Les fonctions fléchées en Dart

1. Présentation

- Les **fonctions fléchées en Dart**, également appelées "**fonctions lambda**" ou "**fonctions à flèche**", sont des fonctions anonymes qui peuvent être définies de manière concise à l'aide de l'opérateur =>. Ces fonctions sont couramment utilisées pour effectuer des opérations simples et sont utiles lorsque vous avez besoin d'une fonction temporaire pour effectuer un traitement rapide.
- Dart prend en charge une syntaxe abrégée pour des fonctions avec une seule expression. Cette syntaxe utilise une flèche au lieu de return.
- Dart a une syntaxe spéciale pour le corps de la fonction, qui n'est qu'une ligne. La fonction fléchée est représentée par le symbole **=>**. C'est une syntaxe abrégée pour toute fonction qui n'a qu'une seule expression.
- Considérez la fonction nommée suivante **add** qui additionne deux nombres :



> - *La fonction **flèche** nous permet de créer une fonction simplifiée composée d'une seule expression.*
> - *Dans Dart, nous avons une notation de flèche grasse (=>). Une grosse flèche est utilisée pour définir une seule expression dans une fonction.*

2. Syntaxe et utilisation

- La syntaxe de la fonction Dart flèche, ou lambda fonction.
- Dart a une syntaxe spéciale pour les fonctions dont le corps est uniquement une ligne.
- **`returnType functionName(parameters...) => expression;`**
- L'extrait de code ci-dessus contient les composants suivants :

  `ReturnType`: contient des types de données tels que `void`, `int`, `bool`, etc. **`functionName`**: indique le nom de la fonction. **`Params`**: représente l'ensemble de paramètres requis par la fonction. **`Expression`**: est la seule instruction à exécuter.

> *Lorsque le corps de la fonction ne comporte qu'une seule ligne, vous pouvez utiliser la grosse flèche au lieu des **accolades** et de l'instruction `return`.*

3. Application

1. App01

- Écrire un programme en Dart qui calcule le carré de chaque élément d'une liste donnée. Le programme doit inclure deux fonctions distinctes, l'une sous forme de fonction régulière et l'autre sous forme de fonction fléchée. Après avoir calculé les carrés, affichez les deux résultats à l'écran.
  Solution

```
void main() {
  List<int> numbers = [1, 2, 3, 4, 5];

  // Fonction régulière
  List<int> squaredNumbersRegular = squareListRegular(numbers);

  // Fonction fléchée
  List<int> squaredNumbersArrow = squareListArrow(numbers);
  print("Les éléments de la liste de départ : $numbers");
  print("Carrés des éléments de la liste (fonction régulière) :
$squaredNumbersRegular");
  print("Carrés des éléments de la liste (fonction fléchée)   :
$squaredNumbersArrow");
}

// Fonction régulière pour calculer le carré de chaque élément
List<int> squareListRegular(List<int> numbers) {
  List<int> squaredNumbers = [];
  for (int number in numbers) {
    squaredNumbers.add(number * number);
  }
  return squaredNumbers;
}

// Fonction fléchée pour calculer le carré de chaque élément
List<int> squareListArrow(List<int> numbers) => numbers.map((number) => number
* number).toList();
```

2. App02

apcpedagogie.com

- Les fonctions données ci-dessous peuvent être optimisées à l'aide de la grosse flèche dans Dart.

```
 void main() {
  trouverPerimetre(9, 6);
  var surfaceRec = trouverSurface(10, 6);
  print('La surface est $surfaceRec');
}

void trouverPerimetre(int longueur, int largeur) {
  var perimeter = 2 * (longueur + largeur);
  print('Le périmètre est $perimeter');
}

int trouverSurface(int longueur, int largeur) {
  return longueur * largeur;
}
```

Solution

```
void main() {
  trouverPerimetre(9, 6);
  var surfaceRec = trouverSurface(10, 6);
  print('La surface est $surfaceRec');
}
void trouverPerimetre(int longueur, int largeur) =>
  print('Le périmètre est ${2 * (longueur + largeur)}');
int trouverSurface(int longueur, int largeur) =>
   longueur * largeur;
```

3. **App03**

- Les fonctions données ci-dessous peuvent être optimisées à l'aide de la grosse flèche dans Dart.

```
// Une fonction sui calcule l'aire d'une triangle

void surfaceTriangle(int base, int height) {
  var area = 0.5 * base * height;
  print('La surface du triangle est: $area');
}
```

apcpedagogie.com

```
void main() {
  surfaceTriangle( 7, 12);
}
```

Solution

```
void surfaceTriangle(int base, int height) =>
  print('La surface du triangle est: ${0.5 * base * height}');
void main() {
  surfaceTriangle( 7, 12);
}
```

4. **App04**

   - Rédigez un programme en Dart qui génère les nombres premiers inférieurs à 100 en utilisant deux fonctions distinctes, l'une en tant que fonction standard et l'autre en tant que fonction fléchée. Après avoir identifié les nombres premiers, affichez les deux ensembles de résultats à l'écran en utilisant la méthode "stdout.write" et en les affichant chacune sur une seule ligne.
     Solution

```
import 'dart:io';

bool estPremier(int nombre) {
  if (nombre <= 1) {
    return false;
  }
  for (int i = 2; i * i <= nombre; i++) {
    if (nombre % i == 0) {
      return false;
    }
  }
  return true;
}

void main() {
  stdout.write("Nombres premiers (fonction standard) : ");
  for (int i = 2; i < 100; i++) {
    if (estPremier(i)) {
```

apcpedagogie.com

```
      stdout.write('$i ');
    }
  }

 bool Function(int) estPremierFleche = (int nombre) =>
    nombre > 1 &&
    !List.generate(nombre - 2, (index) => index + 2)
        .any((diviseur) => nombre % diviseur == 0);


  stdout.write("\nNombres premiers (fonction fléchée) : ");
  for (int i = 2; i < 100; i++) {
    if (estPremierFleche(i)) {
      stdout.write('$i ');
    }
  }
}
```

Cours Flutter

Source : https://apcpedagogie.com/utiliser-loperateur-fleche-dans-une-fonction-dart · téléchargé le 08/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com