Mobile Application Development

Week 6. Using Keys / Team Project

Joon-Woo Lee

School of Computer Science and Engineering
College of Software
Chung-Ang University

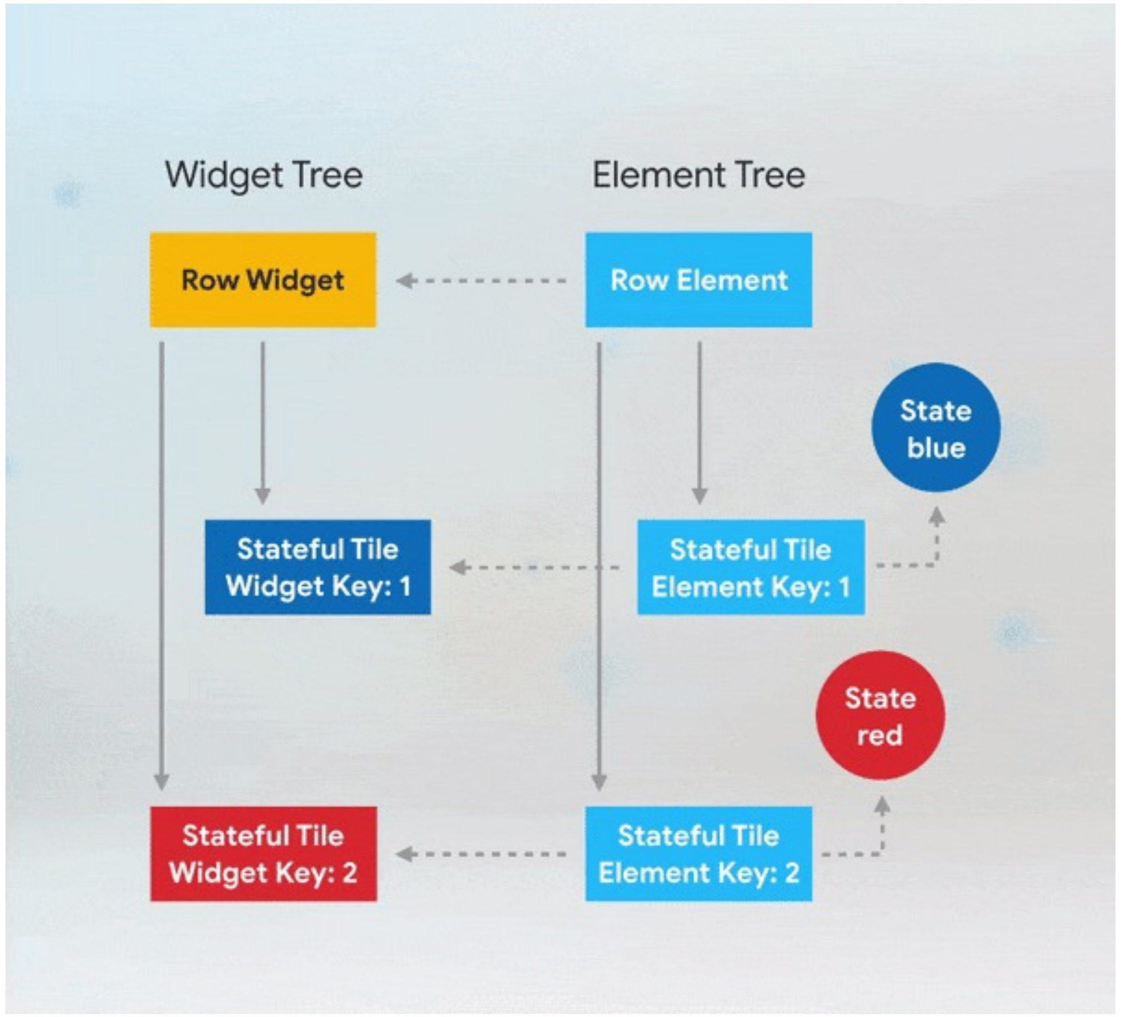
How the Flutter uses keys

- Assume that we have one parent widget (e.g. Row or ListView) and ma ny children widgets with the same type (e.g. Padding with Container or ListTile)
- In the subtree of each child widget, there is at least one stateful widget .
- Then we have given a change to the children widgets (e.g. swapping, in serting, or removing)

