Coroutines & Qt

Mårten Nordheim

- Senior Software Engineer
- 6 years @ Qt
- Maintainer of Qt Network
- Co-maintainer of Qt WebSockets

Quick intro

General

- "Cooperative multitasking"
- "Functions that can be paused"

Quick intro

C++

- A compatible return type
- Must use co_await/co_return/co_yield

An awaitable QFuture

I QPromise, maybe

- Generic type for returning a value later
- Enable it for co_await
- QtFuture::connect(<object>, <signal>)
- Use it as the return-type for coroutines?

- Nothing in this presentation is merged or in a usable state yet
  - But there are some 3rd-party projects already available

Why care about coroutines?

(Implementation subject to change)

- Readability
  - Looks synchronous, acts asynchronous

```
1 QFuture<QByteArray> co_download(QUrl url)
2 {
3     QNetworkReply *reply = netMan->get(QNetworkRequest(url));
4
5     co_await QtFuture::connect(reply, &QNetworkReply::finished);
6     reply->deleteLater();
7
8     if (reply->error() == QNetworkReply::NoError)
9         co_return reply->readAll();
10    co_return "";
11 }
12
13 QFuture<void> co_foo()
14 {
15    using namespace Qt::StringLiterals;
16    QUrl url = u"https://qt.io"_s;
17    std::optional<QByteArray> result = co_await co_download(url);
18    QByteArray bytes = result.value();
19    if (!bytes.isEmpty())
20        process(bytes);
21 }
```

For reference

- Fairly contrived, but keeps semantics
  - ‘foo’ starts download and calls process()
  - ‘download’ downloads, turns errors into empty byte arrays

```
1 using CallbackFn = std::function<void(const QByteArray &)>;
2 void download(const QUrl &url, CallbackFn callback)
3 {
4     auto reply = netMan->get(QNetworkRequest(url));
5     auto invokeCallback = [reply, callback = std::move(callback)]() {
6         reply->deleteLater();
7
8         QByteArray data;
9         if (reply->error() == QNetworkReply::NoError)
10            data = reply->readAll();
11        else
12            data = "";
13        callback(data);
14    };
15    QObject::connect(reply, &QNetworkReply::finished,
16                     reply, invokeCallback);
17 }
18
19 void foo()
20 {
21    using namespace Qt::StringLiterals;
22    QUrl url = u"https://qt.io"_s;
23    auto callProcess = [](const QByteArray &data) {
24        if (!data.isEmpty())
25            process(data);
26    };
27    download(url, callProcess);
28 }
```

Generator

(std::generator is in C++23)

- QStringTokenizer
- QObject::findChildren

- Processing data by chunks

```
1 QFuture<void> co_gen_foo()
2 {
3     using namespace Qt::StringLiterals;
4     QUrl url = u"https://qt.io"_s;
5     QNetworkReply *reply = netMan->get(QNetworkRequest(url));
6     for (QByteArrayView chunk : reply->byChunk(4096))
7         processChunk(chunk);
8     reply->deleteLater();
9     co_return;
10 }
```

- (full disclosure: this snippet isn't tested, final version may differ or not exist :))