ОБЩИЕ ЗАМЕЧАНИЯ

Каждая переменная в Дарте ссылается на объект – экземпляр класса. Следовательно все переменные – объекты, а все типы – классы.

Некоторым типам в Дарте отведены специальные роли:

* *Object*: суперкласс всех дартовских классов, кроме Null
* *Enum*: суперклассы для всех классов-перечислений
* *Future и Stream*: используются в асинхронности и многопоточности
* *Iterable*: используется в циклах for и синхронных функциях-итераторах
* *Never*: обозначает что вычисление никогда успешно не завершится, используется в функциях, которые всегда выбрасывают исключение
* *dynamic*: обозначает что вы не хотите использовать статическую проверку типов
* *void*: обозначает, что значение никогда не используется

ТИПЫ ДАННЫХ

**num** – класс для представления знаковых вещественных и целых чисел, имеет два подкласса:

* **int** – представляет 64-битные целые числа со знаком;
* **double –** представляет 64-битные вещественные числа со знаком.

Значения int могут автоматически конвертироваться в double при необходимости, но double в int не может – это вызовет ошибку.

**String** – тип данных, который представляет объект, хранящий последовательность символов кодировки UTF-16.

Для создания строкового литерала можно использовать и одинарные, и двойные кавычки.

Для интерполяции внутри строки перед идентификатором ставится символ *$str*, если интерполировать надо выражение, то помимо специального символа так же ставятся фигурные скобки *${str.toUpperCase}.*

Оператор ‘==’ используется для сравнения содержимого строк.

Также имеется возможность писать литералы в несколько строк, для этого в начале и конце литерала ставятся три символа одинарной либо двойной кавычки: **var s1 = '''You can create**

**multi-line strings like this one.**

**''';**

Можно создать “сырую” (строку с явным обозначением символов пробела и переноса строки), написав перед литералом букву r:

**var s = r'In a raw string, not even \n gets special treatment.';**

**bool –** тип данных, предназначенный для представления бинарных значений true/false.

**Symbol** – тип данных для создания уникальных идентификаторов, перед значением ставится #. (если понадобится, то лучше почитать дополнительно).

КОЛЛЕКЦИИ

Dart имеет встроенную поддержку 3х видов коллекций: list, set, map. Рассмотрим их подробнее:

* **List<T> –** вид коллекции, представляющий набор элементов определённого типа, заменяет собой массивы и динамические списки. Является динамическим. Чтобы создать List, недоступный для изменений во время выполнения (в плане изменения ссылки и полей), необходимо перед литералом, инициализирующим лист, написать слово const: **var constantList = const [1, 2, 3];**
* **Set<T> –** вид коллекции, представляющий множество уникальных значений. То есть, каждое значение в множестве уникально. Для создания множества используется оператор {}: **var names = <String>{};**
* **Map<T1, T2>** – вид коллекции, представляющий собой набор элементов в виде пары <ключ, значение>. Каждому значению соответствует свой уникальный ключ. Значения могут быть неуникальными и повторяться, однако ключи обязаны таковыми быть.

Dart поддерживает оператор расширения (spread operator), который используется для того, чтобы подставить все значения какой-либо последовательности:

**var list = [1, 2, 3];**

**var list2 = [0, ...list];**

**assert(list2.length == 4);**

ОБОБЩЁННЫЕ/ПАРАМЕТРИЗОВАННЫЕ ТИПЫ (ДЖЕНЕРИКИ)

По кодовому соглашению, имена дженерик-типов пишутся одной заглавной буквой E, T, S, K, V.

Дженерик типы используются для предотвращения дублирования кода. В Dart можно проверять дженерик-типы объектов во время выполнения.

Дженерики можно использовать для определения параметризованных классов:

**class Foo<T extends Object> {**

**// Any type provided to Foo for T must be non-nullable.**

**}**

TYPEDEFS

Для определения собственного типа используется ключевое слово typedef:

**typedef IntList = List<int>;**

**IntList il = [1, 2, 3];**