How the Flutter uses keys

- Then, the key of some Elements doesn’t match the key of the correspo nding widget.
- This causes Flutter to deactivate those elements and remove the refere nces to the Tile Elements in the Element Tree, starting with the first on e that doesn’t match.
- Flutter looks through to non-matched children of the Row for an eleme nt with the correct corresponding key.
- It finds a match and updates its reference to the corresponding widget.

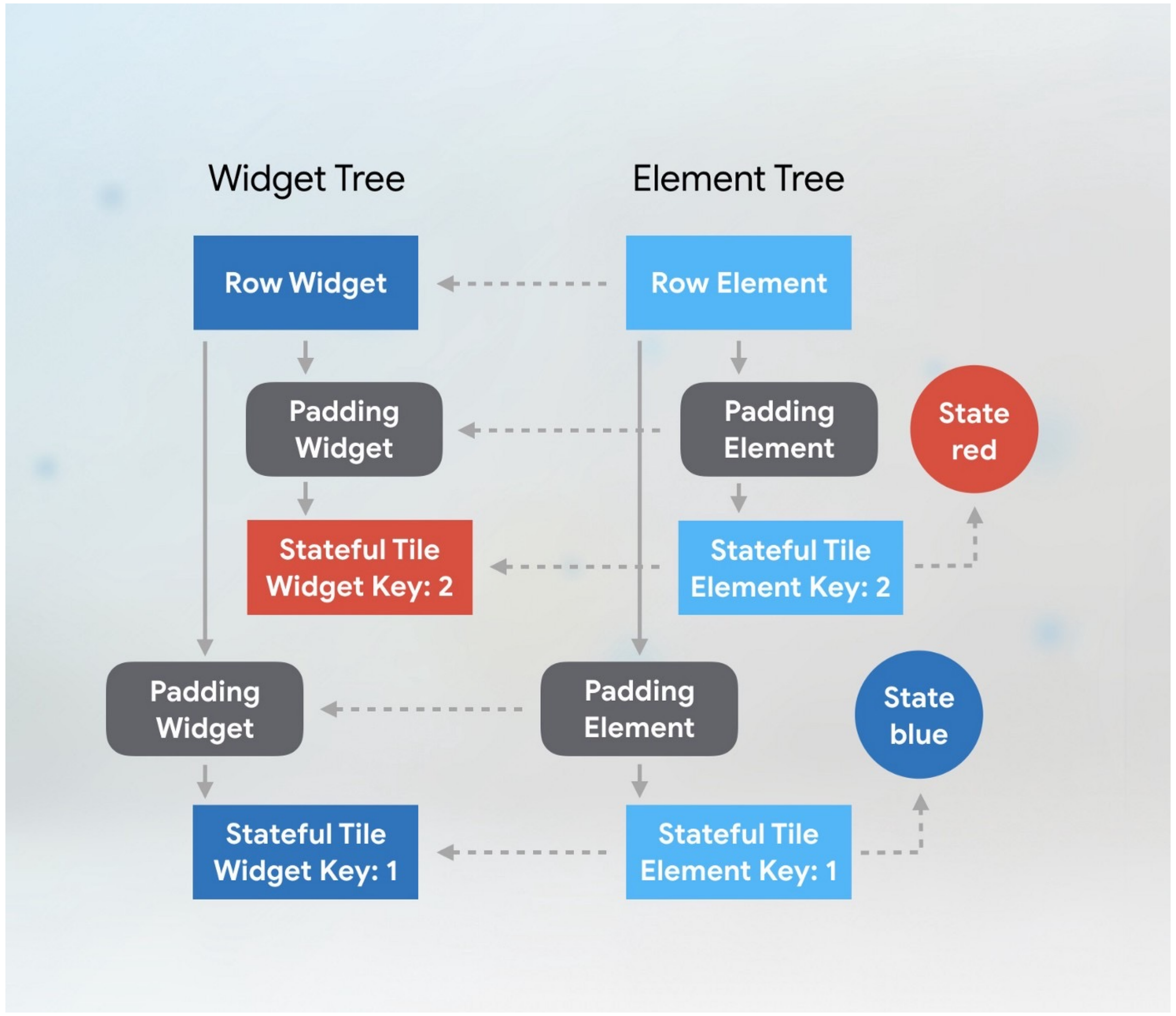
How the Flutter uses keys


Widget Tree
Element Tree
Row Widget
Row Element
State blue
Stateful Tile Widget Key: 1
Stateful Tile Element Key: 1
State red
Stateful Tile Widget Key: 2
Stateful Tile Element Key: 2

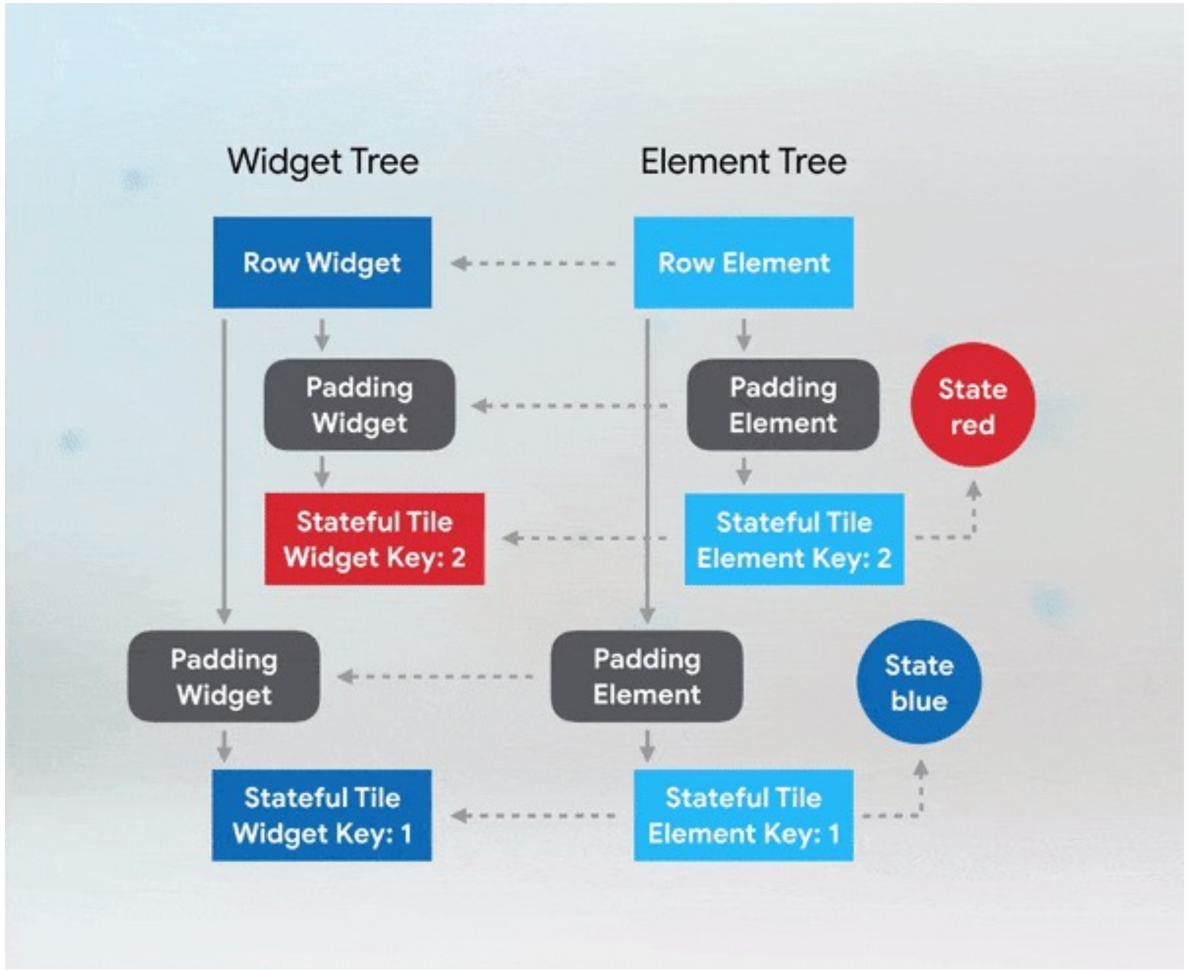
Where we put the keys (Local keys)

