

Utiliser la documentation PHP et le standard de codage PHP

Table des matières

I. Contexte	3
II. La documentation PHP	3
A. La documentation PHP	3
B. Exercice : Quiz	5
III. Les standards de codage PSR	6
A. Les standards de codage PSR	6
B. Exercice : Quiz	11
IV. Essentiel	11
V. Auto-évaluation	12
A. Exercice	12
B. Test	13
Solutions des exercices	14

I. Contexte

Durée : 1 h

Environnement de travail : un ordinateur connecté à Internet

Prérequis : aucun

Contexte

Développer une application ou un site internet nécessite de maîtriser un langage de programmation, de connaître les possibilités qu'il offre et ses limites, et tout cela en fonction de ses versions. Il est bien entendu difficile, voire impossible, de connaître toutes ces informations, d'où l'intérêt d'avoir une bonne documentation et de savoir l'utiliser ; elle sera votre meilleure alliée lors de vos développements.

Cependant, produire une application ou un site ne signifie pas seulement livrer un élément fonctionnel, il faut également que cela soit facilement évolutif pour que le code puisse aussi être appréhendé rapidement par une autre personne. C'est pourquoi il est nécessaire de respecter certains standards mis en place pour unifier et ainsi pouvoir simplifier la manière de coder : ce sont les standards PSR, que nous verrons en seconde partie de ce cours.

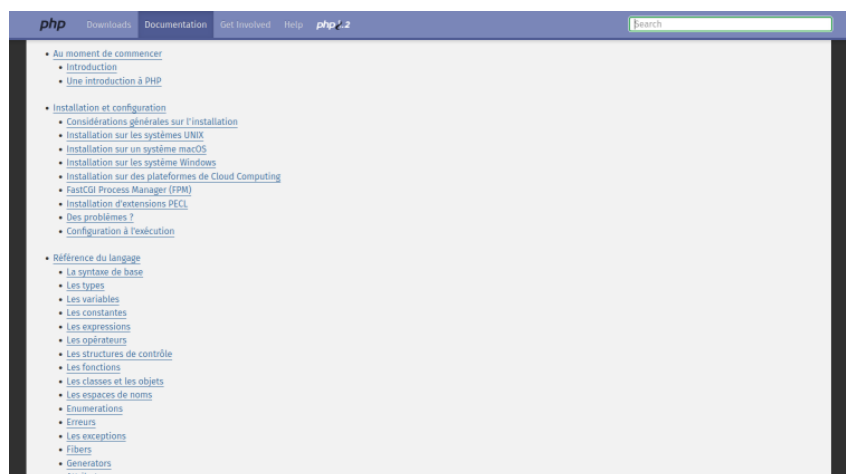
II. La documentation PHP

A. La documentation PHP

L'un des avantages de PHP est d'être open source ([github.com](https://github.com/php)¹). Il possède une communauté très importante qui offre, notamment, une documentation de qualité, très complète et traduite dans de nombreuses langues, dont le français, que vous pouvez retrouver ici : [php.net](https://www.php.net)².

C'est un langage qui évolue donc régulièrement, contrairement à d'autres langages, comme Java ou encore C++. Sa documentation est mise à jour en fonction, ce qui est un gage de qualité. La documentation aborde tous les points de PHP :

- L'histoire du langage, son installation (php - Installation et configuration³),
- Les fonctions disponibles (php - Référence des fonctions⁴),
- Les aspects de sécurité (php - Sécurité⁵).



1 <https://github.com/php>

2 <https://www.php.net/manual/fr/>

3 <https://www.php.net/manual/fr/install.php>

4 <https://www.php.net/manual/fr/funcref.php>

5 <https://www.php.net/manual/fr/security.php>

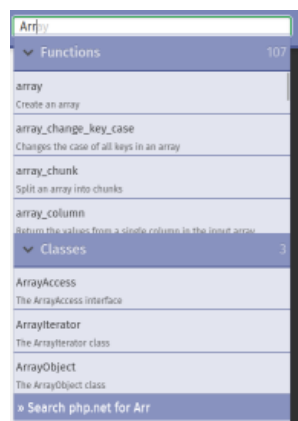
Page d'accueil de la documentation php.net

Source : Manuel PHP¹

Rechercher dans la documentation

Il n'est pas toujours facile de s'y retrouver dans les différents menus de navigation. Heureusement, **le moteur de recherche situé en haut à droite est extrêmement bien fait et performant**, c'est pourquoi nous vous encourageons à l'utiliser.

