Super et this en Dart

Super et this en Dart

Objectifs du cours

- Comprendre le rôle de **`super`** et this dans la programmation orientée objet en Dart.
- Savoir utiliser **`super`** pour accéder aux membres de la classe parente.
- Savoir utiliser this pour faire référence à l'objet actuel.
- Appliquer ces concepts dans des exemples pratiques et des exercices.

1. ### Définitions

    1. **`super`**

        - **`super`**: Un mot-clé utilisé pour accéder aux membres (attributs et méthodes) de la classe parente ou de la superclasse.
        - Le mot-clé **`super`** dans Dart est utilisé pour faire référence aux méthodes ou aux variables de l'objet de classe parent. En termes simples, il est utilisé pour faire référence aux propriétés et méthodes de la superclasse .
        - L'utilisation la plus importante du mot-clé **`super`** est de supprimer l'ambiguïté entre la superclasse et la sous-classe qui ont des méthodes et des variables portant le même nom.
        - Le mot-clé **`super`** est capable d'invoquer la méthode et les champs des objets parent, car lorsque nous créons une instance de la sous-classe dans Dart, une instance de la classe parent est également créée implicitement.

            1. **Accéder à la variable de classe parent**

```
class Animal {
   int count = 150;
}
```

apcpedagogie.com

```
class Dog extends Animal {
   int count = 70;
   void printNumber(){
    print("Afficher le contenu de la variable locale count:
$count");
    print("Afficher le contenu de la variable de la classe Animal
count:${super.count}");
   }
}

void main(){
   Dog obj= Dog();
   obj.printNumber();
}
//Output
//Afficher le contenu de la variable locale count: 70
//Afficher le contenu de la variable de la classe Animal count:150
```

> *Dans l'exemple ci-dessus, nous avons deux classes **Animal** et **Dog**, où **Animal** est la classe `parent` (ou `Superclass`) et **Dog** est la `classe enfant` (ou `sous-classe`). Il convient de noter que la variable nommée **count** est déclarée à la fois dans la **superclasse** et la **sous-classe**, et lorsque nous utilisons `super.count` elle fera référence à la **classe parent** (**superclasse**).*

2. **Accéder aux méthodes de la classe parent**

- En Dart, pour accéder aux méthodes de la classe parent, vous pouvez utiliser le mot-clé super. Le mot-clé super est utilisé pour se référer à la classe parent et accéder à ses propriétés et méthodes.
- Voici comment vous pouvez l'utiliser :

```
class Parent {
  void maMethode() {
    print("Méthode de la classe parent");
  }
}

class Enfant extends Parent {
```

apcpedagogie.com

```
  void maMethode() {
    super.maMethode(); // Appel de la méthode de la classe parent
    print("Méthode de la classe enfant");
  }
}

void main() {
  Enfant enfant = new Enfant();
  enfant.maMethode();
}
```

- Dans cet exemple, la classe **Enfant** hérite de la classe **Parent**. Lorsque vous appelez `super.maMethode()` à l'intérieur de la méthode `maMethode()` de la classe **enfant**, cela appelle la méthode `maMethode()` de la classe **parent**. Ensuite, le reste du code de la méthode de la classe **enfant** est exécuté.

2. **This**

- **this**: Un mot-clé utilisé pour faire référence à l'objet actuel.
- **exemple_this.dart**

```
// exemple_this.dart
// Démonstration complète du mot-clé `this` en Dart
// `this` : référence à l'objet (instance) courant

class CompteBancaire {
  String titulaire;
  double solde;
  static String nomBanque = "Banque Centrale"; // Propriété de classe
(pas d'instance)

  // ? 1. INITIALISATION RACCOURCIE (Initialising Formals)
  // `this.` devant le paramètre = assignation automatique à la
propriété
  CompteBancaire(this.titulaire, [this.solde = 0.0]);

  // ? 2. DÉSAMBIGUÏSATION (Paramètre vs Propriété)
  void deposer(double solde) {
    // `this.solde` = propriété de l'objet
    // `solde`     = paramètre de la méthode
```

