
    }
}
$num8 = [...arrGen()]; // [11, 12, 13, 14]
print_r($num8);
print_r($num6);
?>
```

```
Array
(
    [0] => 11
    [1] => 12
    [2] => 13
    [3] => 14
)
Array
(
    [0] => a
    [1] => b
    [2] => c
```

-

```php
<?php

function getAnimals(){
    return ['dog', 'cat', 'elephant'];
}
$num1 = [...getAnimals(), 'lion', 'tiger', 'giraffe'];
print_r($num1);
?>
```

```
Array
(
    [0] => dog
    [1] => cat
    [2] => elephant
    [3] => lion
    [4] => tiger
    [5] => giraffe
)
```

3)write 5 deprecated functions in php7.4?

- Nested ternary operators without explicit parentheses

```
<?php
1 ? 2 : 3 ? 4 : 5;   // deprecated
(1 ? 2 : 3) ? 4 : 5; // ok
1 ? 2 : (3 ? 4 : 5); // ok
      ?>
```

- Array and string offset access using curly braces

  The array and string offset access syntax using curly braces is deprecated. Use *$var[$idx]* instead of *$var{$idx}*.

- (real) cast and is_real() function

  The *(real)* cast is deprecated, use *(float)* instead.
  The is_real() function is also deprecated, use is_float() instead.

- restore_include_path() function

  The restore_include_path() function is deprecated. It can be replaced by *ini_restore('include_path')*.

- Using array_key_exists() on objects

  Using array_key_exists() on objects is deprecated. Instead either isset() or property_exists() should be used.

4)write three different way to implement dependency injectcion?

Constructor Injection:

In this approach, we can inject an object through the class constructor.

depend.php > ...

```php
<?php
    class Programmer {
        private $skills;
        public function __construct($skills){
        $this->skills = $skills;
        }
        public function totalSkills(){
            return count($this->skills);
        }
    }
    $createskills = array("PHP", "JQUERY", "AJAX");
    $p = new Programmer($createskills);
    echo $p->totalSkills();
?>
```

Setter Injection:

where you inject the object to your class through a setter function.

depend2.php > ...

```php
<?php
    class Profile {
        private $language;
        public function setLanguage($language){
            $this->language = $language;
        }
    }
    $profile = new Profile();
    $language = array("Hindi","English","French");
    $profile->setLanguage($language);

?>
```

Interface Injection

In this type of injection, an interface enforces the dependencies for any classes that implement it, for example:

```
interface someInterface {
    function getUsers(Database $database);
}
```

Now any class that needs to implement `someInterface` must provide `Database` dependency in their `getUsers()` methods.

Dependency Injection Container

We use container in real live projects because it contains many dependency injection and we use it for collecting it systematically.