Flutter

... IN FIFTEEN (ISH) MINUTES

With Sierra,
Senior Android Developer,
not Flutter Developer
(but learning)

Why Flutter?

WHAT IS Flutter VS

Flutter is written in

Dart

Object Oriented

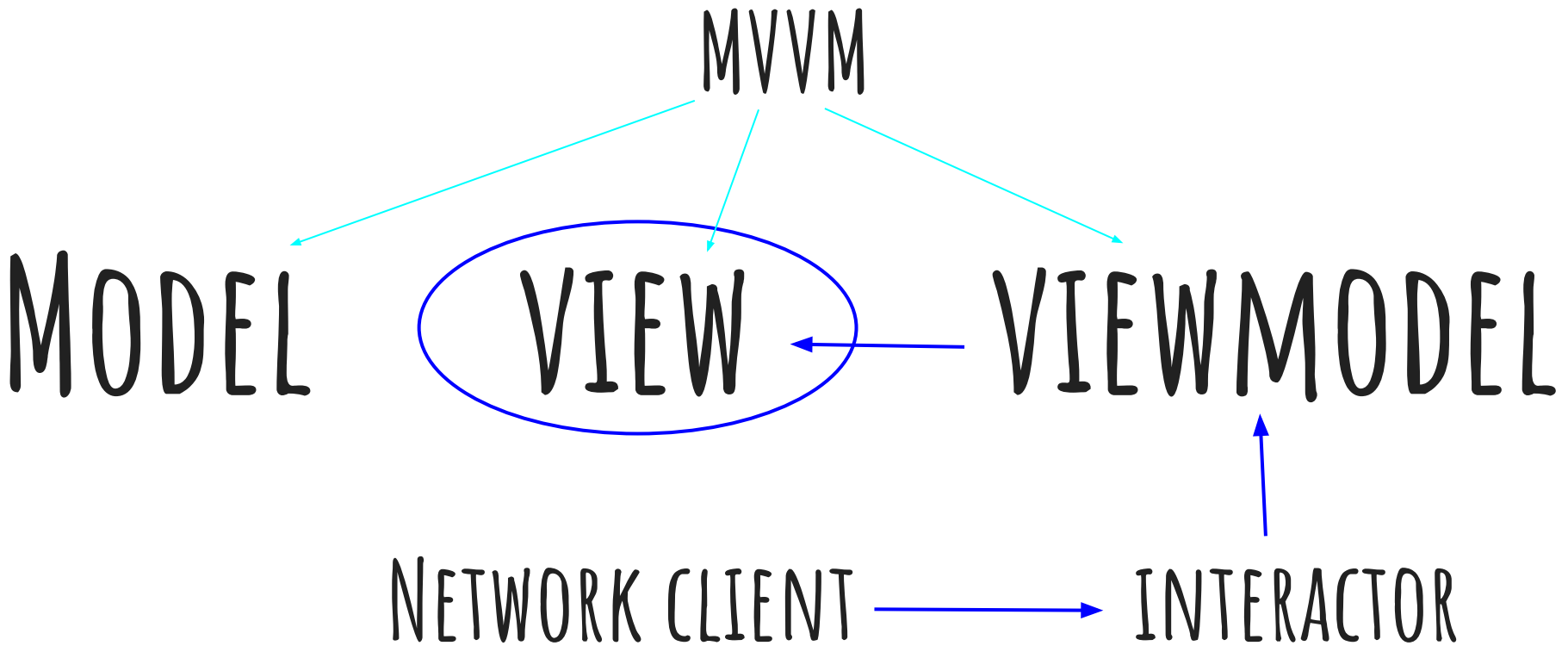
Strongly typed

Kinda like Java

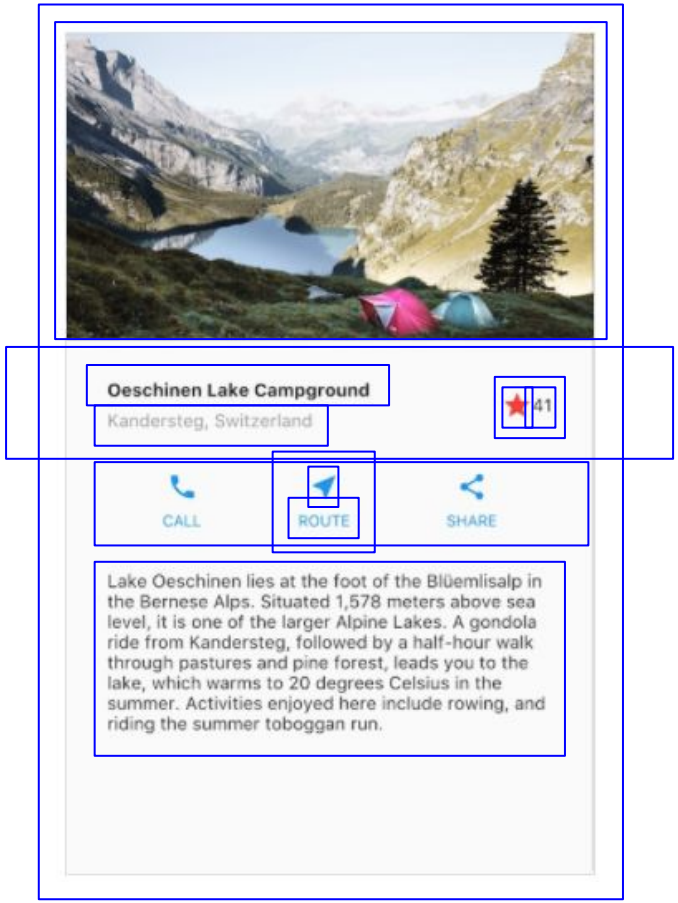
Let's look

```
class MyDartClass extends Object implements List {

  int myInteger = 1;
  String myString = "Sierra";
  bool _isPrivate = true;

  String _privateFunction() {
    return "This function is private";
  }

  void publicFunction(double myDouble) {
    if (myDouble.toInt() > myInteger) {
      print("$myDouble > $myInteger");
    }
  }

  void iterate(List<String> strings) {
    strings.forEach((element) {
      print(element);
    });
  }

}
```

```
import java.util.List;

public class MyJavaClass {

    int myInteger = 1;
    String myString = "Sierra";
    private boolean isPrivate = true;

    private String privateFunction() {
        return "This function is private";
    }

    void publicFunction(double myDouble) {
        if ((int) myDouble > myInteger) {
            System.out.print("$myDouble > $myInteger");
        }
    }

    void iterate(List<String> strings) {
        for (String string : strings) {
            System.out.print(string);
        }
    }

}
```