- If you need to add keys to your app, you should add them at the ***top*** of the widget subtree with the state you need to preserve.
- A common mistake: Putting a key on the first stateful widget.
- Flutter’s element-to-widget-matching algorithm looks at one level in th e tree at a time.
- When matching up widget to elements, Flutter only looks for key matc hes within a particular level in the tree.

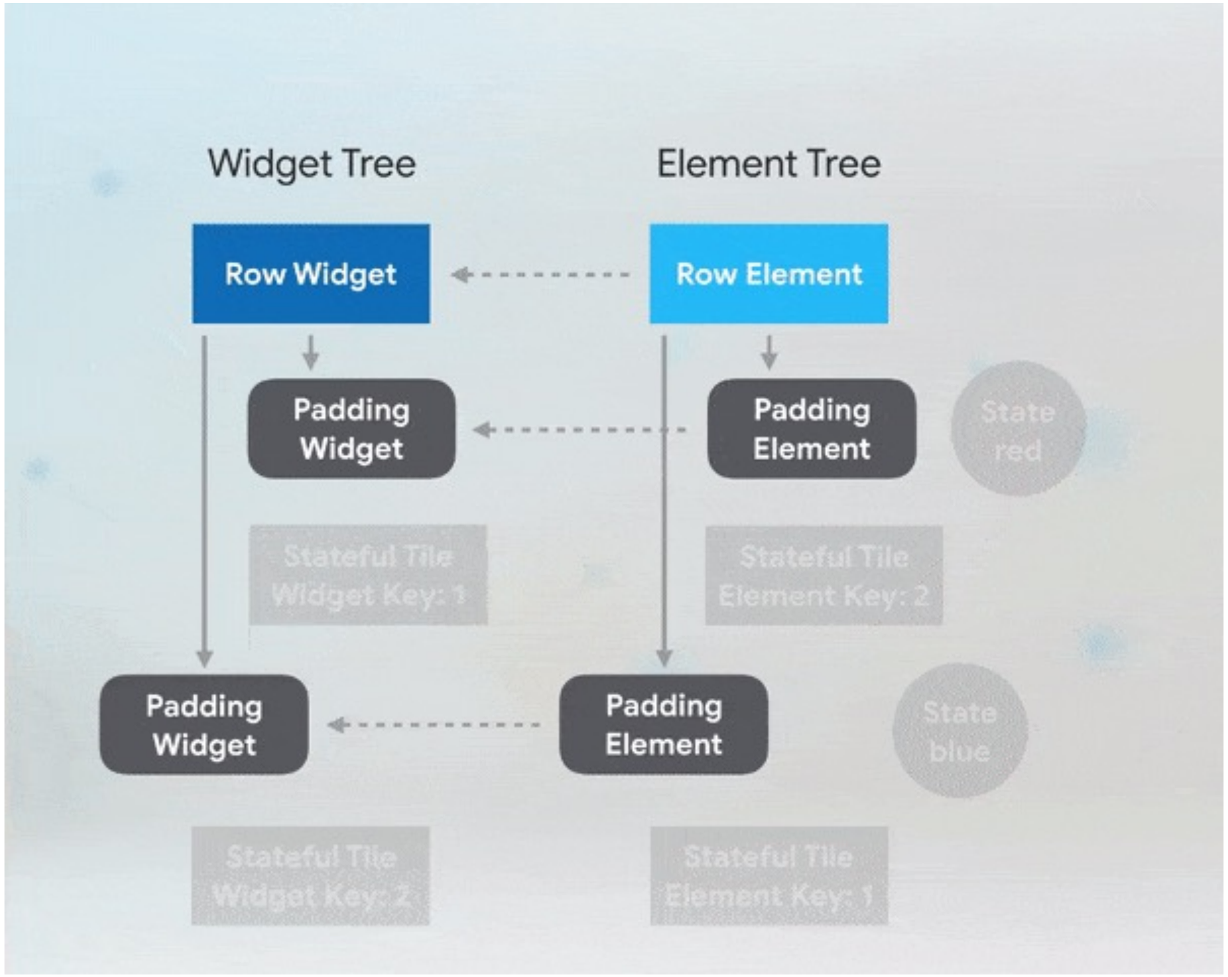
Where we put the keys



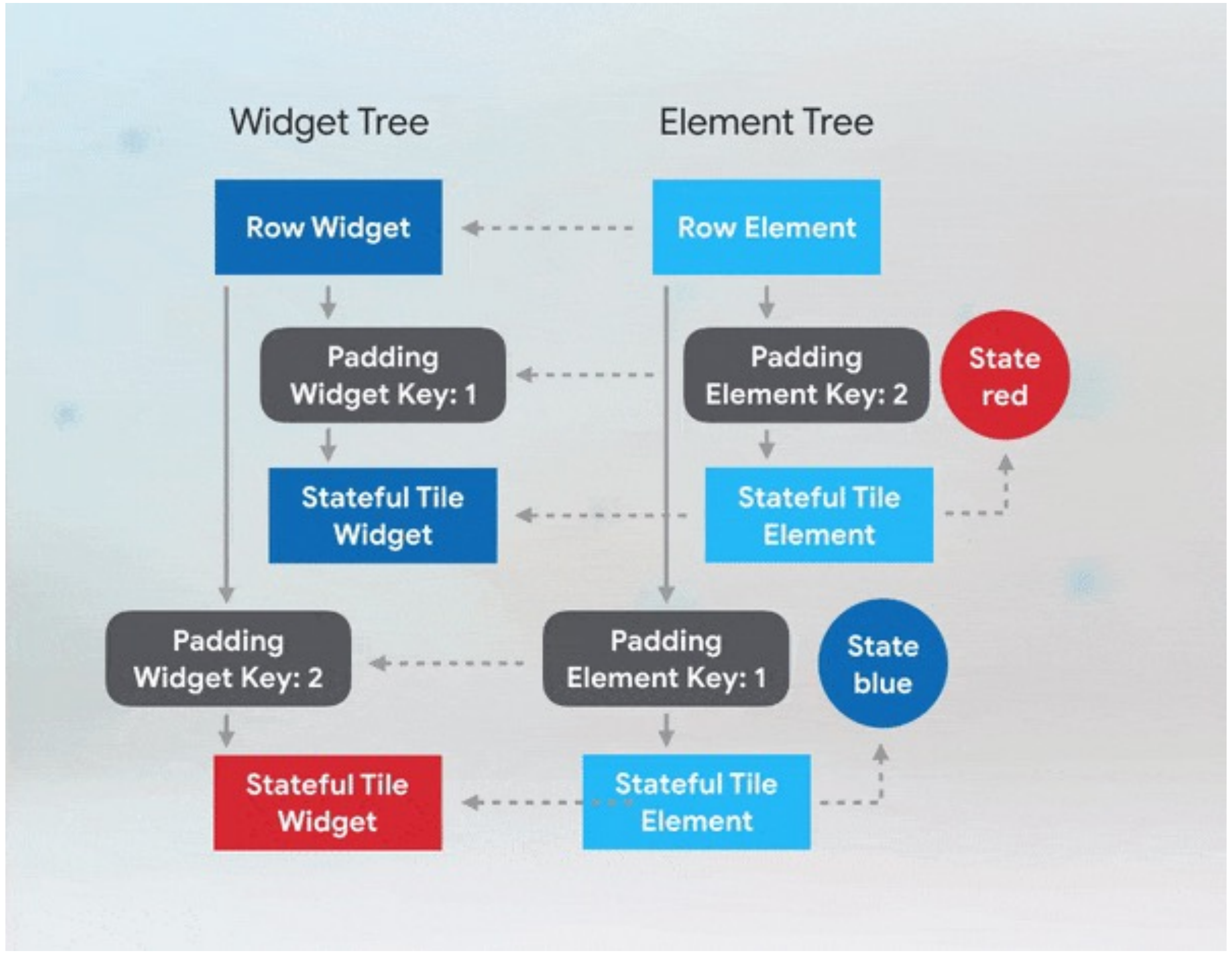
Where we put the keys



