|  |
| --- |
| ***What is Future? How to use?***  Future is dart class that represents result of delayed operation. Future instance can be in two states: uncompleted and completed.  We can return value from Future using Future API (then method) or await keyword.  What happens when we create Future:   1. Created instance 2. Code which can be executed in this future adds to Event queue in Event loop 3. Returns Future instance with state uncompleted |
| ***Can you explain the concept of streams in Flutter?***  Stream is sequence of some values, on which we can subscribe and get notifications when new elements added into sequence. Stream is asynchronous. |
| ***Is a List of Futures the same as a Stream?***  Not the same because we cant subscribe on list of futures. |
| ***What types of Streams are there, what are the differences?*** |
| Single-subscription stream.  Broadcast stream. This stream can have many listeners and we can listen this stream at any time.  ***Explain the concept of a Future/StreamBuilder.***  FutureBuilder is class designated to control Future, get information about Future state and choose what to show when Future executed. In most cases it used in widgets.  StreamBuilder is class designated to control Streams, handle it events and choose what to show when Stream events are not available. It also used in widgets. |
| ***What is the difference between asynchrony and multithreading?*** |
| Разница между асинхронным и многопоточным выполнением следующая: если мы запускаем код, помеченный как асинхронный, то он выполняется в том же потоке, в котором мы его запустили, если мы запускаем код, предназначенный для многопоточного выполнения, то для начала мы должны создать новый поток выполнения (или получить его из пула потоков) и только потом запустить там этот код.  ***Dart multi-threaded or single-threaded?***  Dart is single-threaded language. Dart executes only one instr**u**ction at the same time. He executes instructions coh**e**rently (последовательно). It means that one instruction execution can’t be interrupt (прервано) by another instruction. But for true I feel some contradiction in this words, because we have isolates, which presentated as alternative of threads. |
| ***What is Isolate?***  Isolate is analogue of thread concept in Dart, but with differences. Every isolate has its own Event loop and memory space.  import 'dart:isolate';  void main() async{  print("Start calculation");  // создаем порт приема сообщений от нового изолята  final receivePort = ReceivePort();  // создаем новый изолят  final isolate = await Isolate.spawn(count, receivePort.sendPort);  // запускаем прослушивание входящих сообщений  receivePort.listen((message) {  print(message);  // изолят больше не нужен - завершаем его  receivePort.close();  isolate.kill();  });  print("Some work...");  }  void count(SendPort sendPort) {  var result = 0;  for (var i = 1; i <= 10000000000; i++) {  result = i;  }  sendPort.send(result);  } |
| ***How is Isolate different from a stream?***  У каждого изолята есть свой Event loop и каждый изолят имеет свою область памяти. |
| ***How is communication between Isolates?*** |
| ***What is an Event Loop?*** |
| Event loop (цикл событий) – это процесс, который управляет очерёдностью исполнения инструкций.  Когда этот процесс создан, Дарт автоматически выполняет следующие действия:  \* инициализирует две очереди (Queues) с именами MicroTask (микрозадания) и Event (событие), тип очередей FIFO  \* исполняет метод main() и, по завершении этого метода  \* запускает Event Loop (цикл событий)  Состоит из двух очередей инструкций: Microtask queue и Event queue.  Когда EventLoop встречает асинхронную задачу, то он помещает её в одну из очередей.  Очередь Event  Используется для планирования операций, которые получают результат от:  ● внешних событий, таких как  — операции ввода/вывода  — жесты  — рисование  — таймеры  — потоки  ...  — Future  ***What is a MicroTask Queue?*** |
| MicroTask Queue – первая из очередей Event loop. Используется для очень коротких действий, которые должны быть выполнены асинхронно, сразу после завершения какой-либо инструкции перед тем, как передать управление обратно Event Loop.  В качестве примера очень короткого действия представьте освобождение ресурса сразу после его закрытия.  ***What are yield, yeild\*, sync\*, async\* statements?***  yield – для возвращения значения из генератора.  yield\* – для возвращения другого Stream/Iterator (рекурсивного вызова).  sync\* – синхронный генератор.  async\* – асинхронный генератор. |
| ***Why is it impossible to do long computational operations in an asynchronous method? What should be used in this case?*** |

Why is it imposiible? I think its possible, but in this case there are no sense in asynchronous operation. Because its execution will slow down program same as we will do it without asynchronous operation. Its going on like this because when we create async operation its code postponed in Event queue in Event loop and when it starts to execute then program freeze on long computation.

In this case we should use isolates. We can run long computation in isolate and it also eat more resources, but cant freeze our program.