En effet, celui-ci propose de l'auto-complétion dès que nous commençons à taper quelques caractères, en regroupant les résultats par catégories (*Functions*, *Classes*, *Exceptions*, *Extensions*, *Other Matches*, etc.) : nous pouvons alors cliquer sur le résultat de notre choix ou continuer de taper pour affiner notre recherche.



Moteur de recherche

Source : PHP²

Vérifier la compatibilité

Lorsque nous consultons la documentation, il y a des points sur lesquels être vigilant. Il faut notamment vérifier la version de PHP et la validité d'une fonction.

Par exemple, au-dessous du nom de la fonction `addslashes`, nous retrouvons les **versions de PHP compatibles**, ce qui nous permet de savoir si nous pouvons l'utiliser sur notre projet.



Aperçu des informations de compatibilité

Source : PHP - addslashes³

¹ <http://www.php.net/manual/fr/>

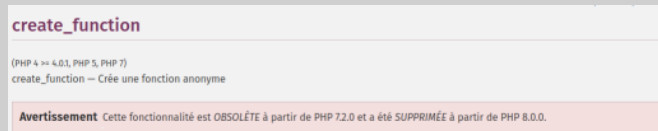
² <https://www.php.net/>

³ <http://www.php.net/manual/en/function.addslashes>

Conseil Repérer les éléments dépréciés

Concernant la validité d'une fonction, il est clairement indiqué dans la documentation PHP si celle-ci est obsolète (encart rouge) et à partir de quelle version de PHP c'est le cas.

Il est même parfois indiqué **si celle-ci a été supprimée** et à partir de quelle version de PHP elle l'a été.

**Message d'avertissement fonction obsolète**

Source : PHP - `create_function`¹

Voici une vidéo vous présentant plus en détail la documentation PHP :

Rappel À retenir

- Ne pas hésiter à utiliser la documentation PHP,
- Utiliser le moteur de recherche pour gagner du temps,
- Attention à bien vérifier la version de PHP compatible par rapport à votre version,
- Vérifier qu'il ne s'agit pas d'une fonctionnalité obsolète, voire supprimée.

B. Exercice : Quiz

[solution n°1 p.15]

Question 1

Depuis quelle version de PHP la fonction `addslashes` est-elle utilisable ?

- ☐ 4
- ☐ 5
- ☐ 6

Question 2

Sur quel site se trouve la documentation officielle PHP ?

- ☐ www.phpdoc.net
- ☐ www.php.net
- ☐ www.phpdoc.org

Question 3

La documentation PHP contient-elle des informations liées à la sécurité ?

- ☐ Oui
- ☐ Non, la sécurité est à gérer au niveau système de l'hébergement

¹ <http://www.php.net/manual/en/function.create-function>

Question 4

PHP est un langage de programmation très complet et très utilisé. Il a cependant un inconvénient : le code source est fermé et détenu par une société privée.

- ☐ Vrai
- ☐ Faux

Question 5

La fonction PHP `create_function` est obsolète depuis la version 7.2 de PHP. Est-elle cependant toujours utilisable en version 7.4 ?

- ☐ Oui
- ☐ Non

III. Les standards de codage PSR

A. Les standards de codage PSR

Mise en situation

Lors d'un développement, il est important de s'assurer de produire la meilleure qualité de code possible, que ce soit dans sa construction ou dans le respect des bonnes pratiques et des standards. Pour cela, des recommandations existent et sont regroupées sous le sigle PSR. Connaître et maîtriser les recommandations PSR va vous aider à atteindre cet objectif.

Définition Les recommandations PSR

PSR signifie **PHP Standards Recommendations** : il s'agit de recommandations émises pour la standardisation des concepts de programmation en PHP.

Chaque recommandation est proposée par les membres du **FIG** (*Framework Interoperability Group*) et votée selon un protocole établi.

Pour simplifier, le but du FIG est de permettre l'interopérabilité des différents frameworks en fournissant une base technique commune.

En suivant les PSR lors de nos développements, nous utilisons des pratiques et conventions communes et connues. Cela permet également à d'autres développeurs connaissant les PSR de plus facilement comprendre et reprendre le code. Ainsi, celui-ci sera plus facilement maintenable et pérenne.

