Types de données

Chaîne de caractères	String
Nombre	Int / double
Booléen	bool (True or False)
Listes	List / Array
Ensemble	Sets
Dictionnaire	Map

Opérateurs

Addition	+
Soustraction	-
Multiplication	*
Division	/
Division entre nombres entiers renvoie un quotient entier	~/
Addition avec affectation	+=
Soustraction avec affectation	-=
Multiplication avec affectation	*=
Division avec affectation	/=
Modulo avec affection calcule le reste d'une division	%=
Plus grand que	>
Plus petit que	<
Plus grand ou égal	>=
Plus petit ou égal	<=
Pas égal à	!=
Vérifier si une variable est d'un type	if (variable **is** String) { }
Vérifier si une variable n'est pas d'un type	if (variable **is!** String) { }

Opérateurs (cont)

Vérifier si une variable peut être convertie	var variable = "-42"; var nombre = variable as int?; if (nombre != null) { }
Opérateur logique ET	&&
Opérateur logique OU	||
Opérateur logique PAS	!
Exprimer une condition (if-else) de manière concise	age >= 18 ? "-Majeur" : "Mineur"
Utiliser une valeur de secours si la première est nulle	nom ?? "Anonyme"
Opérateur en cascade applique à un objet plusieurs opérations sans répéter le nom de l'objet	personne ..sePresenter() ..nom = "Bob" ..age = 25 ..sePresenter();

Fonctions

Execution du programme principal	main()
Syntaxe d'une fonction	String compter(int number) { } -> returnType functionName(parameterType parameters) { }
Paramètres positionnels facultatifs	String compter(int number, *[int? number2]*) {}
Paramètres nommés facultatifs	String compter(int number, *{int? number2}*) {}
Fonctions fléchées si le corps ne contient qu'une ligne	returnType functionName(parametersType parameters...) => expression;

Syntaxe chaînes de caractère

Une seule ligne	Guillemets simples : ' ' guillemets doubles " "
Chaînes multilignes	Guillemets triples : ' ' ' str ' ' ' / " " " str " " "
Concaténation	str1 + " " + str2 / str1.concat([str2])
Liste du code UTF-16 des caractères	str.codeUnits
Vérifier si la chaîne de caractères est vide ou non	str.isEmpty / str.isNotEmpty
Longueur de la chaîne	str.length
Vérifier si une chaîne en paramètre existe dans une autre chaîne	str.contains('toto')
Vérifier si la chaîne commence / se finit par la chaîne en paramètre	str.startsWith('toto') str.endsWith('toto')
Convertir en minuscules / en majuscules	str.toLowerCase() str.toUpperCase()
Fractionner une chaîne selon le paramètre ; renvoie une liste	str.split('\n')
Diviser une chaîne et renvoyer les éléments en chaîne	str.splitMapJoin(RegExp(r'*condition*'), onMatch: (m) => '${m.group(0)}', onNonMatch: (n) => n);
Les premières et dernières correspondances du paramètre dans une chaîne	str.indexOf('toto') / str.lastIndexOf('toto')

Syntaxe chaînes de caractère (cont)

Supprime les espaces blancs de début et de fin, de début ou de fin	str.trim() / str.trimLeft() / str.trimRight()
Ajoute à gauche et à droite la chaîne donnée si elle est inférieure à la longueur spécifiée	str.padLeft(8, 'x') / str.padRight(8, 'x')
Remplace le premier paramètre par le second	str.replaceAll('e', 'é')
Comparer la relation entre les chaînes	str1.compareTo(str2)
Renvoie une sous-chaîne selon les index en paramètre	str.substring(2) / str.substring(2,5)
Créer une chaîne de caractères brutes : les caractères d'échappement ne sont pas interprétés	String str = r'toto'

Variables et constantes

Déclarer une variable (type de données définitive)	var nomVar = valeur;
Déclarer une variable (type de données modifiable)	dynamic nomVar = valeur;
Déclarer une variable avec un type défini	String srt = "toto"; int nb = 2;
Déclarer une variable avec affectation unique (valeur non midifiable)	final nomVar = valeur;
Récupérer le type d'une variable	print(nomVar.runtimeType);

Variables et constantes (cont)

Affecter une variable dans une chaîne	print("Tu es $nom et tu as bientôt ${âge + 1} ans");
Récupérer une ligne écrite dans la console dans une variable	String? nom = stdin.readLineSync();
Définir une constante (const peut être remplacé par final et inversement)	const nomVar = valeur; final dataType nomVar = valeur

Fonctions mathématiques

Rendre la valeur absolue	nb.abd();
Faire un arrondi à l'unité près	nb.round();
Faire un arrondi à l'entier supérieur	nb.ceil();
Faire un arrondi à l'unité inférieur	nb.floor();
Récupérer uniquement l'unité d'un nombre à virgule	nb.truncate();

Structures conditionnelles

if (expBooleenne) { }

if (condition) { } **else** { }

if (condition1) { } **else if** (condition2) { } **else** {}

switch (expression) { **case** condition1 : instruction1; **break**; ... ; **default** : instruction par défaut }

Boucles

for (initialisation, condition, itération) { }

for (var varTemporaire **in** collectionAParcourrir) { }

while (condition) { }

do { instructions } **while**(condition)