# 4A Value widgets

We've got some basic widgets created. In this lab we're going to give them some content and learn some cool other lessons along the way. As you're building widgets, they'll look pretty ugly. Don't worry about their appearance nearly as much as their content. We'll pretty them up later.

Creating the Checkout widget

After the user selects a movie, showing time, and the seats they'd like to reserve, they're going to want to review their selection and pay for the tickets. This is the Checkout process. We're going to create a widget for checking out.

It will eventually look something like this.
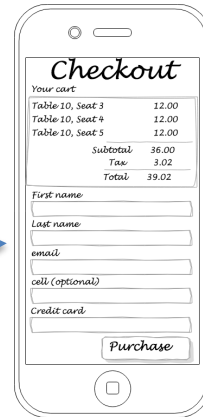
But we're building up to that.

1. Create a new Checkout stateful widget in the lib folder.

2. Since it will be a page-level widget, make sure the build returns a Scaffold() widget.

3. Give the Scaffold an AppBar() with a title of Text("Checkout").

4. Also give the Scaffold a body with a Column that has some children:
   - Text("Checkout")
   - Text("Cart will go here")
   - Text("Form will go here")

5. Make Checkout be your startup widget by editing main.dart and setting the MaterialApp's home property to Checkout() instead of Landing().

6. Run and test. Make sure you can see your new widget.

FABs look great with an Icon
Let's add a floating action button (a FAB) to Checkout().

7. Edit Checkout.

8. Add a FloatingActionButton (a FAB for short). Set the icon to represent "pay". Like a currency icon or something. You can decide what looks best.

9. Give your new FAB an onPressed event that just print()s something.

10. Test out how the FAB looks and works.

Fleshing out the Landing widget

11. Edit main.dart and make Landing your home widget again.

12. In the root of your project (and NOT under lib), create a new folder called assets. A company logo file called daam.png is in the starters folder. Put this file in your new folder.

13. Tell the project about this file by putting it pubspec.yaml kind of like this:

```
assets:
    – assets/daam.png
```

14. Edit landing.dart.

15. Add the Dinner-and-a-movie logo at the top as an Image.
    Hints:
    - Use Image.asset()
    - Provide a height. 75 would be a good size.
    - Provide a fit so it will maintain the right aspect ratio.

16. Put the business name "Dinner and a Movie" in a Text().

17. Have another Text() that says " "Tap a movie below to see its details. Then pick a date to see showtimes."

18. Ensure that the DatePicker widget and the FilmBrief widgets appear next.

19. Run and test Landing, making sure that everything shows up.

ShowingTimes

20. Edit the ShowingTimes widget. Make its constructor receive a film object and a selectedDate.
    Hints:
    - Make these properties required
    - Create class properties to match. You probably want to mark them as final.
    - *film* should be a Film
    - *selected_date* should be a DateTime
    - To format a date (below), you'll need to import 'package:intl/intl.dart'

21. In the build method, format the date string:

```
String selectedDateString =
    DateFormat.MMMMEEEEd().format(widget.selectedDate);
```

22. Give it a text like so: Text("Showing times for $selectedDateString for ${widget.film.title}"),
23. As things now stand you should be able to test ShowingTimes by looking at FilmBrief. Go ahead and do so.
    Hints:
    - Since the constructor of ShowingTimes now requires two parameters, you'll need to provide those values in FilmBrief.

Show all the times!

24. Still in ShowingTimes, declare a variable to show those showings:

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

25. Create an initState() lifecycle handler.

26. In there, make a call to fetchShowingTimes, passing the film's id and the selectedDate. It will return all the showings for that film for that date.

27. Put the showings in your new _showings variable.
    Hint:
    - You can print() those showings to make sure you know what is in them.
    - Remember, if you set the _showing variable properly but forget to call setState(), the user will never see the new values because the widget isn't re-rendered. Don't forget to call setState()!

28. Let's display those times in a column. Dealing with these dates in Dart can be hairy so let us spare you the wrestling. Here's a function that will take in a date string and return a Text() widget with a formatted time of day:

```
Text makeTextWidget(dynamic dateString) {
  DateTime showingTime = DateTime.parse(dateString);
  String timeString = DateFormat.jm().format(showingTime.toLocal());
  return Text(timeString);
}
```

Your mission will be to add a Column() to your widget and populate its children with a list of Text() widgets. Those widgets can be created by calling the makeTextWidget() function once for each thing in the list of _showings.

```
11:15 AM
1:15 PM
3:15 PM
5:15 PM
7:15 PM
9:15 PM
11:15 PM
```

29. See if you can do it without peeking at the answer but it might look something like this:

```
Column(
  children: _showings
      .map((s) => makeTextWidget(s["showing_time"]))
      .toList(),
),
```

30. If you test this out, you've got multiple films with multiple showings appearing which is awesome! But the screen may have overflowed. Want to fix it? Edit landing.dart and wrap Landing()'s Column with a SingleChildScrollView() widget. This is one of several ways to fix that. We can talk about the others later.

Displaying data in FilmBrief

The FilmBrief widget will be important. Its mission is to show brief information about a film, hence the name.

31. Examine the films that you've fetched from the server. Print them out or stop in the debugger and look at your list in Landing(). Take note of the data that you have available to you, especially the posterPath.

32. We're reading a string, a path to the poster. Since this is being read live at run time, we can't use a static asset that we're setting in stone in pubspec.yaml. Instead this will need to be a network image.

33. In FilmBrief, get the movie poster to show up. Give it a height of 100 and pick an appropriate BoxFit. Hint:

- `'${getBaseUrl()}/${film.posterPath}'`

34. Show the title in a Text()

35. Show the tagline in a Text()

36. Run and test. Make sure you can see a poster image followed by the film's title and tagline.

Showing the days in DatePicker

Almost finished. We need to display the days in DatePicker. We're going to do this one as more of a challenge with fewer instructions. See how you do. Ask your instructor for help if you get stymied.

37. Edit DatePicker(). You already have a function to makeConsecutiveDays() in repository.dart. import it and call it to create a list of DateTimes.

38. Use the .map() method to convert your list of five DateTimes into a list of Text() widgets. Hints:

- You can convert a DateTime to a day of the week string like this:

  `String text = DateFormat(DateFormat.WEEKDAY).format(date);`
- Don't forget to import intl for the DateFormat class.

39. Display those five Text() widgets in a Column()'s children property.

Congratulations, you're finished!