ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо видно && слышно?

Ставьте + , если все хорошо
Напишите в чат, если есть проблемы

Flutter Mobile Developer

Тема 6. Dart. Streams

Цель урока

- научиться работать со стримами
- рассмотреть тонкости языка - sync*, async* и прочие;
- после занятия вы сможете:
- писать код, используя потоки данных.

О чем будем говорить

Dart Streams

Generators

Problem

```dart
///
/// Мы разработчики своей библиотеки для чтения каких-либо данных.
/// Представим себе, что для чтения данных мы разработали такой интерфейс.
/// Обратите внимание - мы задействовали здесь даже Generics и наш DataReader
/// может читать любые данные!
/// Но где здесь проблема????
///
abstract class DataReader<T> {
  Iterable<T> read();
}

///
/// Кажется, что такой контракт DataReader можно использовать для реализации
/// конкретных классов для чтения чело-либо. Например, создадим класс для
/// чтения из файла побайтово!
///
class IOFileDataReader implements DataReader<int> {
  @override
  Iterable<int> read() {
```

Problem

```dart
///
/// Мы разработчики своей библиотеки для чтения каких-либо данных.
/// Представим себе, что для чтения данных мы разработали такой интерфейс.
/// Обратите внимание - мы задействовали здест даже Generics и наш DataReader
/// может читать любые данные!
/// Но где здесь проблема????
///
abstract class DataReader<T> {
  Future<Iterable<T>> read();
}

///
/// Кажется, что такой контракт DataReader можно использовать для реализации
/// конкретных классов для чтения чело-либо. Например, создадим класс для
/// чтения из файла побайтово!
///
class IOFileDataReader implements DataReader<int> {
  @override
  Future<Iterable<int>> read() async {
```

Dart Streams

https://api.dart.dev/stable/2.10.2/dart-async/Stream-class.html

Stream<T> class Null safety

A source of asynchronous data events.

A Stream provides a way to receive a sequence of events.

Each event is either a data event, also called an element of the stream, or an error event, which is a notification that something has failed.
When a stream has emitted all its event, a single "done" event will notify the listener that the end has been reached.

Dart Streams

На прошлом уроке мы познакомились с Futures.
Future представляет собой значение, которое будет вычислено асинхронно.
Future можно считать результатом будущего вычисления, или обещанием предоставить нам такое значение.

Stream - это похожая концепция, только вместо одного значения Stream предоставляет нам ноль или больше значений (или ошибки),
которые мы получим в будущем и асинхронно.

Dart Streams

Другая концепция:

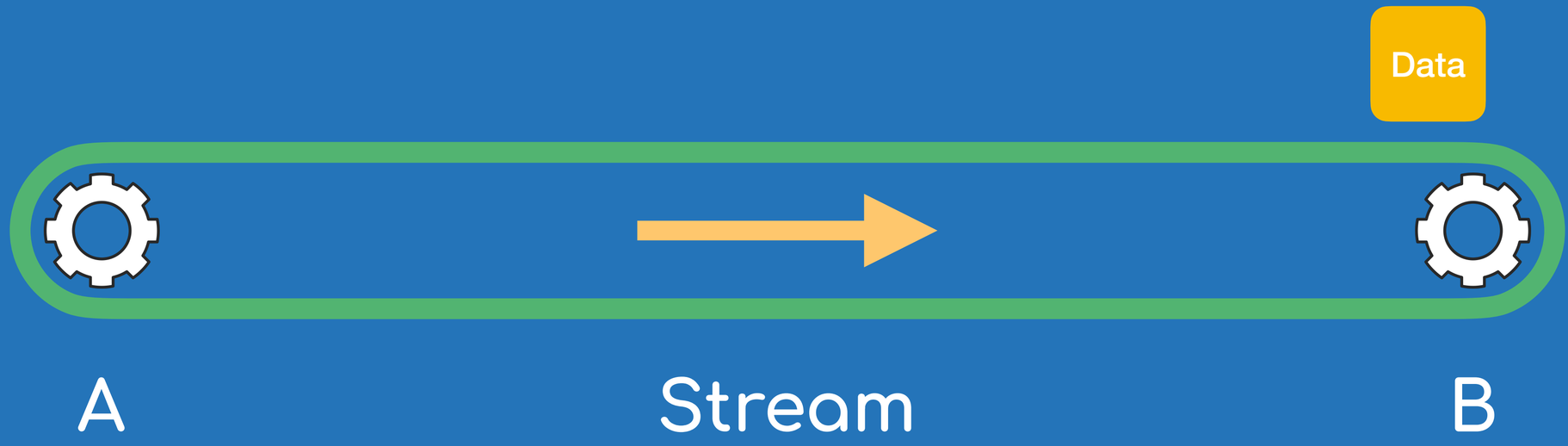
Stream можно представить в виде конвейера, который отправляет нам данные из точки A в точку B.

В процессе передачи таких данных над ними можно выполнить трансформацию перед их получением.