apcpedagogie.com

```
    this.solde += solde;
    print("? Dépôt de ${solde}€. Solde : ${this.solde}€");
  }

  // ? 3. CHAÎNAGE DE MÉTHODES (Method Chaining)
  // Retourner `this` permet d'appeler plusieurs méthodes à la suite
  CompteBancaire retirer(double montant) {
    if (montant <= this.solde) {
      this.solde -= montant;
      print("? Retrait de ${montant}€ réussi.");
    } else {
      print("□ Fond insuffisants pour retirer ${montant}€.");
    }
    return this; // ? Retourne l'instance courante
  }

  // ? 4. PASSER L'INSTANCE COURANTE COMME ARGUMENT
  void transfererVers(CompteBancaire destinataire, double montant) {
    if (this.solde >= montant) {
      this.solde -= montant;
      destinataire.solde += montant;
      print("? Transfert de ${montant}€ vers
${destinataire.titulaire}");
    } else {
      print("□ Transfert impossible : solde insuffisant");
    }
  }

  // ? 5. `this` INTERDIT dans un contexte STATIQUE
  static void afficherBanque() {
    // print(this.nomBanque); // □ ERREUR DE COMPILATION : `this`
n'existe pas ici
    print("? Banque : ${nomBanque}"); // □ Accès direct à la propriété
static
  }

  // ? 6. EXPLICITE vs IMPLICITE
  void afficherInfo() {
    // `this.` est optionnel quand il n'y a pas de conflit de nom
    print("? ${titulaire} | ? ${solde}€");
    // Equivalent à : print("? ${this.titulaire} | ? ${this.solde}€");
  }
}
```

apcpedagogie.com

```
// ? BONUS : Constructeur redirecteur utilisant `this`
class Personne {
  String nom;
  int age;

  Personne(this.nom, this.age);

  // Ce constructeur appelle automatiquement le constructeur principal
  Personne.anonyme() : this("Inconnu", 0);

  void sePresenter() => print("Je suis ${this.nom}, ${this.age} ans.");
}

void main() {
  print("? === Exemples d'utilisation de `this` en Dart ===\n");

  // 1□□ Initialisation & Désambiguïsation
  print("? 1. Initialisation & désambiguïsation");
  var compte1 = CompteBancaire("Alice", 1000.0);
  compte1.deposer(200.0); // Le paramètre `solde` masque la propriété,
`this.solde` résout le conflit

  // 2□□ Chaînage de méthodes (retour de `this`)
  print("\n? 2. Chaînage de méthodes");
  compte1
    .retirer(300)
    .deposer(50)
    .retirer(100);

  // 3□□ Passer `this` à une autre instance
  print("\n? 3. Transfert entre comptes (`this` comme référence)");
  var compte2 = CompteBancaire("Bob", 500.0);
  compte1.transfererVers(compte2, 150.0);

  // 4□□ Contexte statique (`this` interdit)
  print("\n? 4. Méthode statique (pas de `this`)");
  CompteBancaire.afficherBanque();

  // 5□□ Explicite vs Implicite
  print("\n? 5. Accès explicite vs implicite");
  compte1.afficherInfo();
```

apcpedagogie.com

```
  // ? Bonus : Constructeur redirecteur
  print("\n? Bonus : Constructeur avec `this`");
  var inconnu = Personne.anonyme();
  inconnu.sePresenter();
}
```

3. **Exemple complet**

? Fichier 1 : parent.dart (Classe de base)

```
// parent.dart
// Définition de la classe parent avec méthodes et propriétés

class Parent {
  String nom;
  int age;

  // Constructeur de la classe parent
  Parent(this.nom, this.age);

  // Méthode à hériter
  void sePresenter() {
    print("Je suis ${nom}, j'ai ${age} ans. (Parent)");
  }

  // Méthode utilitaire
  String getInfo() {
    return "${nom} - ${age} ans";
  }

  // Méthode virtuelle (à override)
  void action() {
    print("Action générique du parent");
  }
}
```

? Fichier 2 : child.dart (Héritage avec super)

```
// child.dart
// Classe enfant qui étend Parent et utilise super
```

apcpedagogie.com

```
import 'parent.dart';

class Enfant extends Parent {
  String ecole;

  // Constructeur : appel obligatoire à super() si Parent n'a pas de
constructeur sans paramètre
  Enfant(String nom, int age, this.ecole) : super(nom, age);

  @override
  void sePresenter() {
    // Appel de la méthode parent avec super
    super.sePresenter();
    print("Je suis élève à : ${ecole}");
  }

  @override
  void action() {
    // On enrichit le comportement du parent
    super.action();
    print("L'enfant joue et apprend ??");
  }

  // Nouvelle méthode utilisant super pour accéder à une propriété parent
  void afficherInfoParent() {
    print("Info depuis parent : ${super.getInfo()}");
  }
}
```