Where we put the keys



Where we put the keys



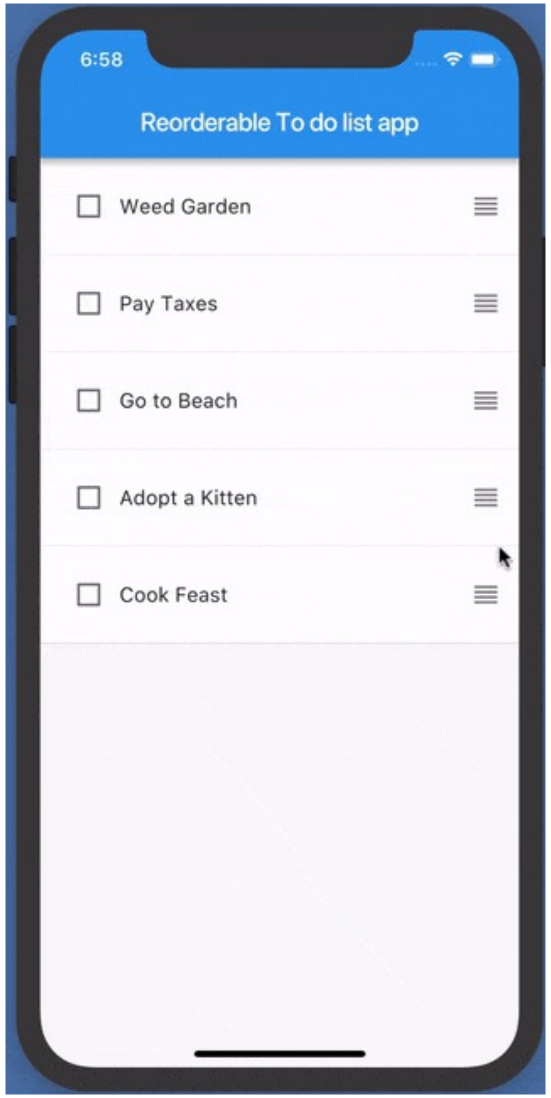
The types of keys we can use

- There are several **local keys** and the **global key**
- Local keys:
  - ValueKey
  - ObjectKey
  - UniqueKey
  - PageStorageKey
- Global key: GlobalKey

