Norme PHP – Web@cadémie

Responsables Pédagogiques

pedagowac@epitech.eu

Table des matières

Qu'est-ce que c'est ?

- Ce document regroupe les bonnes pratiques concernant la rédaction et le découpage d'un code PHP. Il se base sur la norme C-Epitech (https://intra.epitech.eu/file/public/norme.pdf), qui est une convention de programmation qui a été créée par cette école.
- Elle concerne :
  - La présentation globale (paragraphes).
  - La présentation locale (lignes).

La norme est une convention purement syntaxique, à ce titre elle ne peut être utilisée comme excuse si votre script ne fonctionne pas !

Pourquoi existe-t-elle ?

- Pour uniformiser l'écriture du code au sein de la Web@cadémie.
- Pour structurer le code.
- Pour faciliter la lecture du code.
- Pour éviter à vos chers pangolins de pleurer des larmes de sang !

Certains choix dans la norme peuvent paraître un peu trop restrictifs, mais n'oubliez pas que la norme est aussi un outil pédagogique, les 25 lignes sont là par exemple pour accentuer le fait que vous devez absolument structurer vos programmes.

Conseil : On ne remet pas à la norme ! On code à la norme ! C'est-à-dire qu'on écrit à la norme dès le début de son script.

Présentation générale

- Il doit y avoir une instruction par ligne.
- Une ligne ne doit pas excéder 80 colonnes. Cette règle s'applique également pour les commentaires.
- Une fonction ne doit pas excéder 25 lignes entre les accolades.

```
1 /*
2 ** Cette fonction fait 2 lignes
3 */
4 function hello_world()
5 {
6   echo "Hello World\n";
7   return (0);
8 }
```

- Un fichier ne doit pas contenir plus de 5 fonctions (sauf en cas de déclaration d'une classe).

Les commentaires

- Les commentaires peuvent se trouver dans tous les fichiers sources.
- Il ne doit pas y avoir de commentaires dans le corps des fonctions.
- Sauf contraintes liées à l'IDE, les commentaires sont commencés et terminés par une ligne seule. Toutes les lignes intermédiaires s'alignent sur elles, et commencent par '**'.

```
 1 /*
 2 ** cette fonction calcule ....
 3 */
 4 function func_mort_de_rire()
 5 {
 6 }
 7
 8 /*
 9 ** Correct
10 */
11
12 /*
13 * Incorrect
14 */
```

Indentation générale

- L'indentation sera celle obtenue au moyen de la touche Tab (large de 2 espaces).

```
function func($var)
{
  while ($var < 10)
  {
    if ($var == 5)
    {
      $var = 0;
      return (TRUE);
    }
    $var++;
  }
  return (FALSE);
}
```

Nomination

- Les objets (variables, fonctions, classes, fichiers ou répertoires) doivent avoir les noms les plus explicites ou mnémoniques.
- Les abréviations sont tolérées dans la mesure où elles permettent de réduire significativement la taille du nom sans en perdre le sens.
- Tous les identifiants (fonctions, variables, classes, etc...) doivent être en anglais.
- Les noms des variables et des fonctions, doivent être écrit en snake_case : $my_string, function show_string.
- Les noms des classes doit être écrit en CamelCase : class MyClass.
- Un nom de globale doit commencer par g_.
- Toute utilisation de variable globale doit être justifiée.

```
$g_global_lyrics = "My little poney bring me to life!";

class HipHopSinger
{
  private $lyrics = "Put your hands up!";

  public function sing_a_song()
  {
    $full_song_lyrics = $g_global_lyrics . ' ' . this->lyrics;
    echo $full_song_lyrics;
  }
```

```
12 }
```

Déclarations / Affectations

- Abuser du mot clé « static » pour faire des variables globales est interdit. Toute variable statique doit être justifiée.

Structures de contrôle

- Une structure de contrôle sera toujours suivie d'un retour à la ligne.

```
 1 if ($cp) return ($cp); /* Incorrect */
 2
 3 if ($cp) {return ($cp);} /* Incorrect */
 4
 5 if ($cp) /* Incorrect */
 6   return ($cp);
 7
 8 if ($cp) { /* Admis */
 9   return ($cp);
10 }
11
12 if ($cp) /* Correct */
13 {
14   return ($cp);
15 }
```

Paramètres

- Les virgules ne sont autorisées que dans ce contexte.
- Une fonction prendra au maximum 4 paramètres.

```
1 function func($p1, $p2, $p3)
2 {
```

```
3 }
4
5 function func($p1,
6          $p2,
7          $p3)
8 {
9 }
```

Espaces

- Un espace doit être présent à la suite de chaque virgule.
- Il ne doit pas y avoir d'espace entre le nom d'une fonction et la '('.
- Un espace doit apparaître entre un mot clé (avec ou sans argument) et la '(' ou le ';' dans le cas d'un return.
- Il n'y a pas d'espace après un opérateur unaire.
- Tous les opérateurs binaires et ternaires sont séparés des arguments par un espace de part et d'autre.
- Une ligne vide doit séparer les définitions de fonctions.

```
1 if (!cp)
2 {
3   exit(1);
4 }
5 return (cp ? cp + 1 : cp + 2);
```

« return » est un mot clé.

Mots clés interdits

- goto

Le mot clé « goto » est interdit pour ne pas casser l'exécution logique du script.

Déclaration d'une classe

```php
class Toto
{
  private $name;

  public __construct($name)
  {
    $this->name = $name;
  }

  public function show_name()
  {
    echo $this->name;
  }
}
```

Vous vous devez de spécifier la visibilité de tous les membres de la classe.