Il est possible de consulter et de suivre l'état des PSR sur le site du FIG¹.

Complément La numérotation PSR

La numérotation correspond simplement à l'ordre dans lequel sont soumises les recommandations.

Comme les recommandations passent par plusieurs étapes (brouillon, revue, acceptée, abandonnée, etc.) et que certaines recommandations sont plus complexes que d'autres, il est parfaitement possible d'avoir une PSR acceptée ayant un numéro supérieur à une PSR toujours à l'état de brouillon. Nous allons à présent étudier certaines de ces recommandations.

¹ <https://www.php-fig.org/psr/>

PSR-0 / PSR-4 : autoloading

La PSR-0¹ aborde l'*autoloading*, mais cette PSR a été dépréciée au profit de la PSR-4², qui définit donc une nouvelle approche.

Cette PSR décrit une spécification pour l'auto-chargement de classes à partir de chemins de fichiers. Il est également décrit où placer les fichiers qui seront auto-chargés. Il s'agit donc notamment de respecter une syntaxe, une convention de nommage et un chemin lorsque nous créons nos classes PHP.

Par exemple, un projet comme Composer existe grâce à des conventions de nommage.

FULLY QUALIFIED CLASS NAME	NAMESPACE PREFIX	BASE DIRECTORY	RESULTING FILE PATH
\Acme\Log\Writer\File_Writer	Acme\Log\Writer	/acme-log-writer/lib/	/acme-log-writer/lib/File_Writer.php
\Aura\Web\Response\Status	Aura\Web	/path/to/aura-web/src/	/path/to/aura-web/src/Response/Status.php
\Symfony\Core\Request	Symfony\Core	/vendor/Symfony/Core/	/vendor/Symfony/Core/Request.php
\Zend\Acl	Zend	/usr/includes/Zend/	/usr/includes/Zend/Acl.php

Conventions de nommage PSR-4

Source : PHP-FIG³

PSR-1 : Basic Coding Standard

Cette PSR⁴ spécifie ce qui doit être considéré comme les éléments de codage standards qui sont nécessaires pour assurer un niveau élevé d'interopérabilité technique entre les codes PHP partagés, par exemple :

- Tout fichier PHP doit contenir `<?php` ou `<?='`
- Les noms des classes doivent être en **PascalCase**, c'est-à-dire avoir une majuscule à chaque mot (`class AuthTokenController`).
- Les constantes de classe doivent être déclarées en majuscules avec des séparateurs de soulignement (`public const ROLE_DEFAULT = ['ROLE_READER'];`).
- Les noms de méthodes doivent être déclarés en **camelCase**, c'est-à-dire que la première lettre est en minuscule, puis il y a une majuscule à chaque mot (`public function findOrCreateUser()`).

PSR-12 : Extended Coding Style

La recommandation PSR-12⁵ liste des règles et attentes communes sur la façon de formater le code PHP. Il s'agit d'un ensemble de lignes directrices à utiliser pour tous nos projets, ce qui facilite également la collaboration avec d'autres développeurs.

Il est par exemple indiqué :

- De suivre la PSR-1.
- De ne pas mettre de balise fermante `?>` si le fichier ne contient que du PHP.
- Il ne doit pas y avoir d'espace blanc à la fin des lignes.
- L'indentation du code doit être de 4 espaces pour chaque niveau, et la tabulation ne doit pas être utilisée.
- Toute accolade de fermeture ne doit pas être suivie d'un commentaire ni d'une déclaration sur la même ligne.
- Pour une classe, l'accolade d'ouverture doit aller sur sa propre ligne et l'accolade de fermeture pour la classe doit aller sur la ligne suivante, après le corps de la classe.

1 <https://www.php-fig.org/psr/psr-0/>

2 <https://www.php-fig.org/psr/psr-4/>

3 <http://www.php-fig.org/psr/psr-4/>

4 <https://www.php-fig.org/psr/psr-1/>

5 <https://www.php-fig.org/psr/psr-12/>

- De façon générale, les accolades d'ouverture doivent être sur leur propre ligne et ne doivent pas être précédées ni suivies d'une ligne blanche.
- De façon générale, les accolades de fermeture doivent être sur leur propre ligne et ne doivent pas être précédées d'une ligne blanche.
- Les noms de méthodes et de fonctions ne doivent pas être déclarés avec un espace après le nom de la méthode, et il ne doit pas y avoir d'espace après la parenthèse ouvrante ni avant la parenthèse fermante.

