Fiche de Procédure : Les Bases du Langage Dart

1. Qu'est-ce que Dart ?

- Dart est un langage de programmation orienté objet, développé par Google, optimisé pour le développement d'applications mobiles, web, serveur et bureau, notamment avec le framework Flutter.
- Il se distingue par une syntaxe claire, inspirée du C++, et une gestion moderne de l'orienter objet (classes, héritage, interfaces, mixins).

2. Structure de base d'un programme Dart

- L'exécution commence toujours par la fonction `main()` :

```
void main() {
  print('Bonjour, Dart !');
}
```

- Le fichier source porte l'extension `.dart`.

3. Déclaration des variables et types de base

Type	Exemple	Description
int	`int age = 25;`	Nombre entier
double	`double prix = 9.99;`	Nombre à virgule
String	`String nom = 'Marie';`	Chaîne de caractères
bool	`bool actif = true;`	Booléen (vrai/faux)
var	`var ville = 'Paris';`	Inférence automatique du type
List	`List<int> notes = [10, 12,Tableau (liste ordonnée)	
Map	`Map<String, int> ages = {'Paul': 30};`	Dictionnaire clé/valeur

- Dart est fortement typé, mais permet aussi l'inférence de type avec `var` ou `dynamic`.

4. Structures de contrôle

Conditionnelles

```
if (age >= 18) {
  print('Majeur');
} else if (age > 12) {
  print('Adolescent');
} else {
  print('Enfant');
}
```

Boucles

- For :

  ```
  for (int i = 0; i < 5; i++) {
    print(i);
  }
  ```

- While :

  ```
  int compteur = 0;
  while (compteur < 3) {
    print(compteur);
    compteur++;
  }
  ```

5. Fonctions

- Déclaration d'une fonction :

  ```
  void direBonjour(String nom) {
    print('Bonjour $nom !');
  }
  ```

- Appel :

```
direBonjour('Lucie');
```

- Une fonction peut retourner une valeur :

```
int addition(int a, int b) {
  return a + b;
}
```

6. Notions de base sur les classes (OOP)

- Dart est un langage orienté objet : tout est objet, même les types de base.
- Exemple de classe :

```
class Personne {
  String nom;
  int age;

  Personne(this.nom, this.age);

  void sePresenter() {
    print('Je m\'appelle $nom et j\'ai $age ans.');
  }
}

void main() {
  var p = Personne('Marie', 25);
  p.sePresenter();
}
```

- Encapsulation, héritage, polymorphisme et interfaces sont pleinement supportés.

7. Exemple complet

```
void main() {
  String aliment = 'pomme';
  if (aliment == 'pomme' || aliment == 'banane') {
    print('Ceci est un fruit.');
  } else if (aliment == 'carotte') {
    print('Ceci est un légume.');
  } else {
    print('Ceci n'est ni un fruit ni un légume');
  }
}
```

Cet exemple montre l'utilisation des variables, conditions et affichage dans la console.

8. Bonnes pratiques

- Utiliser des noms explicites pour les variables et fonctions[8].
- Indenter le code pour la lisibilité.
- Privilégier l'inférence de type (`var`) pour plus de clarté, mais utiliser les types explicites pour éviter les erreurs[9].
- Organiser le code en classes et fonctions réutilisables.

9. Pour aller plus loin

- Découvrir les collections avancées (List, Set, Map).
- Explorer les concepts avancés de l'OOP (mixins, interfaces, classes abstraites).
- Apprendre à utiliser Dart avec Flutter pour le développement mobile.
- Utiliser les packages et la gestion des dépendances via `pub.dev`.

Astuce :
Pour tester rapidement du Dart, utilisez l'éditeur en ligne officiel DartPad ou installez le SDK Dart sur votre machine.