



# Module 3

## Organisation du code

# Contenu du module

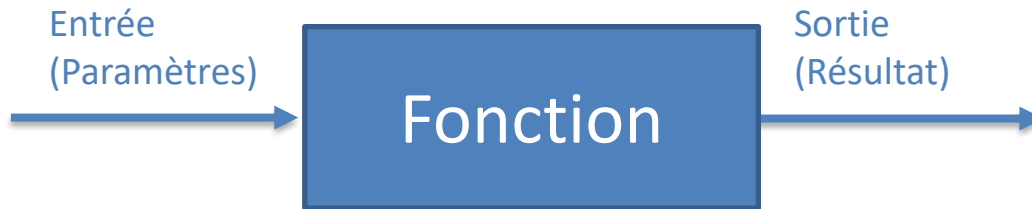
- Les fonctions
  - Déclaration et usages
  - Paramètres et retour
  - Déclaration des types de données de retour des fonctions en PHP 7 : Type Hints.
- Les fonctions intégrées du langage.
  - Gestion des chaînes de caractères.
  - Manipulation de dates.
- Structuration des applications.
  - Création de bibliothèques.
  - Importation de fichiers et de scripts.

A black and white photograph of a person in a suit, with their hands cupped together holding a large, stylized cloud graphic. The cloud has a white outline and a blue gradient fill. A dark blue horizontal band is overlaid across the middle of the cloud, containing the text 'Les fonctions'.

# Les fonctions

# Utilité des fonctions

- En programmation, les fonctions permettent de factoriser un ensemble d'instructions.
  - L'objectif étant la réutilisation !
- Cette séquence d'instruction est le plus souvent configurable par des paramètres d'entrées.
- La fonction peut produire ou non un résultat final qui sera collecté par le code qui appelle la fonction.



# Déclaration d'une fonction

- La déclaration d'une fonction est constituée des éléments suivants :
  - Le mot clé `function`
  - L'identificateur de la fonction (son nom !)
  - Une paire de parenthèses
  - Eventuellement des paramètres sous forme de variables. Ils seront exprimés entre les parenthèses.
- Ceci constitue la **signature** de la fonction.
- Un bloc est ensuite créé pour l'**implémentation** de la fonction (son code interne)

The diagram shows a function declaration in PHP: `function addition($operande1, $operande2) { $resultat = $operande1 + $operande2; return $resultat; }`. Annotations with arrows point to different parts: 'Identificateur' points to 'addition', 'Paramètres' points to '(\$operande1, \$operande2)', and 'Retour de la fonction' points to the closing curly brace '}'.

```
function addition($operande1, $operande2) {  
    $resultat = $operande1 + $operande2;  
    return $resultat;  
}
```

Retour de la fonction

# Utilisation d'une fonction

- Pour utiliser une fonction, il suffit de l'appeler par son identificateur tout en spécifiant des valeurs pour les paramètres.
- Le résultat produit en sortie peut être associé à une variable, ou exploité par une autre fonction, ...


```
$somme = addition(56, 14);
```

- La portée d'une fonction est le script qui la contient.
  - Donc si une fonction est contenue dans un fichier différent de celui où elle est utilisée, le fichier de la fonction doit être importé ! (Notion vue plus loin)

# Les paramètres

- Lorsqu'une fonction déclare des paramètres dans sa signature, il doivent recevoir des valeurs au moment de l'appelle de la fonction !

```
$somme = addition(56, 14);
```



```
function addition($operande1, $operande2) {  
    $resultat = $operande1 + $operande2;  
    return $resultat;  
}
```

The diagram shows two blue arrows originating from the function call `addition(56, 14)` in the code above. One arrow points from the first parameter `56` to the parameter `$operande1` in the function definition. The other arrow points from the second parameter `14` to the parameter `$operande2` in the function definition.

- Les paramètres correspondent à des **variables locales à la fonction**.
  - Ces variables ne sont visibles et utilisables qu'à l'intérieur du **corps** (le bloc) de la fonction.

