Creating a Connected App with Flutter

ASYNCHRONOUS PROGRAMMING WITH DART

Douglas Starnes
AUTHOR / SPEAKER

@poweredbyaltnet douglasastarnes.com

Asynchronous Operations

Asynchronous operations do not require the application to wait on them

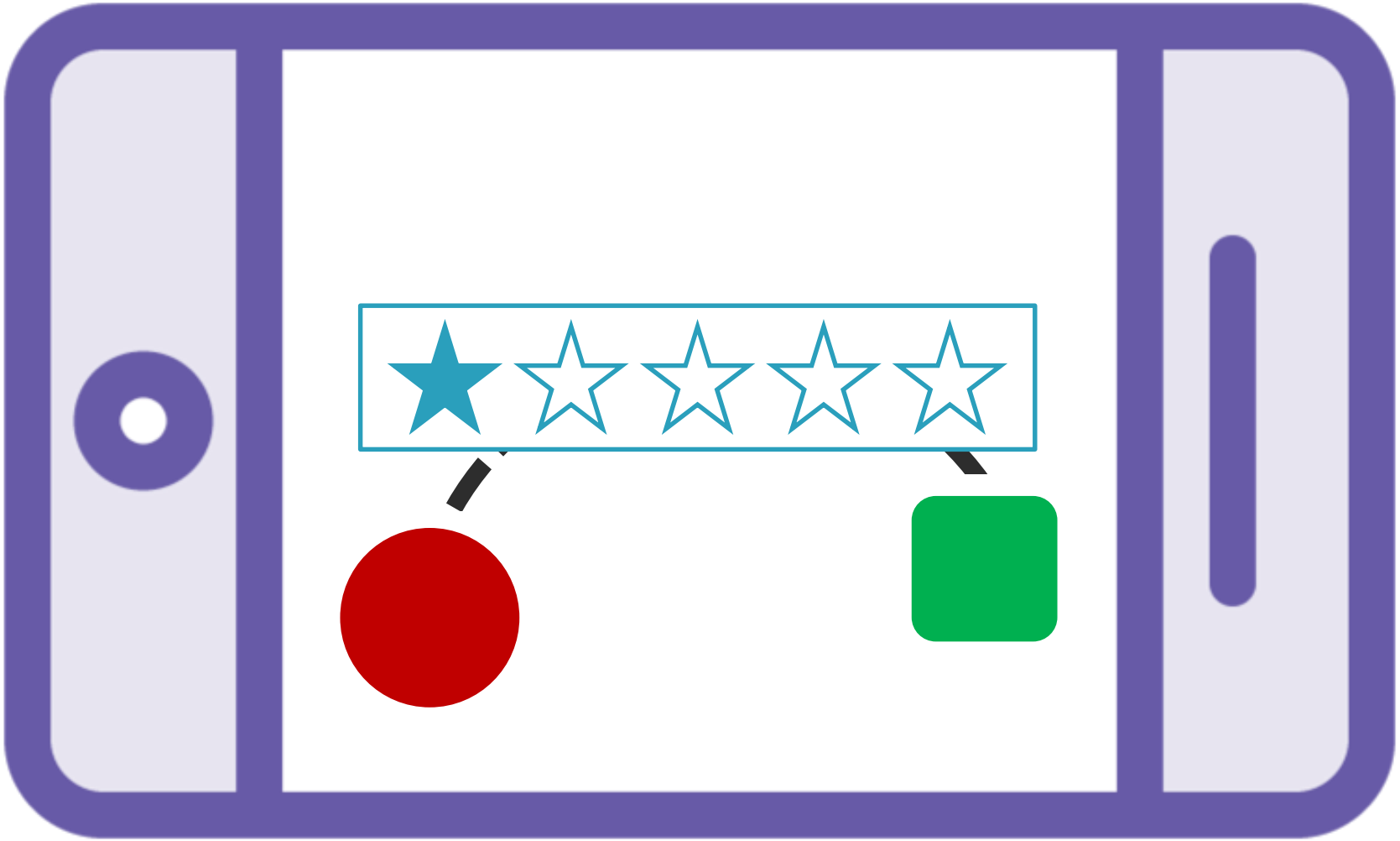
Synchronous operations do require the application to wait on them

- Also called ‘blocking’ the application
- No other operations can execute during a synchronous operation

Asynchronous Operations in Flutter Applications

Asynchronous Operations in Flutter Applications

Asynchronous Operations in Mobile Games



Asynchronous Operations

More than Flutter and Dart

JavaScript, C#, and Python

Network requests, file I/O and database queries

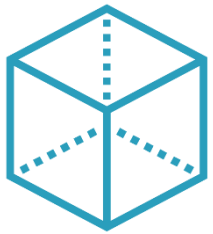
Dart asynchronous operations depend on Futures

Many languages have built in support

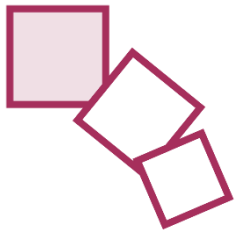
Flutter also supports Futures

Futures

The result of an asynchronous operation in Dart and Flutter is represented by a `Future`



`Future` **is a generic class**



The type of the `Future` is the same as the return value of the asynchronous operation

`Future<void>` **if the `Future` has no return value**

The State of the Future

Uncompleted

Completed

Futures that complete with success contain a value

Futures that complete unsuccessfully throw an error

`async` and `await` keywords

Simplify asynchronous code

Replace nested `then()` calls , making code more readable

Doesn’t block the user interface in Flutter apps

Futures and async/await

Both code snippets do the same thing

Futures.dart

```
pretendHTTPCall().then((s) {
    pretendDatabaseQuery(s);
});
```

Async.dart

```
Future<String> pretendHTTPCall() async {
    return Future.delayed(…);
}
Future<void> main() async {
  var query = await pretendHTTPCall();
  pretendDatabaseQuery(query);
}
```

`await` != block

Handling Errors with Asynchronous Code

Chain the catchError() method to the Future

Use try-catch

Summary

Synchronous vs. asynchronous code

- Avoid blocking the application

Futures

- Holds the result of an asynchronous task
- Return value of asynchronous functions

async and await

- Simplify asynchronous code
- Remove `then()`
- Not the same as blocking

Error handling with Futures

- `catchError()` method
- `try/catch` block