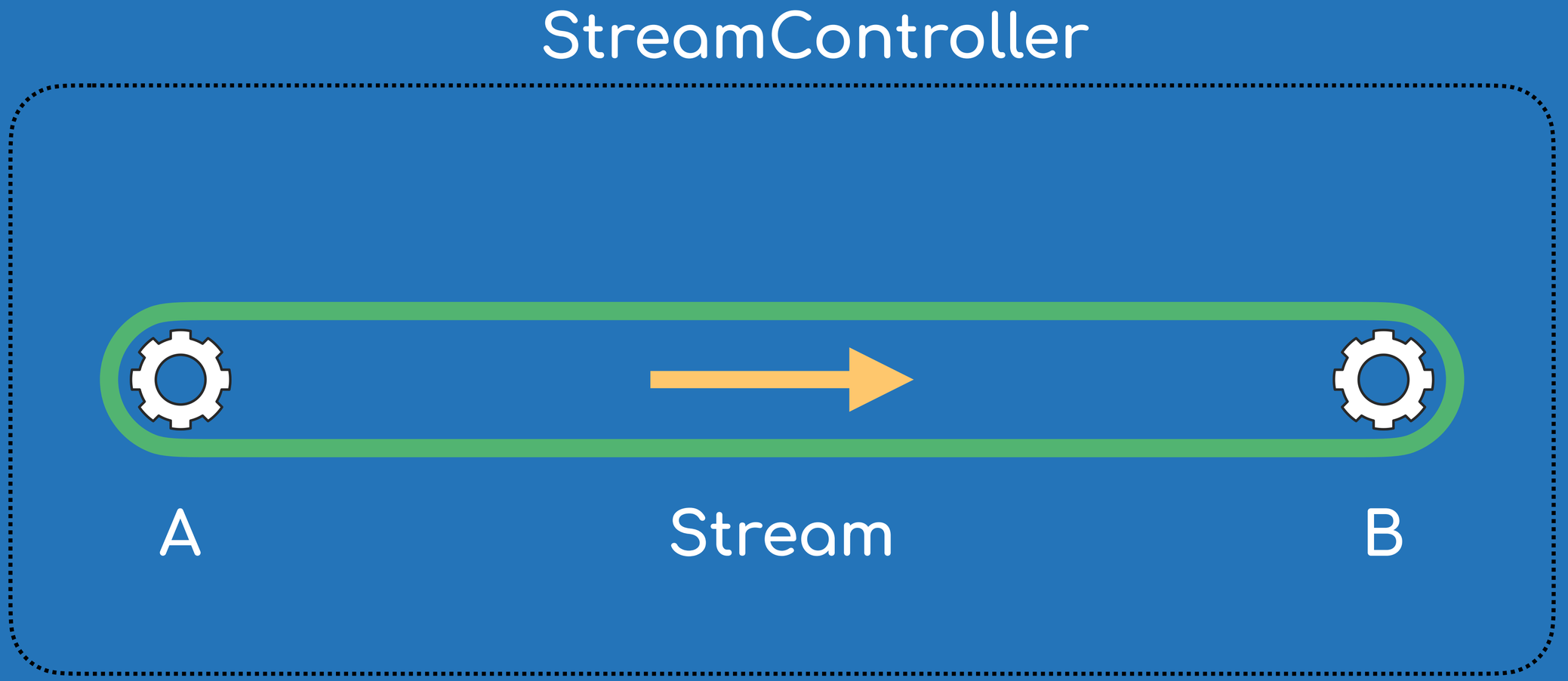
Положить данные в точку A можно при помощи “приемника” данных - Sink.

В точке B данные мы можем “получить”, подписавшись на получение данных из конвейера.

