Cheatsheet : PHP

CheatSheet PHP : Guide complet, résumé, aide-mémoire, tutoriel et conseils clés pour apprendre et maîtriser PHP efficacement et rapidement.

MAJ en sept. 2025 Intermédiaire 13 chapitres

Cours Cheatsheet

Recherche...

Commentaire

```
// Je suis un commentaire

/* Je suis un commentaire
sur plusieurs lignes */
```

Debug

Infos détaillées (type, valeur, structure...) sur une variable.

```php
var_dump($maVariable);
```

Variable

Case mémoire stockant une donnée.

```php
$firstname = "Toto";
$firstname = "Tutu"; // Ecrase la valeur précédente

// Non modifiable
const RATIO = 6.55;
define("RATIO", 6.55); // équivalent
```

- Types principaux : texte, nombre et booléen
- `define()` ne peut pas être utilisée dans une classe

Affichage

Afficher du texte/variables.

```php
// Texte
echo "Hello World";

// Variable
$firstname = "Toto";
echo $firstname;
```

Concaténation

Joindre plusieurs chaînes de caractères et/ou variables.

```php
$firstname = "Toto";
echo "Hello " . $firstname; // "Hello Toto"
```

Opération

```php
$a = 4;
$b = 7;

$addition = $a + $b; // Addition (11)
$soustraction = $b - $a; // Soustraction (3)
$multiplication = $a * $b; // Multiplication (28)
$division = $b / $a; // Division (1.75)
$modulo = $b % $a; // Reste de la division (3)
```

- Si `nombre % 2 = 0` => pair et si `nombre % 2 = 1` => impair
- Opérations complexes => fonctions `sqrt()`, `exp()` ...

Tableau

Structure de données regroupant un ensemble de valeurs.

```php
// Tableau unidimensionnel (indéxé par 0, 1... n)
$fruits = ["Salade", "Tomate", "Oignon"];
echo $fruits[0]; // "Salade"

// Tableau unidimensionnel (associatif)
$user = [
  "firstname" => "John",
  "lastname" => "Doe",
  "premium" => true,
];
echo $user['lastname']; // "Doe"

// Tableau multidimensionnel
$fruits = [
  ["Salade", 1.90],
  ["Tomate", 2.75],
  ["Oignon", 1]
];
echo $fruits[1][1]; // 2.75
```

Premier élément d'un tableau à l'indice `0`

If, Else

Conditionne l'exécution d'un script.

```php
$dice = 5;
if ($dice == 1) {
  // Il vaut mieux relancer le dé...
} else if ($dice > 1 && $dice <= 3) {
  // C'est mieux ...
} elseif ($dice > 3 && $dice <= 5) {
  // Pas mal !
} else {
  // Super !
}
```

- Opérateurs de comparaison : `==`, `===`, `!=`, `!==`, `>`, `>=`, `<` et `<=`
- Opérateurs logiques : `!`, `&&` et `||`
- Il est possible d'imbriquer des conditions entre elles

Switch

Conditionne l'exécution d'un script.

```
$fruit = "Banane";
switch ($fruit) {
  case 'Banane':
    echo $fruit . " (1.93 € / kg)";
    break;
  case 'Pomme':
    echo $fruit . " (2.62 € / kg)";
    break;
  case 'Mangue':
  case 'Papaye':
    echo $fruit . " (6.17 € / kg)";
    break;
  default:
    echo "Désolé, nous ne vendons pas de " . $fruit;
}
```

- `case` déclenché si valeur strictement égale (`===`) à l'expression du `switch`
- Plusieurs `case` à la suite signifient « ou »
- `break` termine l'instruction
- `default` exécuté si aucune correspondance avec les `case`

For

Répète une instruction plusieurs fois.

```
// 0, 1, 2, 3 ... 10
for ($i=0; $i<=10; $i++) {
  echo $i
```

```
}
```

- `$i=0` : point de départ
- `$i<=10` : point d'arrivée
- `$i++` : pas (ou « step ») - raccourci pour `$i=$i+1`
- Alternative `foreach()` utile pour boucler sur des tableaux
- Il est possible d'imbriquer des boucles entre elles

While

Répète une instruction tant qu'une condition est vraie.

```
$n = 0;

while ($n < 10) {
  $n++;
}

echo $n; // 10
```

Il est possible d'imbriquer des boucles entre elles

Fonction

Une portion de code isolée exécutable sur demande.

```
// Déclaration
function isEven(int $number): bool {
  return $number % 2 === 0;
}

// Appels
echo isEven(7); // false
echo isEven(8); // true
```

- `$number` : paramètre de la fonction (typé en `int`)
- `:bool` indique que la fonction retourne un booléen
- `return ...` : valeur retournée par la fonction (facultatif)
- `7` et `8` : arguments (valeurs transmises à un paramètre)

