AA Dart's expected things

As you saw, Dart is an easy-to-learn and effective language with some of the best features of Java, C#, JavaScript, and Python. In this lab we'll get familiar with Dart while we write some code that will be used in all future labs.

Creating a simple class

1. Inside your project's lib folder, create a subfolder called *state* (so "/lib/state") and inside there, create a file called film.dart (so "/lib/state/film.dart").

2. With the server running, open a web browser and make a request for all films at http://localhost:3008/api/films. Look at the shape of a film record. Using that information, create a Film class in that new film.dart file. We'll use this class later when reading from our RESTful API.
   Hints:
   - No need for a constructor yet.
   - You'll get an error about non-nullable types. To resolve that for now, assign it a reasonable initial value like an empty string or the number zero or DateTime.now().
   - Databases often use snake casing like "release_date". But Flutter prefers camel casing like "releaseDate".

Now let's try it out.

3. In main.dart, find the function triggered by the onPressed event. It's probably called _incrementCounter.

4. In that function, instantiate a Film object, giving it some placeholder values. (Hint: you'll need to import ./film.dart)

5. Print() your object.

6. Run and test. Press the button. Look in the debug console to see your created film. It may not be super-exciting at this point. It probably says "Instance of Film$" but if you print film.title or some other value you set, you'll see that the values are saved.

7. Bonus! If you have extra time, try to implement a "toString()" method. If you're a Java/C#/JavaScript dev, this'll be a fun challenge for you.

Reading from the server

In the real world you'll be reading data from an API server constantly. We wanted you to experience that as you're building your widgets throughout this course. We wanted to get you that experience as early as possible, but we don't cover reading from a server until much later. So we've given you some pre-written Dart code to read from the API server and write it to state. Just trust us for now. Later on we'll explain what all of these things mean.

8. Look in the starters folder. You'll see a file called repository.dart.starter. Copy that file into the lib/state folder as repository.dart.

9. Import it and film.dart at the top of main.dart like so:

```
import 'state/repository.dart';
import 'state/film.dart';
```

10. Add this inside the _MyHomePageState class

```
List<Film> _films = [Film()];
```

11. Find the _incrementCounter() method and replace it with this:

```
void _incrementCounter() {
  fetchFilms().then((films) => this.setState(() => this._films = films));
}
```

    Don't worry about the details of all this. It'll be clearer as we move through the course. But if you're just curious, this will make a GET request for all our films and when they arrive via HTTP, we set them in this._films and rerender the widget so that we can see some data.

    Now let's view the data:

12. Find the Text() that says something like 'You have pushed the button this many times:'. Replace that Text widget with this:

```
Text(_films[0].title,),
```

    As before, details will be coming later in the course. But this just says to take the title property of the first element in _films and display it on the screen.

13. Run and test. Click the button. You should see the title of the first film in the middle of your running app. Unless you got extremely lucky, there will be errors when you try this. Do your best to fix those errors on your own. Of course ask for help if you get desperate. ☺

Dynamic typing

We're reading films using a class. Let's look at reading some data without a class. We'll read film showings using a dynamic this time. You can see which you prefer.

14. First, declare a variable to hold the showings we'll read.

```
List<dynamic> _showings = [];
```

15. In the same _incrementCounter method as above, add this:

```
fetchShowings(film_id: 1, date: DateTime.now())
    .then((showings) => this.setState(() => this._showings = showings));
```

16. Print them out. Inside the build method at the top, put this:

```
@override
Widget build(BuildContext context) {
  for (dynamic showing in _showings) {    // <-- Add this...
    print(showing["showing_time"]);       // <-- and this...
  }                                       // <-- and this...
  return Scaffold(
```

17. Run and test. Hit the button. You should see a bunch of showings in the debug console.

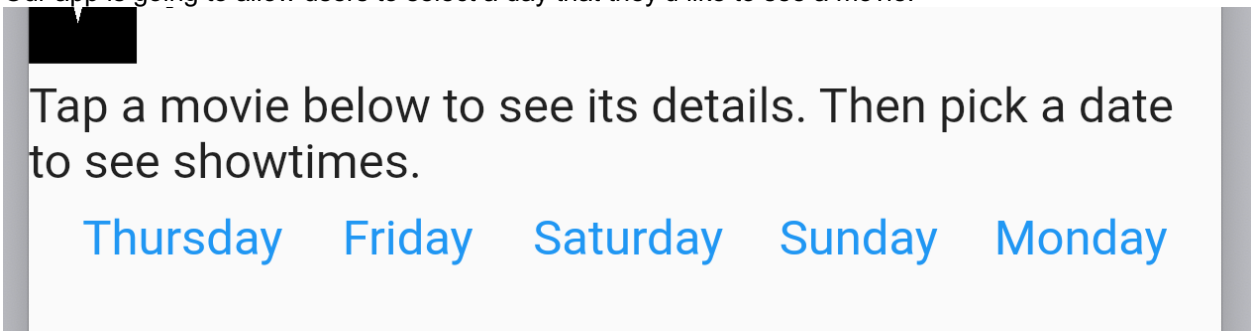
Now, how simple was that? No fuss, no muss, no creation of classes. Very straightforward to get the data. But of course there's no type-checking or runtime safety unless you code it manually.

	Class	Dynamic
Kind of like ...	Java, C#, C++	JavaScript, Python
Speed to develop	Meh	Much faster
Simplicity	Simpler to access properties	Simpler to create initially
Type safety	Yes! Fewer runtime exceptions	No. Might be runtime exceptions

As we go forward, you can decide which method you prefer.

Getting a list of days

Our app is going to allow users to select a day that they'd like to see a movie.



Let's create a function to generate the list of days so we can use it later on.

18. In the repository.dart file, create a function called makeConsecutiveDates(). It should receive howMany, an integer that specifies the number of days to create and startDate, a date from which to begin. It should return a List<DateTime>
19. Create a List<DateTime> called dates. Write a loop to go through five times, and add a DateTime to that list.
    Hint:
    - To get the next day, go someDate.add(Duration(days: 1));
20. Return that list of DateTimes.
21. Test it out by putting them also behind the button click and printing out each date. A good test would be to get five days starting with today.

That's enough fun for now. You can be finished.