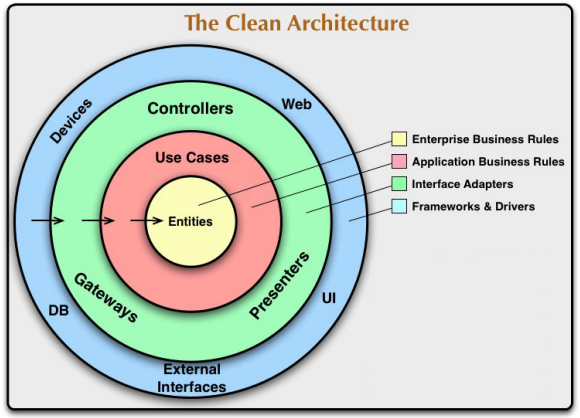
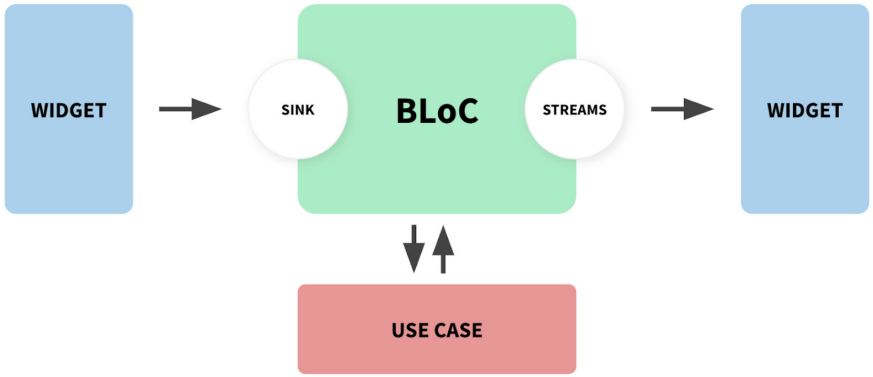
# 1 Clean architecture

Dependencies can only point inwards.
e.g. the inner circle cannot know anything about an outer circle




# 2 Communication in BLoC pattern

- Widgets sends events to the BLoC via sinks.
- BLoC executes business logic, invoking the use-case;
- Widgets are notified by the BLoC via streams.




baseflow

# 3 Useful classes

`Stream` a class representing an asynchronous source of data events.
`StreamController` a class that is used to manage a stream. It can be used to create a simple stream others can listen to, and to push events on that stream.
`StreamTransformer` a class that can be used to transform data that is sent on a stream.
`StreamBuilder` a Flutter widget that builds itself based on the latest snapshot of interaction with a Stream.

**A typical BLoC class should have the following properties:**

- An input sink: the sink property can be used by widgets to notify the BLoC of new events;
- An output stream: the output stream property can be used by widgets to listen for state changes and create the layout accordingly.

# 4 Guidelines

## BLoC

1. Inputs and outputs should be simple (Sinks and Streams only);
2. Dependencies must be injected and must be platform agnostic;
3. Platform branching is forbidden.

## UI

1. Each UI component has a corresponding BLoC;
2. UI components send input “as is”;
3. UI components should show output as close as possible to “as is”;
4. All branching should be based on one simple BLoC boolean output.

## References


**Flutter / AngularDart - Code sharing deep dive by Paolo Soares (video introducing the BLoC pattern at DartConf 2018)**
https://youtu.be/PLHln7wHgPE

**Reactive Mobile apps with Flutter (Google I/O 2018)**
https://youtu.be/RS36gBEp8OI

**Reactive Programming - Streams - BLoC by Didier Boelens:**
https://www.didierboelens.com/2018/08/reactive-programming---streams---bloc/

**The Clean Architecture by Robert C. Martin (Uncle Bob):**
https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html