Exemple Quelques exemples issus de la PSR-12

Source : php-fig.org.

```

1 <?php
2
3 namespace Vendor\Package;
4
5 class ClassName
6 {
7     public $foo = null;
8     public static int $bar = 0;
9 }

```

```

1 <?php
2
3 namespace Vendor\Package;
4
5 class ClassName
6 {
7     public function fooBarBaz($arg1, &$arg2, $arg3 = [])
8     {
9         // method body
10    }
11 }

```

```

1 <?php
2
3 declare(strict_types=1);
4
5 namespace Vendor\Package;
6
7 class ReturnTypeVariations
8 {
9     public function functionName(int $arg1, $arg2): string
10    {
11        return 'foo';
12    }
13
14    public function anotherFunction(
15        string $foo,
16        string $bar,
17        int $baz
18    ): string {
19        return 'foo';
20    }
21 }

```


Exemple **Structure de type If**

Attention au placement des parenthèses, des espaces et des accolades : nous remarquons aussi que `else` et `elseif` sont sur la même ligne que l'accolade de fermeture précédente.

```
1 <?php
2
3 if ($expr1) {
4     // if body
5 } elseif ($expr2) {
6     // elseif body
7 } else {
8     // else body;
9 }
```

Exemple **Structure de type Switch**

Toujours veiller au bon emplacement des accolades et parenthèses.

Notons également les indentations des `case` (1 niveau par rapport au `switch`) et des `break` (1 niveau par rapport au `case`). S'il n'y a pas de `break`, on le précise avec un commentaire : `// no break`.

```
1 <?php
2
3 switch ($expr) {
4     case 0:
5         echo 'First case, with a break';
6         break;
7     case 1:
8         echo 'Second case, which falls through';
9         // no break
10    case 2:
11    case 3:
12    case 4:
13        echo 'Third case, return instead of break';
14        return;
15    default:
16        echo 'Default case';
17        break;
18 }
```

Exemple **Structure de type while, for, foreach**

Une attention particulière doit encore être portée sur l'emplacement des accolades et parenthèses.

```
1 <?php
2
3 while ($expr) {
4     // structure body
5 }
6
7 for ($i = 0; $i < 10; $i++) {
8     // for body
9 }
10
11 foreach ($iterable as $key => $value) {
12     // foreach body
13 }
```

Complément Quelques spécificités

Nous allons évoquer quelques spécificités hors PSR, que nous rencontrons souvent dans le milieu professionnel :

- Coder en anglais (noms des variables, fonctions, classes, fichiers, etc.).
- Toujours donner un nom clair aux variables et fonctions (`$password`, `public function createUser()`).
- Commenter son code (en anglais également).
- Toujours utiliser l'égalité / inégalité stricte, triple égal au lieu de double égal et point d'exclamation et double égal au lieu de point d'exclamation et signe égal (`if ($isValid === true)` ou `if ($isValid !== true)`).
- Éviter d'utiliser des variables globales.
- Pour les chaînes de caractères, il est préférable d'utiliser les simples quotes (`$message = 'Bienvenue sur notre site';`). Lors de l'utilisation de doubles quotes, PHP va vérifier si la chaîne contient ou non des variables et va les remplacer par leur valeur. Le traitement sera donc plus lent qu'avec de simples quotes.

Conseil Les PSR au quotidien

Il est important de maîtriser au mieux les normes PSR afin de fournir un code d'une qualité optimale.

Néanmoins, l'humain n'est pas une machine. Ces normes évoluent quotidiennement, et peuvent tout simplement être compliquées à retenir ou à appliquer de façon systématique, même si de nombreux automatismes arriveront très vite.

Au quotidien, certains outils seront des alliés essentiels afin d'appliquer au mieux ces normes :

- La plupart des éditeurs de code proposent des options de « *re-formatage du code* » afin de parer les oublis. Ces options sont puissantes et complètes, il ne faut pas hésiter à en faire usage.
- Il est de plus en plus courant de retrouver, sur un projet, la présence de scripts permettant l'analyse du respect des standards, ainsi que la correction automatique et systématique au besoin. Ces scripts permettent de conserver une qualité de code constante. L'un des plus connus est par exemple PHP-CS-Fixer (Github¹), mais il en existe d'autres (PHPMD², PHPSTAN³, etc.). Vous pouvez en trouver d'autres sur le site PHP Quality Assurance⁴. La qualité du code est une préoccupation importante dans beaucoup d'équipes.