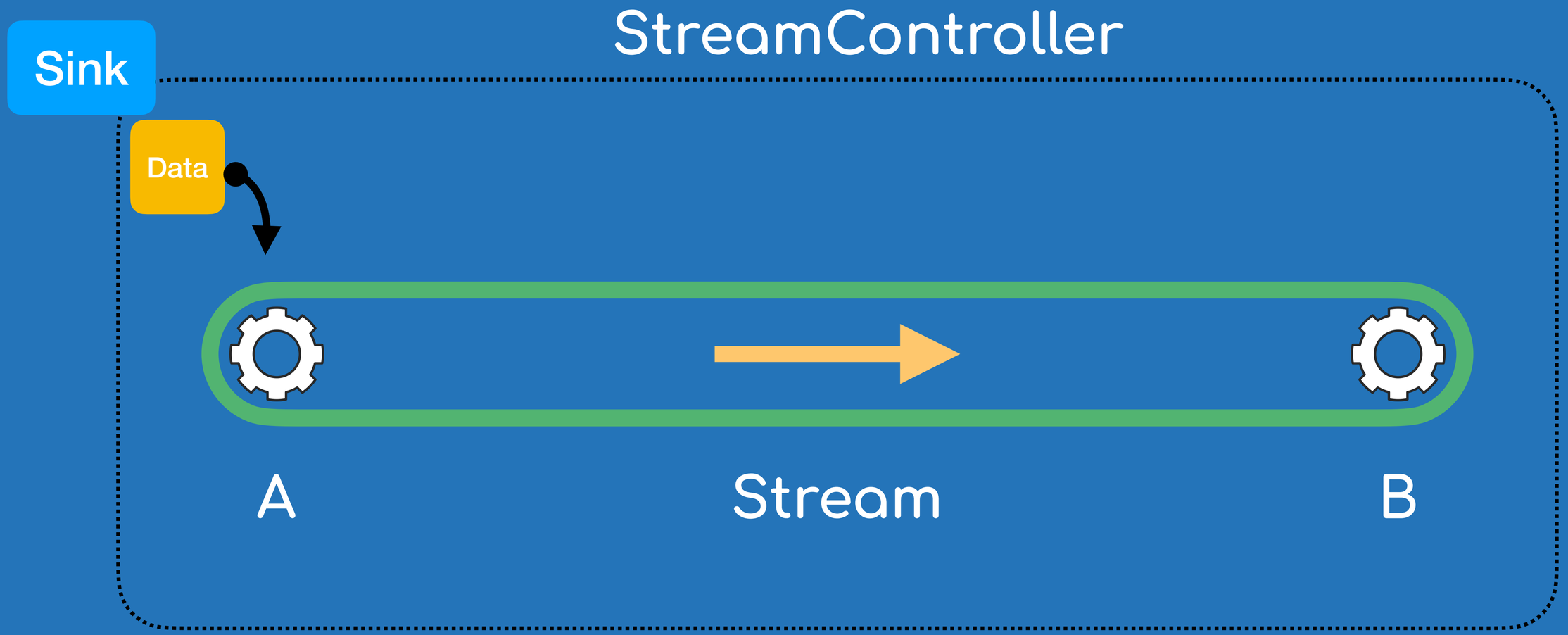
Dart Streams



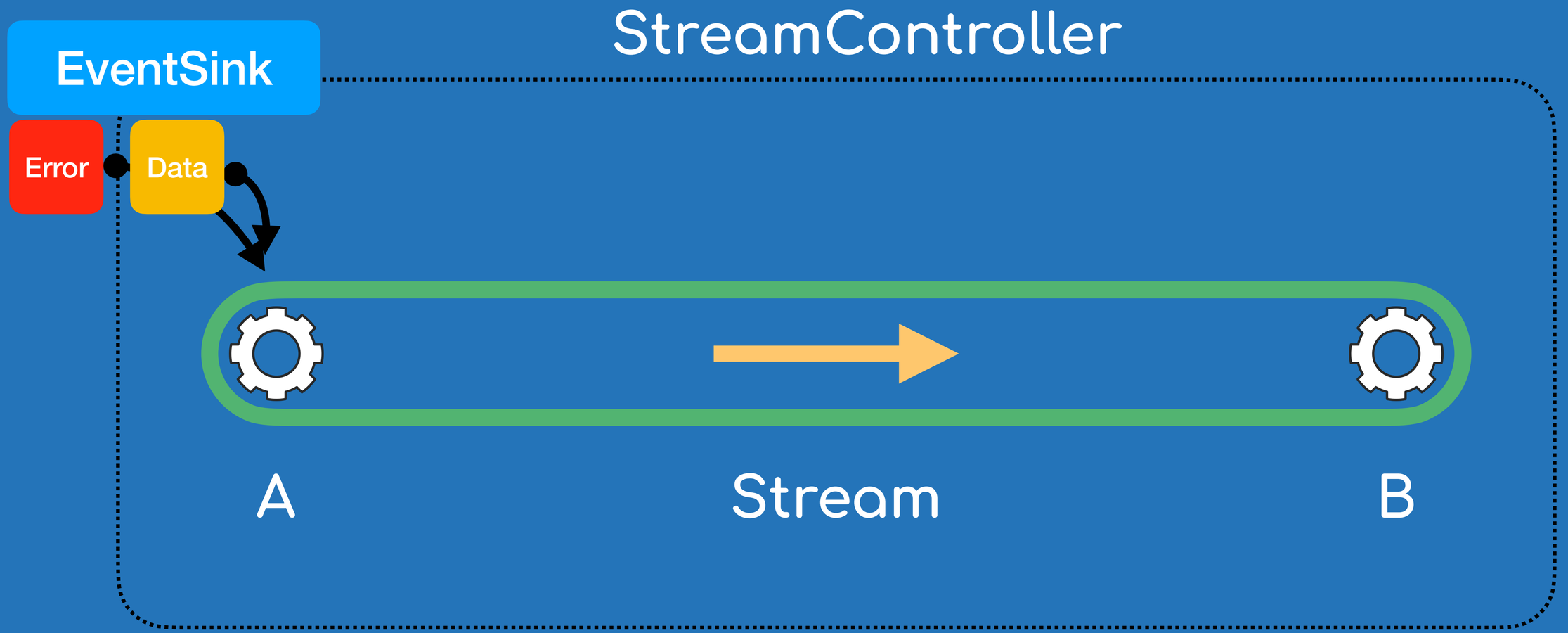
Dart Streams



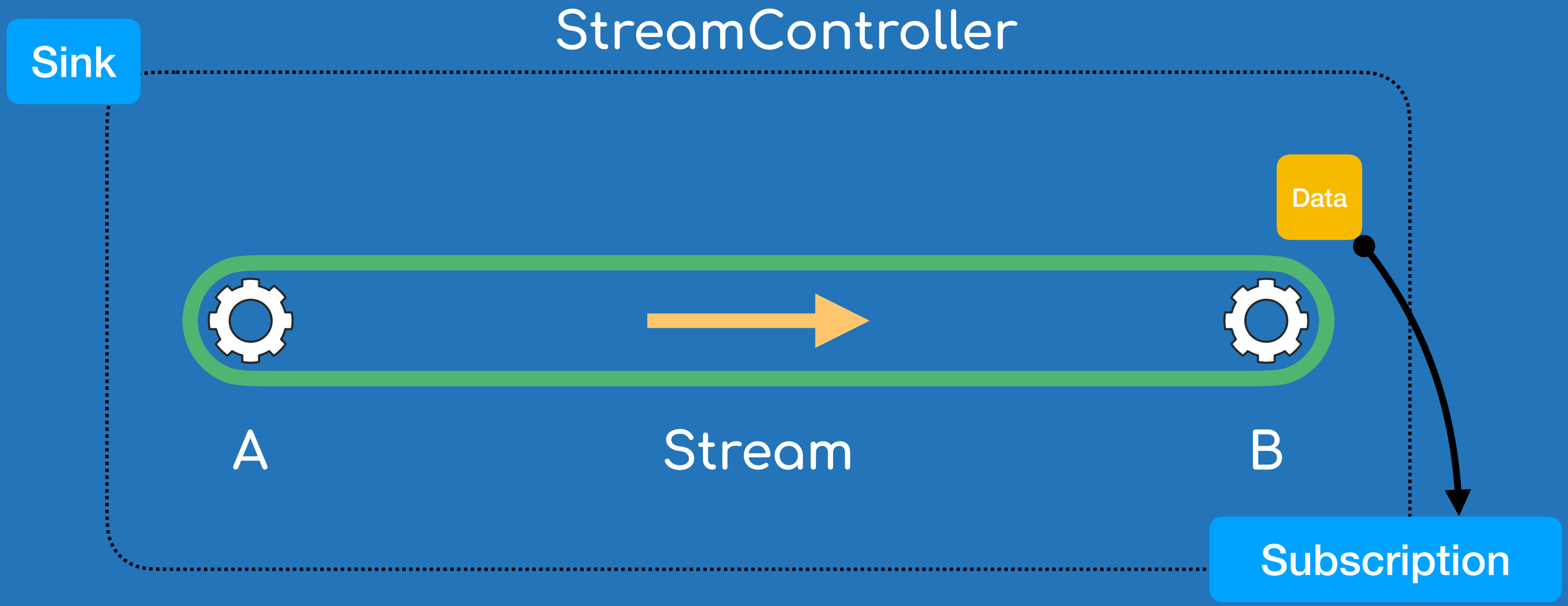
Dart Streams



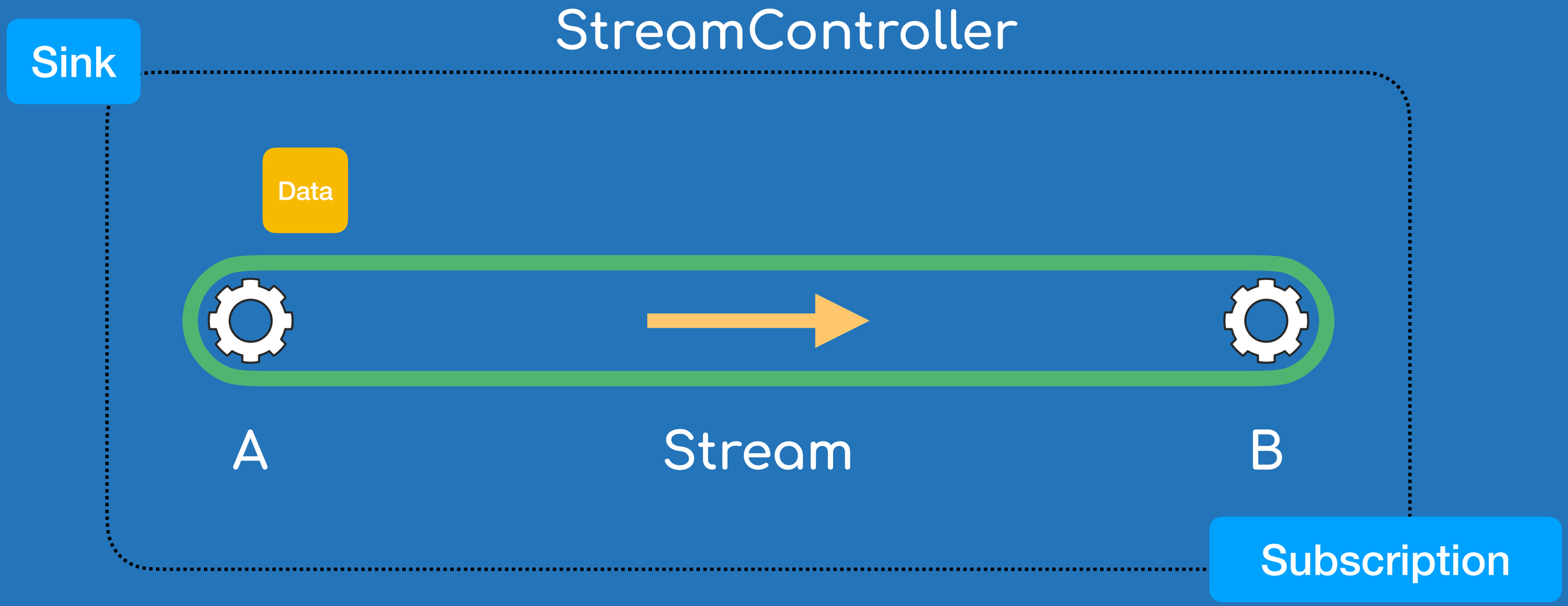
Dart Streams



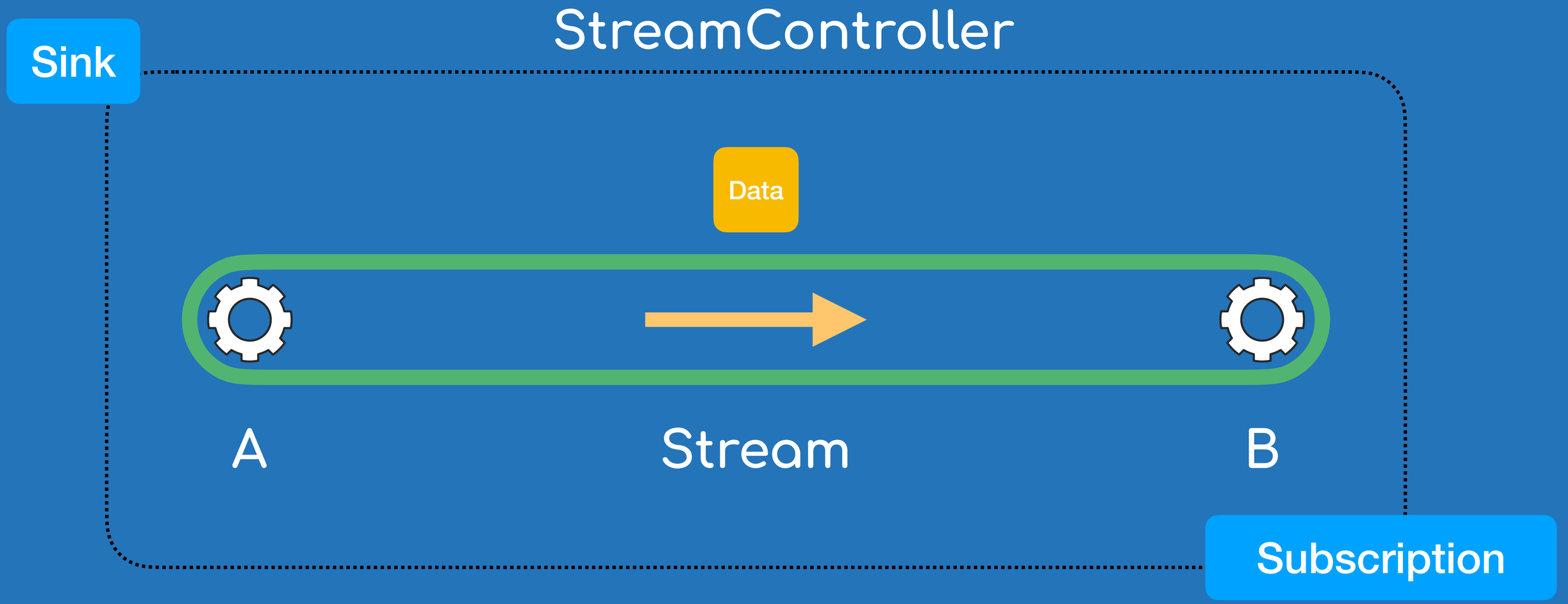
Dart Streams



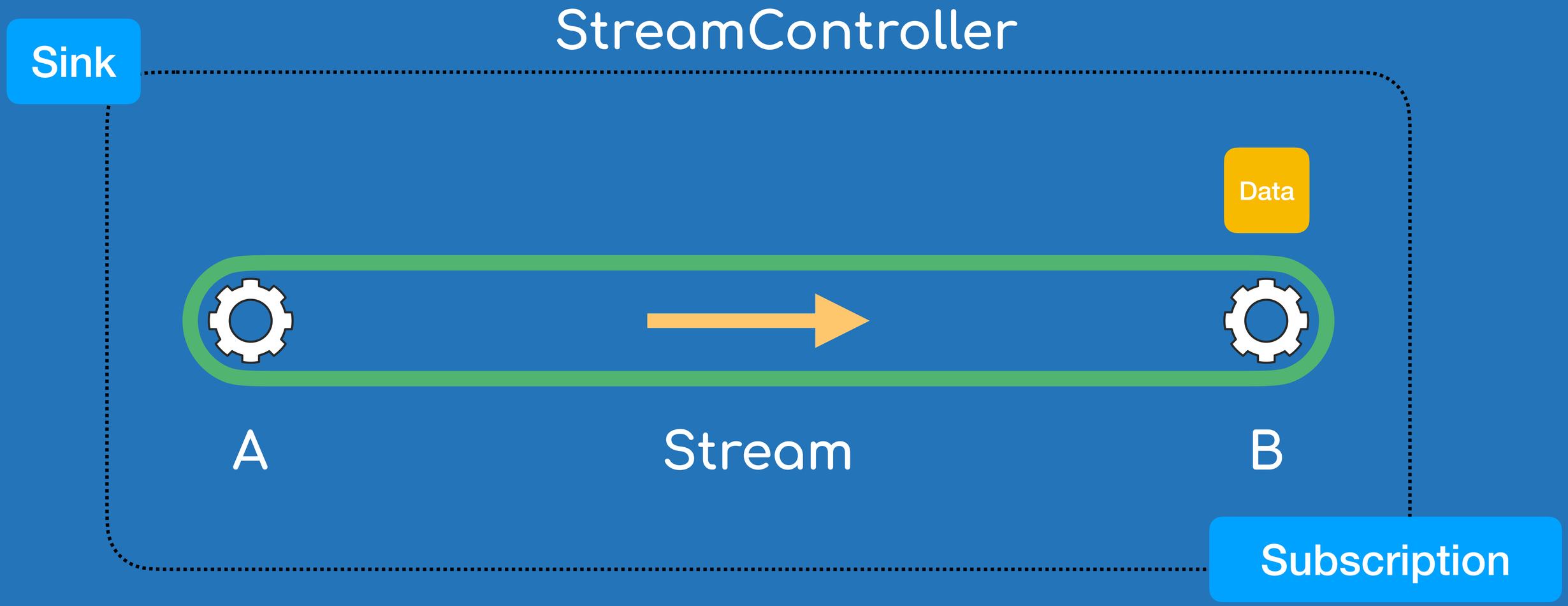
Dart Streams



Dart Streams



Dart Streams



Dart Streams

03_stream_controller_creation.dart

```dart
///
/// Создаем поток, обощив его типом int для его элементов
///
final streamController = StreamController<int>();

///
/// Положим в Stream при помощи Event Sink два элемента
///
streamController.sink
  ..add(elementData)
  ..add(nextElementData);

///
/// for each для коллекций выполняется синхронно
///
//  Iterable.generate(10).forEach((element) => print('element data is $element'));

///
/// Как вы считаете, как будет выполнен for each для потока?
///
//  streamController.stream.forEach((element) {
//    print('element data is $element');
//  });

