|  |
| --- |
| Is Dart object oriented? Why?  Да, дарт является ООП языком, потому что в Дарте используется объектно-ориентированный подход при написании кода, в Дарте все переменные являются объектами, имеют свои конструкторы, а все типы являются наследниками базового класса Object, соответственно, все типы в Дарте – классы, а класс является базовым элементом Объектно-ориентированного программирования. Также дарт использует сборщик мусора для управления памятью. |
| What are the access modifiers in Dart?  Public, protected, private.  Public – доступны из любого места (явно указывать не надо).  Protected – доступны только внутри пакета (явно указывать надо).  Private – доступны только внутри одного файла (указывается явно с помощью нижнего подчёркивания). |
| Given we have class A and class B in one file, can we access private field of class A from class B? What if they are in separate files? |
| Да, можем. Если классы будут в разных файлах, то не сможем. Это специфика работа private. |
| What is the difference between JIT and AOT?  Just-in-time – компиляция на-лету, во время исполнения программы. Блоки кода компилируются только перед своим исполнением. Однако для работы приложения необходим компилятор. JIT даёт флаттеру такое преимущество, что изменения кода применяются сразу, без необходимости перезапуска приложения и его полной перекомпиляции.  Ahead of time – компиляция полностью выполняется перед запуском программы. АОТ компиляция создаёт самодостаточное приложение, не требующее компилятора Дарт для выполнения программы, в отличие от JIT, однако при внесении каких-либо изменений в код приложение нужно будет полностью перекомпилировать и перезапустить, что намного дольше, чем при внесении изменений при JIT-компиляции.  JIT-компиляция происходит по умолчанию, чтобы скомпилировать через AoT, необходимо указать при компиляции флаг –аоt. |
| What is the difference between final and const?  Константы при объявлении должны сразу инициализироваться, а файналы могут позже. |
| Does Flutter has interfaces?  Интерфейсы объявляются ключевым словом interface, не имеют реализации (абстрактные классы могут иметь частичную реализацию), класс может реализовывать множество интерфейсов, но наследовать только один абстрактный класс.  Mixins? |
| What built-in Dart types and collections do you know? |
| Каждая переменная в дарте – это объект. Object – суперкласс для всех классов, кроме null.  **Void** используется для нормальных функций без возвращаемого значения, а **never** - для аномальных функций, которые никогда не завершатся или всегда завершатся исключением. **Dynamic** используется, если мы хотим обрабатывать значения любого типа.  **Numbers** (тип num) – тип данных, представляющий собой целые и вещественные числа. Имеет подтипы в виде int (64-битное целое) и double (64-битное вещетсвенное). Int автоматически конвертируется в double при необходимости.  **String** – тип данных, преставляющий объект, который содержит строку символов utf-16 (2 или 5 байта).  **Bool** – тип данных, имеющих 2 значения (истина или ложь), переменная занимает 1 байт.  **Runes** – тип данных для работы со строкой как с символами Unicode, предоставляет возможность работать с каждым отдельным символом, но занимает больше места. Каждый элемент runes – это число типа int, которое представляет символ юникода.  **Symbol** – тип данных для создания уникальных идентификаторов, перед значением ставится #.  Name the OOP principles |
| Наследование, полиморфизм, инкапсуляция, абстрагирование  What are generics? |
| Дженерики (обобщённые типы) – это типы данных, инициализация которых происходит во время компиляции (важно!). То есть мы вместо типа, который определён как дженерик можем подставить любой другой тип. Нельзя в константах  Does Dart have overriding and overloading? |
| Да, дарт имеет возможность перегрузки и переопределения методов. Рассказать что это.  How do you handle exceptions in Dart? |

С помощью конструкции try-on-catch-finally:

try {

// Код, который может генерировать исключение

} on ExceptionType1 catch(e) {

// Обработка исключения ExceptionType1

} on ExceptionType2 catch(e) {

// Обработка исключения ExceptionType2

} on Exception catch(e) {

// Обработка всех других исключений

} catch(e) {

// Обработка всех исключений

} finally {

// Код, который выполнится в любом случае

}