Rappel À retenir

- **PSR signifie PHP Standards Recommendations** : il s'agit de recommandations émises pour la standardisation des concepts de programmation en PHP.
- Il est conseillé de les consulter régulièrement afin de rester informé des nouveautés ou mises à jour.
- Il est également important de développer en respectant les PSR, afin de respecter les standards et bonnes pratiques.

Complément

PHP Standards Recommendations : PHP-FIG⁵.

1 <https://github.com/PHP-CS-Fixer/PHP-CS-Fixer>

2 <https://phpmd.org/>

3 <https://phpstan.org/>

4 <https://phpqa.io/>

5 <https://www.php-fig.org/psr/>

B. Exercice : Quiz

[solution n°2 p.16]

Question 1

D'après la PSR1, quel doit être l'encodage d'un fichier PHP ?

- ☐ UTF8 sans BOM
- ☐ ISO 8859-1
- ☐ ISO 8859-15

Question 2

Quelle manière d'écrire une classe et une fonction est correcte par rapport à la PSR ?

- ☐ Classe **Mon_Nom_De_Classe** et fonction **monNomDeFonction**
- ☐ Classe **MonNomDeClasse** et fonction **monNomDeFonction**
- ☐ Classe **monNomDeClasse** et fonction **mon_nom_de_fonction**

Question 3

Quelle est la recommandation PSR qui aborde l'autoloading des classes en PHP ?

- ☐ PSR-1
- ☐ PSR-4
- ☐ PSR-12

Question 4

Quelle est la recommandation PSR qui spécifie les éléments de codage standards pour assurer l'interopérabilité technique entre les codes PHP partagés ?

- ☐ PSR-0
- ☐ PSR-1
- ☐ PSR-12

Question 5

Si je suis seul à travailler sur un projet, il n'est pas important de respecter les PSR, car je sais ce que je fais et comment utiliser mon code.

- ☐ Vrai
- ☐ Faux

IV. Essentiel

Vous disposez à présent de 2 outils cruciaux pour produire du code de qualité.

Le premier, **la documentation PHP**, dont le moteur de recherche peut vous faire gagner un temps considérable en appuyant les connaissances que vous avez déjà du langage. Attention cependant, quand vous trouvez une fonction dans la documentation et que vous voulez l'intégrer à votre projet, vérifiez toujours que la version de PHP est compatible avec la vôtre et que cette fonction n'est pas obsolète, ou pire, supprimée.

Le second outil est **les recommandations PSR** (PHP Standards Recommendations). Si vous les respectez en développant, alors votre code gagnera la qualité apportée par ces bonnes pratiques. Comme elles évoluent en continu, gardez toujours un œil dessus pour rester au fait des dernières nouveautés.

V. Auto-évaluation

A. Exercice

Vous récupérez une ancienne application d'un client utilisant la version 5.6 de PHP et qui a été codée sans respecter les PSR :

```

1 <?php
2
3 session_start();
4
5 class user {
6
7     // string
8     public $name;
9
10    // entier
11    public $age;
12
13    // string
14    public $defaultRole;
15
16    public $favoritePages = [];
17
18    public function __construct(
19 $name, $age
20 ){
21     $this->name = $name; $this->age = $age;
22
23     $this->defaultRole = isset($_SESSION['DEFAULT_ROLE']) ? $_SESSION['DEFAULT_ROLE'] :
24 'ROLE_DEFAULT'; }
25
26    public function add_to_favorites(
27 $link
28 )
29 {
30     if (filter_var($link, FILTER_VALIDATE_URL) && !in_array($link, $this->favoritePages))
31 { array_push($this->favoritePages, $link); return true; }
32
33     return false;}
34
35    public function remove_from_favorites( $link) {
36
37     if (filter_var($link, FILTER_VALIDATE_URL) && in_array($link, $this->favoritePages))
38 {
39         unset($this->favoritePages[array_search($link, $this->favoritePages)]); return
40 true;
41     }return false;}}
42
43 $user = new user('Eric', 45);
44
45 // true
46 echo $user->add_to_favorites( "https://www.google.com/");
47 // false