# Le retour d'une fonction

- Le retour d'une fonction est le résultat qu'elle produit à la fin de l'exécution de son code d'implémentation.
  - Certaines fonctions n'ont pas de retour.
- Ce retour met fin à l'exécution de la fonction.
  - Les instructions situées après cette instruction de retour ne seront donc pas exécutées !
- Le retour de la fonction est introduit par le mot clé return.

```
function addition($operande1, $operande2) {  
    $resultat = $operande1 + $operande2;  
    return $resultat;  
}
```



# Les paramètres optionnels

- En PHP, il est possible de définir des fonctions possédant des paramètres optionnels.
  - Il n'est donc pas obligatoire de leur attribuer de valeurs au moment de l'appel de la fonction.
- Le principe consiste à attribuer une valeur par défaut à ces paramètres. Si une valeur est donnée au moment de l'appel, alors elle est utilisée, sinon, c'est la valeur par défaut.
  - ATTENTION : De manière évidente, les paramètres optionnels doivent se situer en fin de signature !

```
function maFonction($param1, $param2=0, $param3=0) {  
    $resultat = $param1 + $param2 + $param3;  
    return $resultat;  
}  
  
$somme = maFonction();           // Erreur ! $param1 doit être fourni  
$somme = maFonction(10);        // Renvoi 10  
$somme = maFonction(10, 20);    // Renvoi 30  
$somme = maFonction(10, 20, 30); // Renvoi 60
```

# Les paramètres en nombre variable

- Dans la signature d'une fonction, il est possible d'exprimer que celle-ci peut recevoir un nombre variable de paramètres.
  - Un seul paramètre est alors exprimé.
  - Son nom est préfixé d'une ellipse (...)
  - Le paramètre est alors exploitable sous forme d'un tableau

```
function moyenne(...$valeurs) {  
    $somme = 0;  
    foreach($valeurs as $val) {  
        $somme += $val;  
    }  
    $moyenne = $somme / count($valeurs);  
    return $moyenne;  
}  
  
$resultat = moyenne(15, 10, 12, 17, 8);    // Renvoi 12.4
```

# Paramètres nommés – PHP 8

- Les arguments nommés permettent de passer des arguments à une fonction en fonction du nom du paramètre, plutôt que de sa position.

- Il suffit d'ajouter le nom du paramètre devant sa valeur.

```
callFunction(name: $value);
```

- Avantages :

- Code plus compréhensible : auto-documenté
  - `array_fill(start_index: 0, num: 100, value: 50);`
- Les paramètres nommés sont indépendants de l'ordre
  - `array_fill(value: 50, num: 100, start_index: 0);`
- Permettent de spécifier uniquement les paramètres dont on veut changer la valeur de celle par défaut
  - `htmlspecialchars($string, double_encode: false);`
  - Au lieu de :
  - `htmlspecialchars($string, default, default, false);`

# « Type Hint » - PHP 7

- Le Type Hint de PHP permet d'améliorer la robustesse et la lisibilité du code en indiquant des types de données sur :
  - Les paramètres des fonctions / méthodes (v 7.0)
  - Le retour des fonctions / méthodes (v 7.0)
  - Les attributs (v 7.4) (*Evoqués plus loin...*)
- Le non-respect des types imposés déclenchera une **Fatal Error**, il s'agit donc bien d'une vraie contrainte apportée sur le code !

# Typage des paramètres

- L'apport du typage des paramètres permet de s'assurer que les données transmises aux méthodes/fonctions sont bien celles qu'elles attendent.

- Exemple :

```
function diviser(int $numérateur, int $denominateur) { ... }
```

- Si l'on souhaite autoriser la transmission de référence nulles, il faut préfixer le type d'un ?

- Exemple :

```
function calculate(array valeurs, ?object calculator) { ... }
```

# Typage du retour des fonctions

- Spécifier le type de retour des méthodes est tout aussi utile pour la robustesse des programmes.
- Cela permet notamment d'éviter de se retrouver avec des fonctions qui retournent des types de données différents en fonction de leur comportement.
  - Encore trop d'exemple dans l'API PHP malheureusement ...
- Exemple :

```
function diviser(int $numérateur, int $dénominateur) : float {  
    ...  
}
```
- De même que pour les paramètres, si la fonction peut également renvoyer null, il faut le préciser avec un ? devant le nom du type.
- Exemple :

```
function diviser(int $numérateur, int $dénominateur) : ?float {  
    ...  
}
```
- Le type **void** permet de préciser que la fonction ne renvoi pas de valeur

A black and white photograph of a person in a suit, with their hands cupped together holding a large, stylized white cloud shape. A blue horizontal band with a white cloud-like border is superimposed over the cloud, containing the title text.