? Fichier 3 : this_example.dart (Usage de this)

```
// this_example.dart
// Démonstration complète du mot-clé `this`

class Produit {
  String nom;
  double prix;
  static int compteur = 0; // Propriété statique

  // Initialising formals : syntaxe raccourcie avec this
  Produit(this.nom, this.prix) {
    compteur++;
```

apcpedagogie.com

```
    print("Produit créé : ${this.nom}"); // this optionnel ici car pas
d'ambiguïté
  }

  // Méthode avec paramètre portant le même nom qu'une propriété
  void appliquerRemonte(double prix) {
    // this.prix = propriété, prix = paramètre
    this.prix += prix;
    print("Nouveau prix de ${this.nom} : ${this.prix}€");
  }

  // Méthode qui retourne l'instance courante (chaînage)
  Produit appliquerRemise(double pourcentage) {
    this.prix *= (1 - pourcentage / 100);
    return this; // Permet le method chaining
  }

  // Méthode statique : `this` n'est PAS disponible ici □
  static void afficherCompteur() {
    // print(this.nom); // □ Erreur : this interdit dans un contexte statique
    print("Nombre de produits créés : ${compteur}");
  }
}

// Exemple d'utilisation du chaînage
void demoChaining() {
  var p = Produit("Ordinateur", 1000.0)
    .appliquerRemise(10)   // -10%
    .appliquerRemonte(50); // +50€
  print("Prix final : ${p.prix}€");
}
```

? Fichier 4 : combined_example.dart (this + super ensemble)

```
// combined_example.dart
// Combinaison avancée de `this` et `super` dans une même classe

import 'parent.dart';

class Employe extends Parent {
  double salaire;
  static const double TAUX_COTISATION = 0.15;
```

```
  // Constructeur avec this pour les propriétés de cette classe, super pour le parent
  Employe(String nom, int age, this.salaire) : super(nom, age);

  @override
  void sePresenter() {
    // super : accès au comportement parent
    super.sePresenter();
    // this : accès aux membres de cette instance
    print("${this.nom} gagne ${this.salaire}€/mois");
  }

  // Méthode utilisant à la fois this et super
  double calculerSalaireNet() {
    // this.salaire : propriété locale
    // super.getInfo() : méthode héritée (juste pour l'exemple)
    print("Calcul pour : ${super.getInfo()}");
    return this.salaire * (1 - TAUX_COTISATION);
  }

  // Factory constructor avec validation
  factory Employe.withMinimumSalary(String nom, int age, double salaire) {
    if (salaire < 1500) {
      print("⚠ Salaire trop bas, ajustement à 1500€");
      salaire = 1500;
    }
    // `this` n'est pas disponible ici (contexte statique du factory)
    return Employe(nom, age, salaire);
  }

  // Méthode de comparaison utilisant this
  bool estPlusAgeQue(Employe autre) {
    return this.age > autre.age; // this.optionnel mais explicite
  }
}
```

? Fichier 5 : main.dart (Point d'entrée et tests)

```
// main.dart
// Fichier principal pour exécuter tous les exemples
```

apcpedagogie.com

```
import 'parent.dart';
import 'child.dart';
import 'this_example.dart';
import 'combined_example.dart';

void main() {
  print("? === Exemple 1 : Héritage avec super ===");
  var enfant = Enfant("Léa", 10, "École Primaire");
  enfant.sePresenter();
  enfant.afficherInfoParent();
  enfant.action();

  print("\n? === Exemple 2 : Usage de this ===");
  demoChaining();
  Produit.afficherCompteur();

  print("\n? === Exemple 3 : this + super combinés ===");
  var employe = Employe.withMinimumSalary("Thomas", 35, 1400);
  employe.sePresenter();
  print("Salaire net : ${employe.calculerSalaireNet().toStringAsFixed(2)}€");

  var employe2 = Employe("Marie", 28, 2000);
  print("${employe.nom} est plus âgé que ${employe2.nom} ?
${employe.estPlusAgeQue(employe2)}");

  print("\n? === Exemple 4 : Accès explicite à super ===");
  class A {
    String valeur = "Classe A";
    void afficher() => print(valeur);
  }
  class B extends A {
    String valeur = "Classe B";
    void afficher() {
      print(this.valeur);   // "Classe B"
      print(super.valeur);  // "Classe A"
      super.afficher();     // Appel méthode parent
    }
  }
  B().afficher();
}
```

apcpedagogie.com

2. Types d'utilisation

- **`super`** est utilisé dans le contexte de l'héritage pour accéder ou modifier les membres de la classe parente.
- this est utilisé pour distinguer les membres d'un objet des paramètres ou des variables locales ayant le même nom.

