Await

Using the “await” keyword in Dart you can wait on asynchronous operations and grab their result before continuing with your work. Await is always used together with functions marked as “async”. You cannot await on a function that synchronously returns its value. Using “await” substantially increases productivity in working with futures and streams in Dart

Further reading:

- Asynchrony support - dart.dev
- Asynchronous programming codelab - dart.dev
- Future in dart: When to use async, await, and then - dev.to
- Using await async in Dart - GeeksforGeeks
- Async/Await - Flutter in Focus - YouTube

Examples

```dart
// this function produces the integer value of 10
// after waiting 3 seconds. This is an asynchronous
// function, as denoted by its return type of Future
Future<int> produceValueAfter3Seconds() {
  return Future.delayed(Duration(seconds: 3), () => 10);
}
// this function is marked as "async" meaning that
// inside this function you can use the "await"
// keyword to wait on the result of other async functions
void waitOnValue() async {
  // here we "await" the result of the function above
  // using the await keyword
  final value10 = await produceValueAfter3Seconds();
  print(value10);
}
```