# Les fonctions du langage PHP

# Les fonctions intégrées du langage

- En tant que langage de haut niveau, PHP propose de manière native tout un ensemble de fonctions directement exploitables dans les développements
- Elles couvrent un spectre de besoins assez large et sont toutes expliquées et illustrées dans la documentation officielle disponible sur [www.php.net](http://www.php.net)



# Fonctions sur les variables

- **empty**

- Indique si une variable est vide ou non
- `empty(mixed $var): bool`

- **isset**

- Indique si une ou plusieurs variables sont définies ou non
- `isset(mixed $var, mixed ...$vars): bool`

- **unset**

- Supprime une ou plusieurs variables
- `unset(mixed $var, mixed ...$vars): void`

- **var\_dump**

- Affiche des informations sur une ou plusieurs variables (type et valeur)
- `var_dump(mixed $value, mixed ...$values): void`

# Fonctions sur les types de données

- **is\_\***
  - Indique si la variable est du type donné par \* :
    - array, bool, callable, countable = dénombrable (tableau, par exemple),
    - float, double, real
    - int, integer, long = entier
    - iterable, null
    - numeric = entier ou nombre à virgule flottante ou chaîne contenant un nombre (entier ou décimal)
    - object, resource, scalar = type scalaire, string
- **strval**
  - Convertit une variable en chaîne
  - `strval(mixed $value): string`
- **floatval, doubleval** (alias)
  - Convertit une variable en nombre à virgule flottante
  - `floatval(mixed $value): float`
- **intval**
  - Convertit une variable en entier
  - `intval(mixed $value, int $base = 10): int`
- **boolval**
  - Convertit une variable en booléen
  - `boolval(mixed $value): bool`

# Quelques fonctions sur les tableaux...

- **count**
  - Compte le nombre d'éléments dans un tableau.
- **in\_array**
  - Teste si une valeur est présente dans un tableau.
- **array\_search**
  - Recherche une valeur dans un tableau.
- **array\_replace**
  - Remplace des valeurs dans un tableau.
- **[a|k][r]sort**
  - Trie un tableau (plusieurs variantes possibles).
- **explode**
  - Découpe une chaîne selon un séparateur et stocke les éléments dans un tableau.
- **implode**
  - Regroupe les éléments d'un tableau dans une chaîne à l'aide d'un séparateur.
- **max**
  - Retourne la plus grande valeur stockée dans un tableau.
- **min**
  - Retourne la plus petite valeur stockée dans un tableau.
- **str\_split**
  - Découpe une chaîne en morceaux de longueur fixe et stocke les éléments dans un tableau.
- **array\_column**
  - Retourne les valeurs d'une colonne d'un tableau multidimensionnel.
- **array\_key\_first**
  - Retourne la première clé d'un tableau.
- **array\_key\_last**
  - Retourne la dernière clé d'un tableau.

Liste exhaustive : [www.php.net/manual/fr/ref.array.php](http://www.php.net/manual/fr/ref.array.php)

# Quelques fonctions sur les chaînes de caractères...

- **strlen**
  - Retourne le nombre de caractères d'une chaîne
- **strtolower, strtoupper, ucfirst, ucwords, lcfirst**
  - Conversions minuscules/majuscules éventuellement limitées au(x) premier(s) mot(s)
- **strcmp, strcmpi**
  - Comparaison de chaîne (sensible à la casse ou non)
- **[s]printf, v[s]printf**
  - Mise en forme d'une chaîne (identique aux fonctions C équivalentes)
- **number\_format**
  - Mise en forme d'un nombre
- **[l|r]trim**
  - Suppression de caractères "blancs"
- **substr**
  - Extraction d'une sous-chaîne dans une chaîne
- **str[r][i]pos**
  - Recherche de la position d'une occurrence (caractère ou chaîne) à l'intérieur d'une chaîne
- **str[i]str, strrchr**
  - Extraction de la sous-chaîne dans une chaîne commençant à partir d'une certaine occurrence d'un caractère ou d'une chaîne
- **str\_[i]replace**
  - Remplacement des occurrences d'une chaîne par une autre chaîne
- **strtr**
  - Remplacement des occurrences d'un caractère par un autre caractère ou d'une chaîne par une autre chaîne
- **str\_contains**
  - Détermine si une chaîne contient une autre chaîne (PHP 8)
- **str\_starts\_with**
  - Détermine si une chaîne commence par une autre chaîne (PHP 8)
- **str\_ends\_with**
  - Détermine si une chaîne se termine par une autre chaîne (PHP 8)

Liste exhaustive : [www.php.net/manual/fr/ref.strings.php](http://www.php.net/manual/fr/ref.strings.php)

# Manipulation de dates

- PHP ne gère pas les dates avec un type de donnée spécifique
- Néanmoins, elles peuvent être manipulées, soit :
  - Sous la forme d'une chaîne de caractères
  - Sous la forme d'un timestamp Unix
    - Nombre de secondes écoulées depuis le 1er janvier 1970 01:00:00
- Il existe aussi plusieurs classes qui offrent des fonctionnalités avancées pour la manipulation des dates (classe `DateTime`) et des intervalles (classe `DateInterval`) sous une forme orientée objet.

