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. With the server running, open a web browser and make a request for all films at http://localhost:3007/api/films. Look at the shape of a film record. Using that information, create a Film class in a file called Film.dart. We'll use this class later when reading from our RESTful API.

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

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

4. Print() your object.

5. Run and test. Press the button. Look in the debug console to see your created film.

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.

6. Your instructor will provide you with the location of a file called state.dart. Copy that file into the lib folder.

7. Import it and Film.dart at the top of main.dart like so:

```
import 'state.dart';
import 'Film.dart';
```

8. Add this inside the _MyHomePageState class

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

9. 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:

10. Find the Text() that says something like 'You have clicked 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. If it is null, show an empty string.

11. 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.

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

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

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

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

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

```
@override
Widget build(BuildContext context) {
  _showings.forEach((showing) => print(showing["showing_time"])); // <-- Add this line
  return Scaffold(
```

15. 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.