MVVM
MODEL
VIEW
VIEWMODEL
NETWORK CLIENT
INTERACTOR

Widgets == the view



👋Everything👋is👋a👋Widget👋

Composition (not inheritance)

Declarative UI

Material Design

Hot reload

(Some) Important Widgets

- Container
- Column & Row
- Inkwell for onTap
- Listview + Builder

Stateless vs stateful widgets (i.e. immutable vs mutable)

Stateless widgets cannot change their state during the runtime of the app => cannot be redrawn

```
class ErrorScreen extends StatelessWidget {

  @override
  Widget build(BuildContext context) {
    return Container(
      color: primaryColorLight,
    );
  }

}
```

How do I redraw the Widget?

```
void updateUI() {
  setState(() {
    // change something the UI depends on
  });
}
```

Stateful means the widget can be redrawn as many times as you want

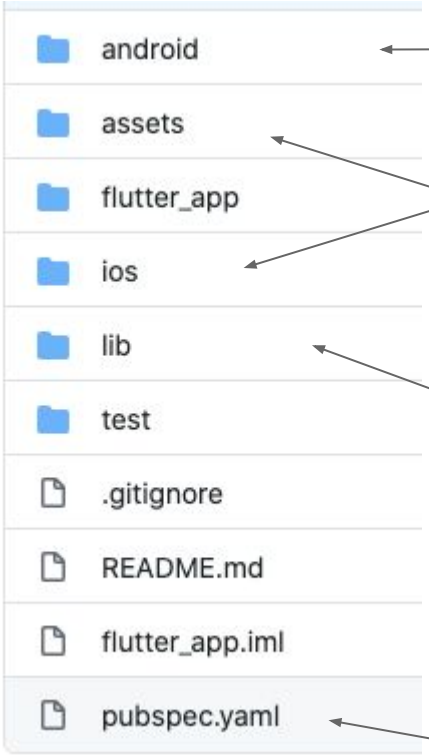
```
class ContactScreen extends StatefulWidget {

  @override
  State<StatefulWidget> createState() {
    return _ContactScreen();
  }

}

class _ContactScreen extends State<ContactScreen> {

  @override
  Widget build(BuildContext context) {
    return Container(
      color: primaryColorLight,
    );
  }

}
```

A TOUCH ON ASYNC

What will we be doing that is asynchronous? Network calls

```
Future<List<String>> buildAsyncList() async {
  var list = ["async", "is", "cool"];
  return list;
}
```

- Uses RxDart
- BehaviorSubject, PublishSubject, Stream(streams), Subscription (listen)
- Provider(provides) + StreamBuilder



Platform specifics things like API keys and permissions

Images and things

Most of your code!

app, data, ui, utils, values

Dependencies + libraries

A little about PRs

https://github.com/WomenWhoCodeCincy

- Issues are under the issue tab in each repo (some depend on each other - please read them carefully)
- When you're ready to start working on the issue, comment on the issue (please do not work on more than one issue simultaneously)
- Fork the repo and make a branch, make your changes, and push a PR
- Tag an admin to review your PR
- Collect hacktoberfest t-shirt

What happens now?

Three break out rooms - Database, Gatsby, Flutter

In each breakout room, we will do a live demo of how to contribute to the project - we will start from an issue, make a new branch, give more details, add our code, and open a PR.

This is your chance to see the actual code base and ask more specific questions in real time!

Then you can pick an issue and contribute. All the rooms will stay open with a moderator until one so you can get help if you have trouble or need a hint!

What should you do?

Put in the chat which room you would like to join and we will assign you to start.

What should you do if you want to switch rooms?

Exit the room you're currently in and you'll be dropped back in the main room. Mackenzie will be monitoring the main room and can reassign you to your new room!

Feel free to ask questions on slack as well (now through forever!)