Inclusions

Insère le contenu d'un fichier dans un autre.

```
// Si fichier à inclure non trouvé => warning (script continu son ex
include('path/to/my/file.php');
// Comme `include` + ne réinclut pas un fichier déja inclus auparava
include_once('path/to/my/file.php');

// Si fichier à inclure non trouvé => erreur (script arrêté)
require('path/to/my/file.php');
// Comme `require` + ne réinclut pas un fichier déja inclus auparava
require_once('path/to/my/file.php');
```

`include('path/to/my/file.php')` peut être écrit `include 'path/to/my/file.php'`

Paramètres GET

Transmet des informations au serveur afin de récupérer des ressources.

```
// Considérons l'URL https://domaine.com/page?param1=toto&param2=tut

// Récupère un paramètre GET
$_GET['param1']; // "toto"
$_GET['param2']; // "tutu"
```

- Paramètres `GET` transitent via l'URL
- Signe `?` devant le 1er paramètre et signe `&` devant chaque suivant
- Formulaire HTML envoyé en `GET` => `$_GET['attribut-name-du-champ']`
- `$_GET` = variable superglobale (accessible de partout)

Paramètres POST

Transmet des informations au serveur pour traitement.

```
// Récupère un paramètre POST
echo $_POST['demo'];
```

- Paramètres `POST` transitent via le corps de la requête HTTP
- Formulaire envoyé en `POST` => `$_POST['attribut-name-du-champ']`
- `$_POST` = variable superglobale (accessible de partout)

Session

Stockage temporaire côté serveur pour persister des données entre plusieurs pages.

```
// Démarrer une session / charger une session existante
session_start();

// Déclaration d'une variable de session
$_SESSION['username'] = "Toto";
// Affichage d'une variable de session
echo $_SESSION['username'];

// Détruire une variable de session spécifique
unset($_SESSION['username']);
// Détruire l'ensemble des variables de session
$_SESSION = array();
// Détruire la session en elle-même
session_destroy();
```

- `session_start()` (début du script PHP) avant d'utiliser les sessions
- `$_SESSION` = variable superglobale (accessible de partout)
- Session automatiquement détruite à la fermeture du navigateur

Connexion

Connexion à une base de données.

```
try {
  /*
    Paramètre 1 : chaîne de connexion
    Paramètre 2 : nom de l'utilisateur
    Paramètre 3 : mot de passe utilisateur
  */
  $db = new PDO(
    'mysql:host=localhost;dbname=nom_bdd;charset=utf8;port=3307',
    'root',
    ''
  );
} catch (PDOException $e) {
  die('Erreur : ' . $e->getMessage());
}
```

Si non précisée, la valeur du port par défaut est `3306`

Récupération

Récupérer des données.

```
$db = new PDO(...);
$query = $db->query("SELECT ...");
// Récupère plusieurs résultats
$articles = $query->fetchAll();
// Récupère un seul résultat
$article = $query->fetch();
```

- `query()` exécute une requête `SQL` de lecture
- `fetch()` récupère un seul résultat et `fetchAll()` récupère plusieurs résultats

Insertion

Insérer des données.

```php
$db = new PDO(...);
$query = $db->exec("INSERT INTO ...");
```

`exec()` exécute une requête `SQL` d'écriture

Modification

Modifier des données.

```php
$db = new PDO(...);
$query = $db->exec("UPDATE ...");
```

`exec()` : exécute une requête `SQL` d'écriture

Suppression

Supprimer des données.

```php
$db = new PDO(...);
$query = $db->exec("DELETE FROM ...");
```

`exec()` exécute une requête `SQL` d'écriture

Requête préparée

Sécuriser ses requêtes dynamiques (injections SQL).

```php
$db = new PDO(
  'mysql:host=localhost;dbname=nom_bdd;charset=utf8',
  'root',
  ''
);

// ⚠️ Injection SQL
$query = $db->exec("DELETE FROM articles WHERE id = " . $id);
```

```php
// ✅ Requête préparée
$query = $pdo->prepare("DELETE FROM articles WHERE id = :id");
$query->bindParam(':id', $id, PDO::PARAM_INT);
$query->execute();
```

- `:id` => marqueur indiquant l'emplacement d'un paramètre dynamique dans la requête
- `prepare()` + `execute()` viennent remplacer les méthodes `query()` ou `exec()`
- `bindParam()` permet d'attacher des paramètres aux marqueurs `:x`

Sécurisation XSS

Se protéger des failles XSS.

```php
$maVariable = "<script src='https://pirate.com/attaque.js'></script>

// ⚠️ Le script sera exécuté dans la page
echo $maVariable;
// Affiche <script src='https://pirate.com/attaque.js'></script>

// ✅ Le script sera converti en entité HTML et donc non exécuté
echo htmlspecialchars($maVariable);
// Affiche : &lt;script src=&#039;https://pirate.com/attaque.js&#039
```

