HW1 Dry Part – Oriel Avraham 315479808

Exercise 1 Dry part:

1)

The 2 lines of code that make the list infinitely scrolling are:

```
if (index >= _suggestions.length) {
  _suggestions.addAll(generateWordPairs().take(10));
}
```

If we remove these lines, and scroll to the end of the screen we get this error: "RangeError(index): Invalid value: Not in inclusive range 0..9: 10" and that is because at the bottom of the screen the value of index is >= to suggestions.length so after deleting these 2 lines we won't add more elements to suggestions and we will try to access suggestions at index but it's not in its bounds.

2)

A different method to construct such a list with dividers is using ListView.separated.

In my opinion using ListView.separated is better because we don't have to manually place the separators like we did with the indices in our exercise (index = i~/2 etc), but instead we can comfortably use seperatorBuilder in order to define our seperators.(using Widgets only instead of adding extra code).

3)

We need setState() inside the onTap() handler because in our case this handler needs to change the UI(scroll up or down), and calling setState() notifies the framework that the internal state of this object has changed in a way that might impact the UI in this subtree, and this causes the framework to schedule a build for this state object, which draws the UI according to the widget's current state.

Exercise 2 Dry part:

1)

MaterialApp widget is a convenience widget that wraps a number of widgets that are commonly required for Material Design applications.

home: The widget for the default route of the app. This is the route that is displayed first when the application is started normally, unless initialRoute is specified.

theme: Default visual properties, like colors fonts and shapes, for this app's material widgets.

routes: The application's top-level routing table.

2)

In general, state and access the state at different times or maintain it while modifying the widget tree. In our case, the key argument is needed because Dismissibles are commonly used in lists and removed from the list when dismissed, and that is done with the keys. Without keys, the default behavior is to sync widgets based on their index in the list, which means the item after the dismissed item would be synced with the state of the dismissed item. Using keys causes the widgets to sync according to their keys and avoids this pitfall.

MEME IN THE NEXT PAGE!!!!!!


MEME
ONE
100 PLEASE
ME
SEGEL
MEME
imgflip.com