# Quelques fonctions sur les dates...

- **checkdate**
  - Vérifie que trois entiers représentant le jour, le mois et l'année correspondent à une date valide.
- **date**
  - Convertit en chaîne une date donnée sous la forme d'un timestamp Unix.
- **strtotime**
  - Convertit en chaîne une date donnée sous la forme d'un timestamp Unix, en utilisant des caractéristiques locales.
- **getdate**
  - Stocke dans un tableau les différentes composantes d'une date donnée sous la forme d'un timestamp Unix.
- **date\_parse\_from\_format**
  - Stocke dans un tableau les différentes composantes d'une date donnée sous la forme d'une chaîne de caractères.
- **time**
  - Donne le timestamp Unix actuel.
- **mktime**
  - Crée un timestamp Unix à partir des différentes composantes d'une date.
- **microtime**
  - Donne le timestamp Unix actuel accompagné du nombre de microsecondes écoulées depuis la dernière seconde.
- **hrtime**
  - Retourne un temps écoulé à partir d'un instant arbitraire, avec une grande précision.
- **idate**
  - Donne les composantes d'une date fournie sous la forme d'un timestamp Unix.

A grayscale photograph of a person in a business suit, with their hands cupped together in front of them. Overlaid on the image is a large, stylized white cloud with a blue outline. A horizontal blue band cuts across the middle of the cloud, containing the title text.

# Structuration des applications

# Importation de fichiers et de scripts

- L'importation de scripts depuis d'autres est un besoin très fréquent en PHP
  - Pour rappel : la portée des éléments (fonctions, variables, ...) définis dans un script est le script lui-même !
- Pour pouvoir utiliser des fonctions définies dans un script à l'intérieur d'un autre, il faut d'abord inclure le script qui les contient
- Il est également parfois utile d'avoir recours à la composition de page
  - Technique qui consiste à assembler des fragments de code HTML pour construire une page complète et cohérente
  - Eviter de répéter plusieurs fois le même bandeau de menu ou pied de page dans plusieurs pages par exemple !
- Pour ces besoins :
  - `include()`, `include_once()`
  - `require()`, `require_once()`



# Usages

- `include()` et `include_once()`
  - Retournent 1 en cas de succès
  - FALSE en cas d'erreur
    - Elle génèrent une simple erreur de niveau E\_WARNING qui n'interrompt pas l'exécution du script.
- `require()` et `require_once()`
  - N'ont pas de code de retour
  - Provoquent une erreur fatale interrompant l'exécution du script !
- Le fichier inclus peut contenir du code HTML, du code PHP ou les deux
  - Les variables, constantes classes et fonctions, définies dans le fichier inclus sont utilisables dans le script appelant et réciproquement
  - Tout se passe comme s'il n'y avait qu'un seul script après l'inclusion !

# Redondance...

- `include()` et `require()`
  - L'inclusion est répétée autant de fois que la directive est utilisée
  - Pratique pour insérer du code HTML en boucle
    - Génération de lignes d'un tableau par exemple
  - Attention aux boucles !
- A savoir !
  - L'inclusion d'une fonction ou d'une classe ne peut se faire qu'une seule fois !
    - Sinon, erreur fatale !
  - La redondance est donc un problème dans le cas d'un script contenant des définitions de fonctions qui serait inclus plusieurs fois
- `include_once()` et `require_once()`
  - Garantissent qu'un fichier ne sera inclus qu'une fois, même si l'inclusion est appelée plusieurs fois !

# Bonnes pratiques !

- Utiliser les fonctions `*_once()` pour inclure des scripts qui contiennent des bibliothèques de code
  - Ensembles de fonctions, fichiers de classes, ...
- Utiliser les fonctions `include*()` pour les inclusions sans incidences sur l'exécution correct du code
  - Inclusion de bannières, publicités, ...
- Gérer correctement les erreurs sur les inclusions
  - Code de retour des fonctions `include*()`
  - Gestionnaires d'erreurs sur les fonctions `require*()`

# Travaux pratiques