```

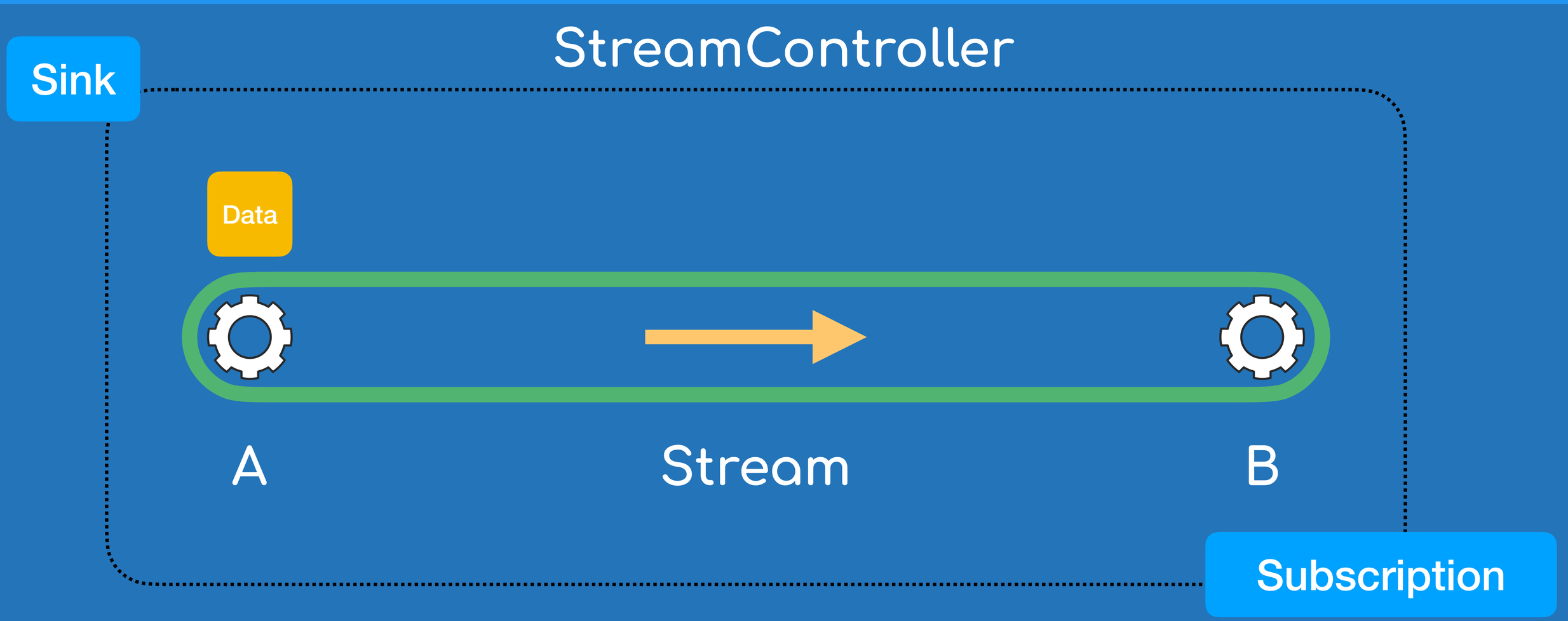
Dart Streams Transformations

Dart Streams Transformations

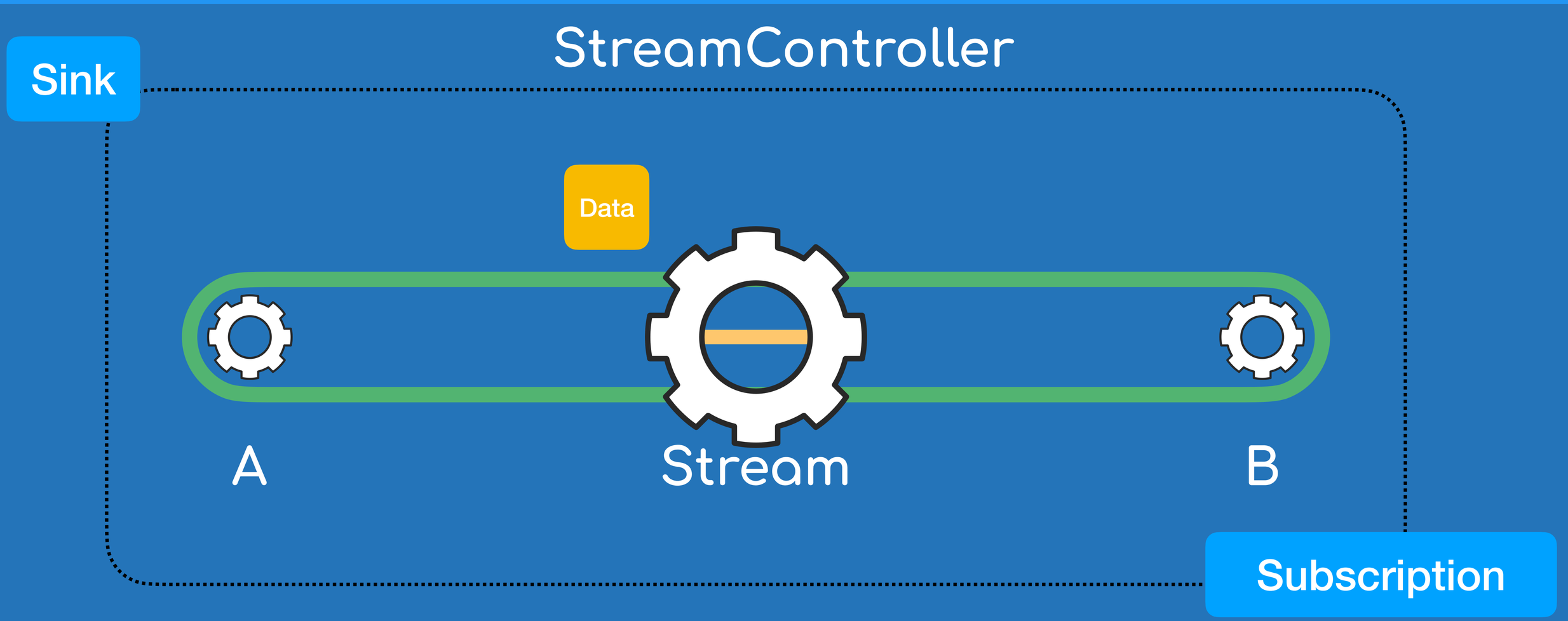
Один из наиболее часто используемых сценариев
такой, что вы можете создать новый Stream, используя другой в качестве исходного.