1. **Utilisation de `super`**

1. **Activité d'exemple :**

- Définissez une classe parente appelée Véhicule avec un attribut vitesseMaximale et une méthode afficherVitesseMaximale() qui affiche la vitesse maximale du véhicule.
- Définissez une classe dérivée appelée Voiture qui hérite de la classe Véhicule. Surchargez la méthode **afficherVitesseMaximale()** pour afficher également le type de **véhicule** (voiture).
- Créez une instance de Voiture, appelez la méthode **afficherVitesseMaximale()** et observez les résultats.

2. **Exercice :**

- Définissez une classe parente appelée Forme avec un attribut couleur et une méthode afficherCouleur() qui affiche la couleur de la forme.
- Définissez une classe dérivée appelée Cercle qui hérite de la classe Forme. Surchargez la méthode afficherCouleur() pour afficher également le type de forme (cercle).
- Créez une instance de Cercle, appelez la méthode afficherCouleur() et observez les résultats.

2. **Utilisation de `this`**

1. **Activité d'exemple :**

- Définissez une classe Personne avec un attribut nom. Déclarez une méthode afficherNom() qui affiche le nom de la personne.

apcpedagogie.com

- Déclarez une autre méthode afficherNomAvecThis() qui utilise this pour afficher le nom de la personne.
- Créez une instance de Personne, appelez les deux méthodes et observez les résultats.

2. **Exercice :**

   - Définissez une classe Produit avec des attributs nom et prix. Déclarez une méthode afficherDetails() qui affiche le nom et le prix du produit.
   - Déclarez une autre méthode afficherDetailsAvecThis() qui utilise this pour afficher les détails du produit.
   - Créez une instance de Produit, appelez les deux méthodes et observez les résultats.

3. ## Exercice Pratique : Système de Gestion de Bibliothèque

   - **Code de départ**

```
// exercice_this_super.dart
// COMPLETER LES TODO SANS MODIFIER LA STRUCTURE EXISTANTE

class Document {
  String titre;
  String auteur;
  bool disponible;

  // TODO 1 : Initialisez les propriétés avec `this` (syntaxe
courte Dart)
  Document(String titre, String auteur, [bool disponible = true])
{
    // À compléter
  }

  void emprunter() {
    if (this.disponible) {
      this.disponible = false;
      print("? '${this.titre}' est maintenant emprunté.");
    } else {
      print(" '${this.titre}' est indisponible.");
    }
  }
```

apcpedagogie.com

```
  // TODO 2 : Méthode de chaînage. Retournez l'instance courante.
  Document reserver() {
    print("? Réservation de '${this.titre}' confirmée.");
    return ???; // Remplacez ???
  }

  String description() {
    return "'${this.titre}' par ${this.auteur} | ${this.disponible
? '□ Dispo' : '? Emprunté'}";
  }

  // TODO 3 : Méthode statique. Pourquoi `this` est-il interdit
ici ?
  static void afficherRegles() {
    // print(this.titre); // Décommentez pour voir l'erreur
    print("? Règle : Emprunt max 14 jours.");
  }
}

class Livre extends Document {
  int nbPages;
  String genre;

  // TODO 4 : Appelez le constructeur parent avec `super`
  Livre(String titre, String auteur, this.nbPages, this.genre)
      : super(???, ???) { // Passez les bons arguments
  }

  @override
  String description() {
    // TODO 5 : Utilisez `super.description()` + infos spécifiques
    return ???;
  }
}

class Magazine extends Document {
  String titre; // ⚠□ Shadowing intentionnel (masque
Document.titre)
  int numeroEdition;

  Magazine(String titreDoc, String auteur, this.numeroEdition,
this.titre)
```

apcpedagogie.com

```
      : super(titreDoc, auteur);

  void afficherDoubleTitre() {
    // TODO 6 : Affichez le titre local et le titre parent
    print("? Magazine (local) : ${???}");
    print("? Document (parent) : ${???}");
  }
}

void main() {
  print("?□ === Exercice Bibliothèque ===\n");

  // 1□□ Test Document
  var doc = Document("Dart pour tous", "Jean Dupont");
  doc.reserver().emprunter(); // Chaînage

  // 2□□ Test Livre (héritage + super)
  var livre = Livre("L'Apprenti Dart", "Marie Code", 320,
"Technique");
  print(livre.description());

  // 3□□ Test Magazine (shadowing + this/super)
  var mag = Magazine("MagTech", "Éditeur X", 42, "Dart Monthly");
  mag.afficherDoubleTitre();

  // 4□□ Test statique
  Document.afficherRegles();
}
```

- **Tâches à réaliser**
  - Remplacez tous les ??? par le code correct.
  - Compilez et exécute

Cours Flutter

Source : https://apcpedagogie.com/super-et-this-en-dart · téléchargé le 08/11/2026 · © APC Pédagogie · apcpedagogie.com