Autoload

Chargement automatique des classes.

```php
spl_autoload_register(function (string $class): void {
  require($class . '.php');
});
```

Déclarer une classe

Représentation abstraite d'un élément partageant des caractéristiques communes.

```php
class Character {

  // Attributs (caractéristiques)

  private int $id;
  private string $pseudo;
  private int $xp;

  const XP_MAX = 1000;

  // Méthodes (actions/comportement)

  // Constructeur
  public function __construct(
    int $id,
    string $pseudo,
    int $xp
  ) {
    $this->id = $id;
    $this->pseudo = $pseudo;
    $this->xp = $xp;
  }

  // Getters (accesseurs)
  public function getId(): int {
    return $this->id;
  }

  public function getPseudo(): string {
    return $this->pseudo;
  }

  public function getXp(): int {
    return $this->xp;
  }

  // Setters (mutateurs)
  public function setPseudo(string $pseudo): void {
    $this->pseudo = $pseudo;
```

```
  }

  public function setXp(int $xp): void {
    $this->xp = $xp;
  }

  // Méthode perso
  public function infos(): string {
    return "Je suis " . $this->pseudo . " et j'ai " . $this->xp . "/
  }

}
```

- Constructeur : initialise les attributs d'un objet lors de sa création
- Getter : retourne la valeur d'un attribut
- Setter : modifie la valeur d'un attribut
- `$this` fait référence à l'objet courant
- `self` fait référence à la classe courante
- Encapsulation : accès aux attributs `private` contrôlé par des méthodes `public`
- Classe dans un fichier `.php` du même nom (en `PascalCase`)
- Jamais de `setId()` car un `id` est immuable
- Nom de constante en `UPPERCASE` (ex: `XP_MAX`)

Créer un objet

Créer un objet à partir d'une classe.

```
$perso1 = new Character(1, "Toto", 450);

// Accès à un attribut public (⚠️ encapsulation)
echo "Je suis " . $perso1->pseudo;
// Appel d'une méthode public
echo "Je suis " . $perso1->getPseudo();
echo "J'ai " . $perso1->getXp() . "/" . Character::XP_MAX . " XP";
```

- `new` permet de créer un objet à partir d'une classe (= instanciation)
- Instancier une classe revient implicitement à appeler `__construct()`
- L'opérateur `->` permet d'accéder à un attribut/une méthode
- Accéder à une constante : `NomClasse::NOM_CONSTANTE`

Visibilité

Détermine l'accès aux attributs et méthodes.

```
class Character {

  private string $attribut1;
  public string $attribut2;
  protected string $attribut3;

}
```

- `private` : accessible depuis la classe
- `public` : accessible de partout (classe + objets instanciés)
- `protected` : comme `private` + accessible depuis classes filles

Classe enfant

Classe qui hérite des attributs d'une classe parente.

```
class Wizard extends Character {

  private int $magicPower;

  public function __construct(
    int $id,
    string $pseudo,
    int $xp,
    int $magicPower
  ) {
    parent::__construct($id, $pseudo, $xp);
```

```
    $this->magicPower = $magicPower;
  }

  public function castSpell(): int {
    $minPower = $this->magicPower;
    $maxPower = $this->magicPower * 2;
    return random_int($minPower, $maxPower);
  }

}
```

- `parent` fait référence à la classe parente
- `extends` indique qu'une classe enfant hérite d'une classe parente

Classe parente

Modèle commun partagé par d'autres classes.

```
abstract class Character {

  // Attributs
  protected int $id;
  protected string $pseudo;
  protected int $xp;

  const XP_MAX = 1000;

  // Constructeur
  public function __construct(
    int $id,
    string $pseudo,
    int $xp
  ) {
    $this->id = $id;
    $this->pseudo = $pseudo;
    $this->xp = $xp;
  }
```

```
    // ...

}
```

- Classes enfants pourront accéder aux attributs `protected`
- classe `abstract` => ne peut pas être instanciée

Static

Accéder à un attribut/méthode sans instancier la classe.

```php
class Car {

  // ...

  public static int $counter = 0;

  public function __construct(
    int $id,
    string $brand,
    string $model
  ) {
    $this->id = $id;
    $this->brand = $brand;
    $this->model = $model;
    self::$counter++;
  }

  // ...

  public static function getCounter() {
    return self::$counter;
  }

}

echo Car::$counter;
echo Car::getCounter();
```

- `self::` pour accéder à un élément `static` depuis la classe
- `NOM_CLASSE::` pour accéder à un élément `static` depuis un objet

laConsole, plateforme d'e-learning dédiée au développement web.

M'abonner à la newsletter

status : **200**

E-LEARNING

Formations

Cheatsheets

Roadmaps

Boîte à outils

Blog

COMMUNAUTÉ

Faire ma veille

Le Shop

A propos

Contact

LÉGAL

Mentions légales

Politique de confidentialité