### Les flèches => et ->

#### Flèche =>

**La flèche =>** est utilisée dans plusieurs contextes en PHP, notamment dans les tableaux associatifs et les expressions match.

* **Tableaux associatifs** : Elle associe une clé à une valeur.

php

$array = [

'key1' => 'value1',

'key2' => 'value2',

];

* **Expression** match : Elle associe une condition à une valeur retournée.

php

$result = match ($variable) {

'case1' => 'result1',

'case2' => 'result2',

};

#### Flèche ->

**La flèche ->** est utilisée pour accéder aux propriétés et méthodes des objets.

php

class CoffeeMachine {

public function makeEspresso() {

return 'Espresso made';

}

}

$machine = new CoffeeMachine();

echo $machine->makeEspresso(); // Appelle la méthode makeEspresso de l'objet $machine

### Différences entre => et ->

* => : Utilisée pour **associer des clés à des valeurs** dans les **tableaux associatifs** et dans les **expressions match**.
* -> : Utilisée pour **accéder aux propriétés et méthodes des objets**.

Les flèches => et -> ont des utilisations distinctes selon qu'on travaille avec des tableaux ou des objets.

**A quoi servent les ::**

En PHP, les deux points :: sont utilisés pour accéder aux membres statiques (propriétés et méthodes) d'une classe. C'est ce qu'on appelle l'opérateur de résolution de portée.   
**Il existe deux cas d'utilisation principaux** :

## 1. Accéder à une propriété ou méthode statique depuis l'extérieur de la classe

php

ClassName::$staticProperty; // Accéder à une propriété statique

ClassName::staticMethod(); // Appeler une méthode statique

## **2. Accéder à une propriété ou méthode statique depuis l'intérieur de la classe**

php

class ClassName

{

private static $instance;

public static function getInstance()

{

if (is\_null(self::$instance)) {

self::$instance = new self();

}

return self::$instance; //self fait reference à la classe courante dans laquelle il est utilisé et sert à acceder aux methodes et attributs static

}

}

Dans cet exemple, self::$instance fait référence à la propriété statique $instance de la classe courante. self représente la classe elle-même. L'opérateur :: est également utilisé avec les **constantes de classe et les appels de méthodes ou propriétés statiques** de classes parentes ou d'autres classes.

php

ClassName::CONSTANT\_NAME; // Accéder à une constante de classe

ParentClass::staticMethod(); // Appeler une méthode statique d'une classe parente

En résumé, les deux points :: permettent d'accéder aux membres statiques d'une classe, que ce soit depuis l'intérieur ou l'extérieur de la classe. C'est une fonctionnalité clé de la programmation orientée objet en PHP.