```

```

46 echo $user->add_to_favorites( "https://www.google.com/" );
47
48 // true
49 echo $user->remove_from_favorites("https://www.google.com/" );
50
51 // ROLE_DEFAULT
52 echo $user->defaultRole;
53
54 $_SESSION['DEFAULT_ROLE'] = 'ROLE_USER';
55 $user2 = new user( "Marie" , 40 );
56
57 // ROLE_USER
58 echo $user2->defaultRole;
59
60 unset($_SESSION[ "DEFAULT_ROLE" ]);
61
62 ?>

```

Question 1

[solution n°3 p.17]

Modifiez le code précédent pour lui faire respecter les normes de codage PSR actuelles.

Question 2

[solution n°4 p.18]

Faites évoluer ce code pour utiliser les dernières fonctionnalités de PHP 8.1.

B. Test

Exercice 1 : Quiz

[solution n°5 p.19]

Question 1

La numérotation des recommandations PSR indique :

- ☐ Leur importance, car plus le chiffre est élevé, moins ces recommandations sont importantes
- ☐ Leur ordre de déclaration
- ☐ Leur importance, car plus le chiffre est bas, moins ces recommandations sont précises

Question 2

Selon la documentation, nous savons que la fonction `intval` est accessible :

- ☐ Jusqu'à la version 8 de PHP
- ☐ Depuis la version 7, mais obsolète sur la dernière version
- ☐ À partir de la version 7

Question 3

Quelle affirmation est vraie ?

- ☐ Les PSR sont des éléments figés qui n'évoluent pas
- ☐ Les recommandations en statut « *accepté* » seront valides pour toujours
- ☐ Les recommandations évoluent avec le temps et le langage

Question 4

D'après la norme PSR-12, l'omission en fin de fichier d'un tag ?> lorsque celui-ci ne contient que du PHP est :

- ☐ Conseillée
- ☐ Déconseillée
- ☐ Obligatoire

Question 5

Concernant les mots-clés réservés, ceux-ci doivent être écrits en :


- ☐ Majuscules
- ☐ Minuscules
- ☐ camelCase

Solutions des exercices

Exercice p. 5 Solution n°1**Question 1**

Depuis quelle version de PHP la fonction `addslashes` est-elle utilisable ?


- ☒ 4
- ☐ 5
- ☐ 6

 La fonction `addslashes` existe depuis la version 4 de PHP.

Question 2

Sur quel site se trouve la documentation officielle PHP ?


- ☐ www.phpdoc.net
- ☒ www.php.net
- ☐ www.phpdoc.org

 Le site contenant la documentation PHP est www.php.net.

Question 3

La documentation PHP contient-elle des informations liées à la sécurité ?


- ☒ Oui
- ☐ Non, la sécurité est à gérer au niveau système de l'hébergement

 Oui, la documentation PHP contient bien des informations importantes concernant la sécurité des applications réalisées, que vous pouvez trouver à cette adresse : <https://www.php.net/manual/fr/security.php>.

Question 4

PHP est un langage de programmation très complet et très utilisé. Il a cependant un inconvénient : le code source est fermé et détenu par une société privée.

- ☐ Vrai
- ☒ Faux

 PHP est un langage open source et visible à cette adresse : <https://github.com/php>.

Question 5

La fonction PHP `create_function` est obsolète depuis la version 7.2 de PHP. Est-elle cependant toujours utilisable en version 7.4 ?

- ☒ Oui
- ☐ Non

- Q En version 7.4, la fonction `create_function` est bien obsolète, mais toujours existante, donc encore utilisable. Elle n'a été supprimée qu'à la version 8.0.

Exercice p. 11 Solution n°2

Question 1

D'après la PSR1, quel doit être l'encodage d'un fichier PHP ?

- ☒ UTF8 sans BOM
- ☐ ISO 8859-1
- ☐ ISO 8859-15

- Q Il est important de bien vérifier l'encodage de ses fichiers PHP pour ne pas avoir de problème d'affichage par la suite ou sur un autre projet reprenant votre code. Ils doivent être encodés au format UTF8 sans BOM.

Question 2

Quelle manière d'écrire une classe et une fonction est correcte par rapport à la PSR ?

