Chapitre 3
Utiliser les fonctions PHP

1. Préambule

L'objectif de ce chapitre est de présenter les fonctions les plus utiles dans le cadre du développement d'un site web.

PHP propose de nombreuses fonctions ; la description de chaque fonction est accessible en ligne sur le site www.php.net.

Version 8

Depuis la **version 8**, il est possible de passer des paramètres à une fonction en utilisant le nom du paramètre au lieu de sa position. Cette fonctionnalité est présentée dans le chapitre Écrire des fonctions et des classes PHP, mais elle peut être utilisée pour les fonctions natives du langage PHP, et donc pour les fonctions présentées dans ce chapitre. Par contre, dans ce chapitre, les noms réels des paramètres des fonctions ne sont pas présentés (ils sont traduits) ; pour les connaître, consultez la documentation en ligne des fonctions.

Depuis la **version 8.1**, passer la valeur `NULL` à un paramètre qui n'est pas explicitement optionnel est déprécié et génère donc une alerte de niveau `E_DEPRECATED`.

Exemple

```
<?php
$x = null;
$n = strlen($x);
?>
```

Résultat

```
Deprecated: strlen(): Passing null to parameter #1 ($string) of type string
is deprecated in /app/scripts/index.php on line 3
```

2. Manipuler les constantes, les variables et les types de données

2.1 Constantes

PHP propose un certain nombre de fonctions utiles sur les constantes :

Nom	Rôle
`defined`	Indique si une constante est définie ou non.
`constant`	Retourne la valeur d'une constante.

defined

La fonction `defined` permet de savoir si une constante est définie ou non.

Syntaxe

```
booléen defined(chaîne nom)
```

`nom` Nom de la constante.

La fonction `defined` retourne `TRUE` si la constante est définie et `FALSE` dans le cas contraire.

Exemple

```
<?php
// Tester si la constante CONSTANTE est définie.
$ok = defined('CONSTANTE');
if ($ok) {
  echo 'CONSTANTE est définie.<br />';
} else {
  echo 'CONSTANTE n\'est pas définie.<br />';
};
// Définir la constante CONSTANTE
define('CONSTANTE','valeur de CONSTANTE');
// Tester si la constante CONSTANTE est définie.
$ok = defined('CONSTANTE');
if ($ok) {
  echo 'CONSTANTE est définie.<br />';
} else {
  echo 'CONSTANTE n\'est pas définie.<br />';
};
?>
```

Résultat

```
CONSTANTE n'est pas définie.
CONSTANTE est définie.
```

constant

La fonction `constant` retourne la valeur d'une constante dont le nom est passé en paramètre.

Syntaxe

```
mixte constant(chaîne nom)
```

Avec :

`nom` Nom de la constante.

Cette fonction est pratique pour récupérer la valeur d'une constante dont le nom n'est pas connu a priori.

Exemple

```
<?php
// définir le nom de la constante dans une variable
$nomConstante = 'AUTRE CONSTANTE';
// définir la valeur de la constante
define($nomConstante,'valeur de AUTRE CONSTANTE');
// afficher la valeur de la constante
echo $nomConstante,' = ',constant($nomConstante);
?>
```

Résultat

```
AUTRE CONSTANTE = valeur de AUTRE CONSTANTE
```

D'autres fonctions permettent de connaître le type d'une constante (cf. section Manipuler les constantes, les variables et les types de données - Types de données).

2.2 Variables

PHP propose un certain nombre de fonctions utiles sur les variables :

Nom	Rôle
`empty`	Indique si une variable est vide ou non.
`isset`	Indique si une ou plusieurs variables sont définies ou non.
`unset`	Supprime une ou plusieurs variables.
`var_dump`	Affiche des informations sur une ou plusieurs variables (type et valeur).

empty

La fonction `empty` permet de tester si une variable est vide ou non.

Syntaxe

```
booléen empty(mixte variable)
```

`variable` Variable à tester.

`empty` retourne `TRUE` si la variable est vide et `FALSE` dans le cas contraire.

Une variable est considérée comme vide si elle n'a pas été affectée ou si elle contient une chaîne vide (""), une chaîne égale à 0 ("0"), un 0, `NULL`, `FALSE` ou un tableau vide.

La fonction `empty` peut aussi être utilisée pour tester si une expression est vide ou non.