Например, читаем файл в виде Stream<int> (как поток байт) и сразу отдаем в виде потока символов UTF-8

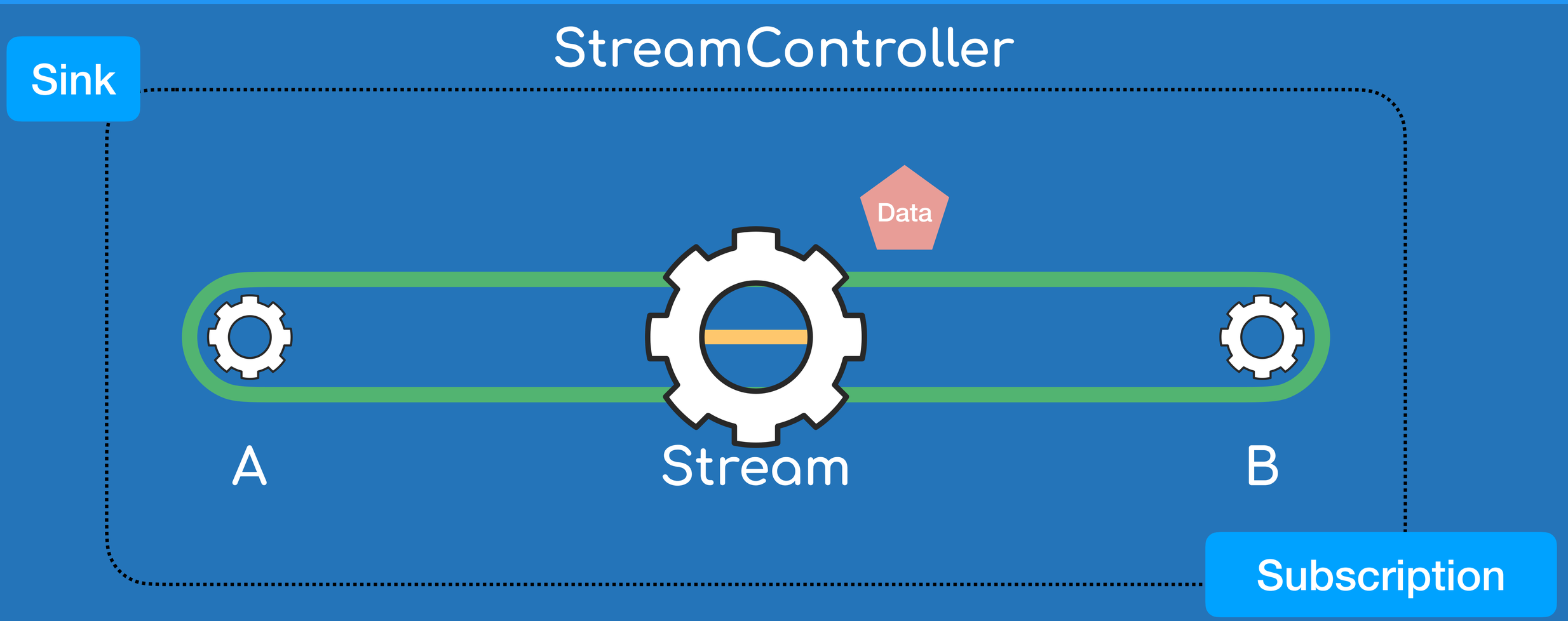
Dart Streams Transformations



Dart Streams Transformations



Dart Streams Transformations

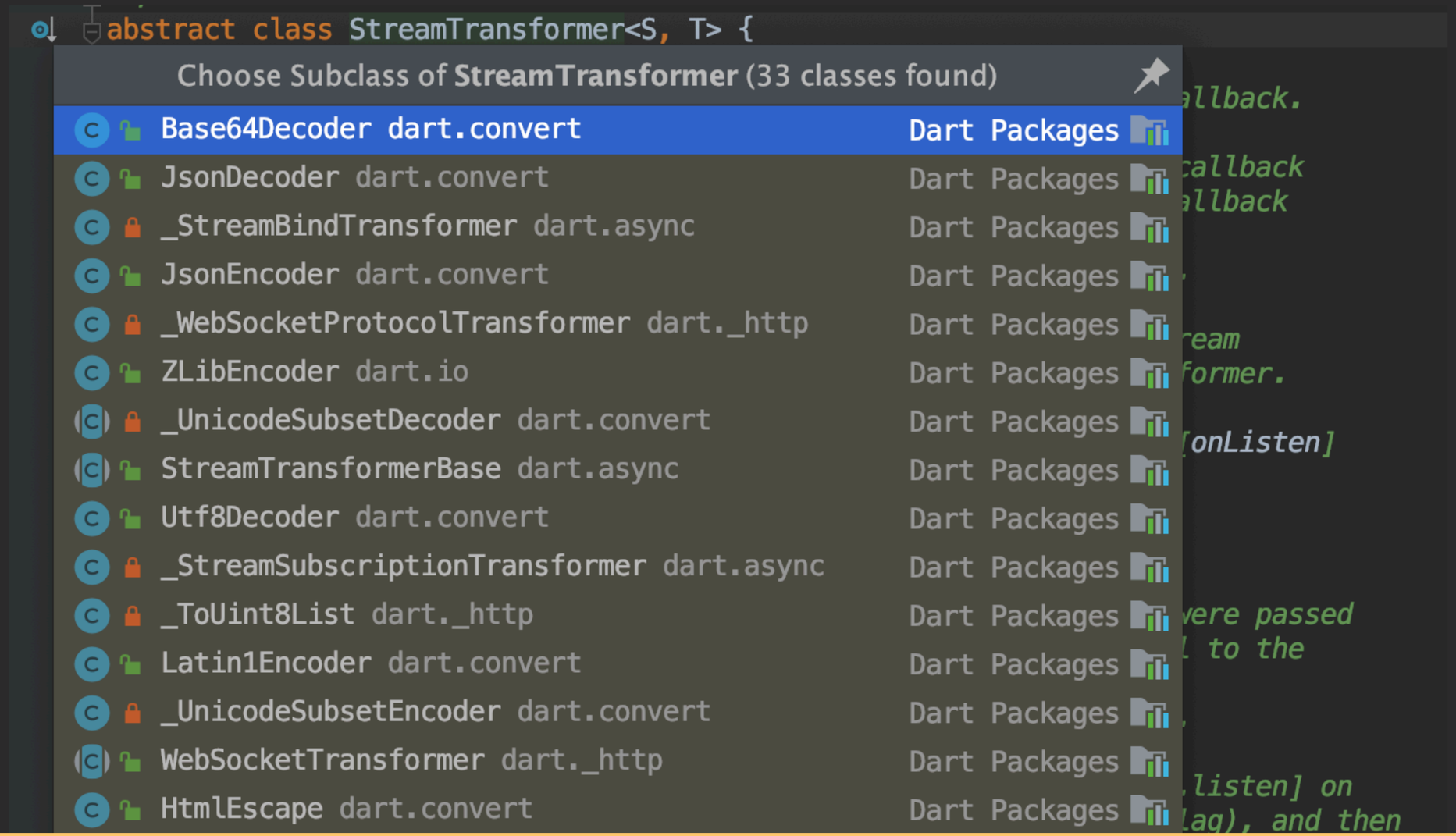


Streams Transformations

```
Stream<S> map<S>(S Function(T event) convert);
Stream<T> skip(int count);
Stream<T> skipWhile(bool Function(T element) test);
Stream<T> take(int count);
Stream<T> takeWhile(bool Function(T element) test);
Stream<T> where(bool Function(T event) test);
Stream<R> cast<R>();
Stream<S> expand<S>(Iterable<S> Function(T element) convert);
```

Streams Transformations

Stream.transform()



О чем будем говорить

Dart Streams

Generators

Generators

https://dart.dev/guides/language/language-tour#generators

Generators

When you need to lazily produce a sequence of values, consider using a generator function. Dart has built-in support for two kinds of generator functions:

- Synchronous generator: Returns an Iterable object.
- Asynchronous generator: Returns a Stream object.

Generators

https://dart.dev/guides/language/language-tour#generators

Важно

Генераторы — это особый класс функций, которые упрощают задачу написания итераторов.

А также служат для ленивой генерации значений, и по требованию.

То есть - пока мы получаем данные, происходит их генерация.
Прекращаем получение - генерация данных заканчивается.