- ☐ Classe **Mon_Nom_De_Classe** et fonction **monNomDeFonction**
- ☒ Classe **MonNomDeClasse** et fonction **monNomDeFonction**
- ☐ Classe **monNomDeClasse** et fonction **mon_nom_de_fonction**

- Q Comme indiqué dans la PSR1, un nom de classe doit être **PascalCase**, soit une majuscule au début de chaque mot, et un nom de fonction doit être **camelCase**, soit une majuscule au début de chaque mot sauf du premier.

Question 3

Quelle est la recommandation PSR qui aborde l'autoloading des classes en PHP ?

- ☐ PSR-1
- ☒ PSR-4
- ☐ PSR-12

- Q La recommandation PSR-4 décrit une spécification pour l'auto-chargement de classes à partir de chemins de fichiers et définit où placer les fichiers qui seront auto-chargés.

Question 4

Quelle est la recommandation PSR qui spécifie les éléments de codage standards pour assurer l'interopérabilité technique entre les codes PHP partagés ?

- ☐ PSR-0
- ☒ PSR-1
- ☐ PSR-12

- Q La recommandation PSR-1 spécifie ce qui doit être considéré comme les éléments de codage standards nécessaires pour assurer un niveau élevé d'interopérabilité technique entre les codes PHP partagés.

Question 5

Si je suis seul à travailler sur un projet, il n'est pas important de respecter les PSR, car je sais ce que je fais et comment utiliser mon code.

☐ Vrai

☒ Faux

Q Il est toujours important de respecter les recommandations des PSR pour plusieurs raisons. Tout d'abord, même si vous travaillez seul sur un projet, respecter les recommandations vous permet de vous y habituer et de prendre de bons réflexes qui vous serviront dans plusieurs cas : comprendre du code venant d'une application tierce que vous utilisez ; travailler à plusieurs sur un projet et avoir un code uniforme. De plus, si vous revenez plusieurs mois ou années plus tard sur votre code, il vous sera plus facile de vous y replonger.

p. 13 Solution n°3

```
1 <?php
2
3 session_start();
4
5 class User
6 {
7     public $name;
8
9     public $age;
10
11     public $defaultRole;
12
13     public $favoritePages = [];
14
15     public function __construct($name, $age)
16     {
17         $this->name = $name;
18         $this->age = $age;
19
20         $this->defaultRole = isset($_SESSION['DEFAULT_ROLE']) ? $_SESSION['DEFAULT_ROLE'] :
21 'ROLE_DEFAULT';
22     }
23
24     public function addToFavorites($link)
25     {
26
27         if (filter_var($link, FILTER_VALIDATE_URL) && !in_array($link, $this->favoritePages))
28         {
29             array_push($this->favoritePages, $link);
30
31             return true;
32         }
33
34         return false;
35     }
36
37     public function removeFromFavorites($link)
38     {
39         if (filter_var($link, FILTER_VALIDATE_URL) && in_array($link, $this->favoritePages))
40         {
41             unset($this->favoritePages[array_search($link, $this->favoritePages)]);
42         }
43     }
44 }
```

```

40
41         return true;
42     }
43
44     return false;
45 }
46 }
47
48 $user = new User('Eric', 45);
49
50 // true
51 echo $user->addToFavorites('https://www.google.com/');
52 // false
53 echo $user->addToFavorites('https://www.google.com/');
54
55 // true
56 echo $user->removeFromFavorites('https://www.google.com/');
57
58 // ROLE_DEFAULT
59 echo $user->defaultRole;
60
61 $_SESSION['DEFAULT_ROLE'] = 'ROLE_USER';
62 $user2 = new User('Marie', 40);
63
64 // ROLE_USER
65 echo $user2->defaultRole;
66
67 unset($_SESSION['DEFAULT_ROLE']);