Exemple

```
<?php
// Test d'une variable non initialisée.
$est_vide = empty($variable);
echo '$variable non initialisé<br />';
if ($est_vide) {
  echo '=> $variable est vide.<br />';
} else {
  echo '=> $variable n\'est pas vide.<br />';
}
// Test d'une variable contenant une chaîne vide.
$variable = '';
$est_vide = empty($variable);
echo '$variable = \'\'<br />';
if ($est_vide) {
  echo '=> $variable est vide.<br />';
} else {
  echo '=> $variable n\'est pas vide.<br />';
}
// Test d'une variable contenant une chaîne égale à 0.
$variable = '0';
$est_vide = empty($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($est_vide) {
  echo '=> $variable est vide.<br />';
} else {
  echo '=> $variable n\'est pas vide.<br />';
}
// Test d'une variable contenant 0.
$variable = 0;
$est_vide = empty($variable);
echo '$variable = ',$variable,'<br />';
```

```
if ($est_vide) {
  echo '=> $variable est vide.<br />';
} else {
  echo '=> $variable n\'est pas vide.<br />';
}
// Test d'une variable contenant une chaîne non vide.
$variable = 'x';
$est_vide = empty($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($est_vide) {
  echo '=> $variable est vide.<br />';
} else {
  echo '=> $variable n\'est pas vide.<br />';
}
?>
```

Résultat

```
$variable non initialisé
=> $variable est vide.
$variable = ''
=> $variable est vide.
$variable = '0'
=> $variable est vide.
$variable = 0
=> $variable est vide.
$variable = 'x'
=> $variable n'est pas vide.
```

isset

La fonction `isset` permet de tester si une ou plusieurs variables sont définies ou non.

Syntaxe

```
booléen isset(mixte variable[, ...])
```

`variable` Variable à tester (éventuellement plusieurs, séparées par une virgule).

`isset` retourne `TRUE` si la variable est définie et `FALSE` dans le cas contraire.

Si plusieurs paramètres sont fournis, la fonction retourne `TRUE` uniquement si toutes les variables sont définies.

Une variable est considérée comme non définie si elle n'a pas été affectée ou si elle contient `NULL`. À la différence de la fonction `empty`, une variable qui contient une chaîne vide (""), une chaîne égale à 0 ("0"), un 0, un `FALSE` ou un tableau vide, n'est pas considérée comme non définie.

Exemple

```
<?php
// Test d'une variable non initialisée.
$est_définie = isset($variable);
echo '$variable non initialisé<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
// Test d'une variable contenant une chaîne vide.
$variable = '';
$est_définie = isset($variable);
echo '$variable = \'\'<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
// Test d'une variable contenant une chaîne égale à 0.
$variable = '0';
$est_définie = isset($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
// Test d'une variable contenant 0.
$variable = 0;
$est_définie = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
// Test d'une variable contenant une chaîne non vide.
$variable = 'x';
$est_définie = isset($variable);
echo '$variable = \'',$variable,'\'<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
?>
```

Résultat

```
$variable non initialisé
=> $variable n'est pas définie.
$variable = ''
=> $variable est définie.
$variable = '0'
=> $variable est définie.
$variable = 0
=> $variable est définie.
$variable = 'x'
=> $variable est définie.
```

unset

La fonction `unset` permet de supprimer une ou plusieurs variables.

Syntaxe

```
unset(mixte variable[, ...])
```

`variable` Variable à supprimer (éventuellement plusieurs, séparées par une virgule).

Après suppression, la variable se trouve dans le même état que si elle n'avait jamais été affectée. L'utilisation de la fonction `isset` sur une variable supprimée retourne `FALSE` notamment.

Exemple

```
<?php
// Définir une variable.
$variable = 1;
// Afficher la variable et tester si elle est définie.
$est_définie = isset($variable);
echo '$variable = ',$variable,'<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
// Supprimer la variable.
unset($variable);
// Afficher la variable et tester si elle est définie.
$est_définie = isset($variable);
echo '$variable = ',$variable??'','<br />';
if ($est_définie) {
  echo '=> $variable est définie.<br />';
} else {
  echo '=> $variable n\'est pas définie.<br />';
}
?>
```

Résultat

```
$variable = 1
=> $variable est définie.
$variable =
=> $variable n'est pas définie.
```

Remarque

Affecter un 0 ou une chaîne vide à une variable ne la supprime pas.

var_dump

La fonction `var_dump` affiche des informations sur une ou plusieurs variables (type et contenu).

Syntaxe

```
var_dump(mixte variable[, ...])
```

`variable` Variable à afficher (éventuellement plusieurs, séparées par une virgule).

La fonction `var_dump` est surtout intéressante lors des phases de mise au point.

Exemple

```
<?php
// afficher les informations sur une variable non initialisée
$variable = NULL;
var_dump($variable);
// initialiser la variable avec un nombre entier
$variable = 10;
// afficher les informations sur la variable
echo '<br />';
var_dump($variable);
// modifier la valeur (et le type) de la variable
$variable = 3.14; // nombre décimal
// afficher les informations sur la variable
echo '<br />';
var_dump($variable);
// modifier la valeur (et le type) de la variable
$variable = 'abc'; // chaîne de caractères
// afficher les informations sur la variable
echo '<br />';
var_dump($variable);
?>
```