Generators - Iterable (sync*)

```dart
///
///
///
Iterable<int> getTopNCollection(int start, int count) sync* {
  for (var i = start; i <= (start + count); i++) {
    print('yield $i');
    yield i;
  }
}
```

Generators - Stream (async*)

```dart
///
///
///
///
Stream<int> getTopNCollection(int start, int count) async* {
  for (var i = start; i <= (start + count); i++) {
    print('yield $i');
    yield i;
  }
}
```

Итоги

Dart Streams

Generators

Single Value & Future & Iterable & Streams

Sync	Int	Iterator<int>
Async	Future<int>	Stream<int>

Event Loop - все гораздо сложнее

Для самостоятельного изучения - вопрос по прошлой теме

https://webdev.dartlang.org/articles/performance/event-loop

Event Loop - все гораздо сложнее

```dart
///
/// Для демонстрауии асинхронного чтения из потока мы в поток сначала положим один элемент
///
streamController.sink.add(elementData);

///
/// Затем разместим в event loop некоторые пользовательские действия
///
Timer.run(() {
  print('user onTap gesture in event loop');
});

///
/// Затем поместим в потом еще один элемент потока
///
// Timer.run(() {
scheduleMicrotask(() {
  streamController.sink.add(nextElementData);
});

Future(() {
  print('another async operation to process http response');
});
```

Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Спасибо за внимание!
Приходите на следующие вебинары

Смирнов Андрей wrike

Курс **Мобильная разработка на Flutter**