```

p. 13 Solution n°4

```

1 <?php
2
3 session_start();
4
5 class User
6 {
7     // Depuis PHP 7.4, il est possible de typer les propriétés d'une classe
8     public string $defaultRole;
9
10    /** @var array<int, string> $favoritePages */
11    public array $favoritePages = [];
12
13    // Les paramètres d'entrée d'une fonction peuvent être typés avec les types string / int
14    // depuis PHP 7.0
15    // PHP 8.0 introduit la promotion de propriété de constructeur. Ces propriétés n'ont plus
16    // besoin d'être déclarées dans la class ni assignés dans le constructeur, c'est fait
17    // automatiquement
18    public function __construct(public string $name, public int $age)
19    {
20        // Depuis PHP 7.0, ce nouvel opérateur permet d'éviter la répétition dont nous
21        // disposions auparavant
22        $this->defaultRole = $_SESSION['DEFAULT_ROLE'] ?? 'ROLE_DEFAULT';
23    }
24
25    // Les paramètres de sortie d'une fonction peuvent être typés depuis PHP 7.0
26    public function addToFavorites(string $link): bool
27    {

```

```
24
25     if (filter_var($link, FILTER_VALIDATE_URL) && !in_array($link, $this->favoritePages))
26     {
27         array_push($this->favoritePages, $link);
28         return true;
29     }
30
31     return false;
32 }
33
34 public function removeFromFavorites(string $link): bool
35 {
36     if (filter_var($link, FILTER_VALIDATE_URL) && in_array($link, $this->favoritePages))
37     {
38         unset($this->favoritePages[array_search($link, $this->favoritePages)]);
39         return true;
40     }
41
42     return false;
43 }
44 }
45
46 $user = new User('Eric', 45);
47
48
49 // PHP 8.0 introduit les arguments nommés en plus des arguments positionnels
50 echo $user->addToFavorites(link: 'https://www.google.com/');
51 // false
52 echo $user->addToFavorites(link: 'https://www.google.com/');
53
54 // true
55 echo $user->removeFromFavorites('https://www.google.com/');
56
57 // ROLE_DEFAULT
58 echo $user->defaultRole;
59
60 $_SESSION['DEFAULT_ROLE'] = 'ROLE_USER';
61 $user2 = new User('Marie', 40);
62
63 // ROLE_USER
64 echo $user2->defaultRole;
65
66 unset($_SESSION['DEFAULT_ROLE']);
```

Exercice p. 13 Solution n°5

Question 1

La numérotation des recommandations PSR indique :

- ☐ Leur importance, car plus le chiffre est élevé, moins ces recommandations sont importantes
- ☒ Leur ordre de déclaration
- ☐ Leur importance, car plus le chiffre est bas, moins ces recommandations sont précises

Q Il est important de bien vérifier l'encodage de ses fichiers PHP pour ne pas avoir de problème d'affichage par la suite ou sur un autre projet reprenant votre code. Ils doivent être encodés au format UTF8 sans BOM.

Question 2

Selon la documentation, nous savons que la fonction `intdiv` est accessible :

- ☐ Jusqu'à la version 8 de PHP
- ☐ Depuis la version 7, mais obsolète sur la dernière version
- ☒ À partir de la version 7

Q Dans la documentation, il est indiqué sous le nom de la fonction : (PHP7, PHP8). Il n'est pas indiqué que cette fonction est obsolète, elle est donc toujours valide pour la dernière version de PHP. Elle est ainsi accessible depuis la version 7.

Question 3

Quelle affirmation est vraie ?

- ☐ Les PSR sont des éléments figés qui n'évoluent pas
- ☐ Les recommandations en statut « *accepté* » seront valides pour toujours
- ☒ Les recommandations évoluent avec le temps et le langage

Q Les PSR sont des recommandations en constante évolution. De nouvelles recommandations apparaissent, certaines « *acceptées* » sont invalidées au profit d'autres. Il est important de rester à jour et de suivre cette évolution pour être toujours en adéquation avec celles-ci.

Question 4

D'après la norme PSR-12, l'omission en fin de fichier d'un tag `?>` lorsque celui-ci ne contient que du PHP est :

- ☐ Conseillée
- ☐ Déconseillée
- ☒ Obligatoire

Q Le respect d'une PSR n'est pas obligatoire, car c'est une recommandation. Cependant, si l'on choisit de la respecter, alors il est obligatoire d'omettre le tag `?>` en fin de fichier ne contenant que du PHP (« The closing `?>` tag **MUST** be omitted from files containing only PHP »).

Question 5

Concernant les mots-clés réservés, ceux-ci doivent être écrits en :

- ☐ Majuscules
- ☒ Minuscules
- ☐ camelCase

Q Comme indiqué dans la PSR-12, les mots-clés réservés PHP doivent être écrits en minuscules.