Value Key

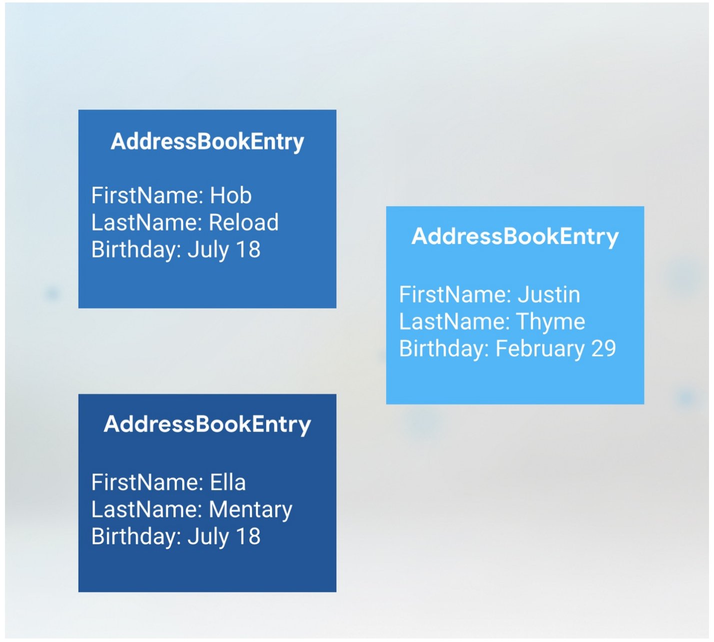
- If you expect some value in the widget to be constan t and unique, then you can use **ValueKey** with this u nique value.
- For example, you might expect the text of a To-do ite m to be constant and unique in To-do list app.

```
return TodoItem(
 key: ValueKey(todo.task),
 todo: todo,
 onDismissed: (direction) => _removeTodo(context, todo),
);
```



Object Key

- If you expect that any of the individual fields might be the same as anoth er entry but the combination is unique, then you can use **ObjectKey** with these combination.
- For example, in an address book app, any of the individual fields like a fir st name or birthday might be the same as another entry, but the combin ation is unique.

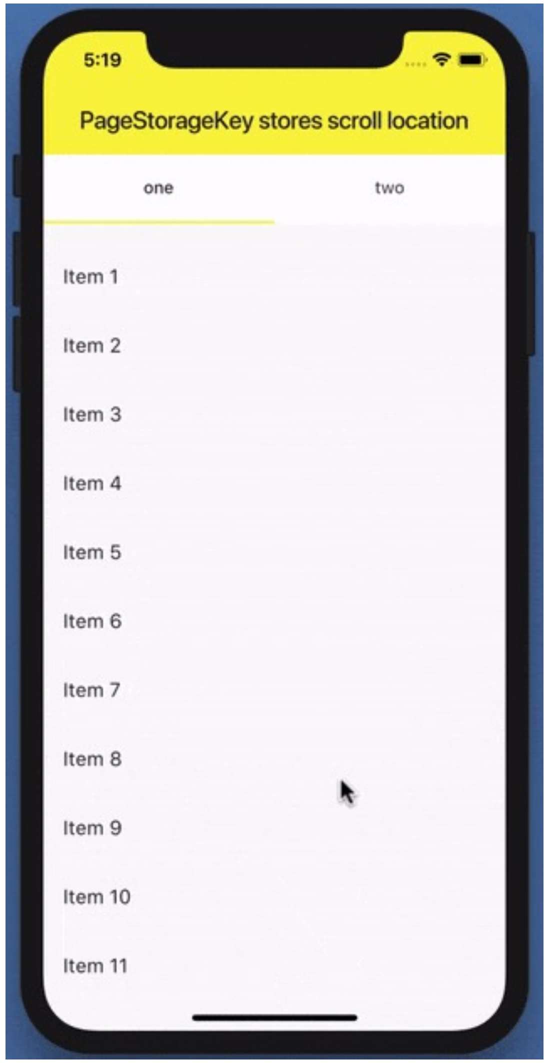


Unique Key

- If you have multiple widgets in your collection with the same value or if y ou want to really ensure each widget is distinct from *all* others, you can u se the **UniqueKey**.
- In the swapping tile example, we didn’t have any other constant data tha t we’re storing in our tiles, which is the reason we use the unique key.
- If you construct a new UniqueKey inside a build method, the widget usin g that key will get a different, *unique* key every time you the build metho d re-executes.
  - This will eliminate any benefits of using keys.

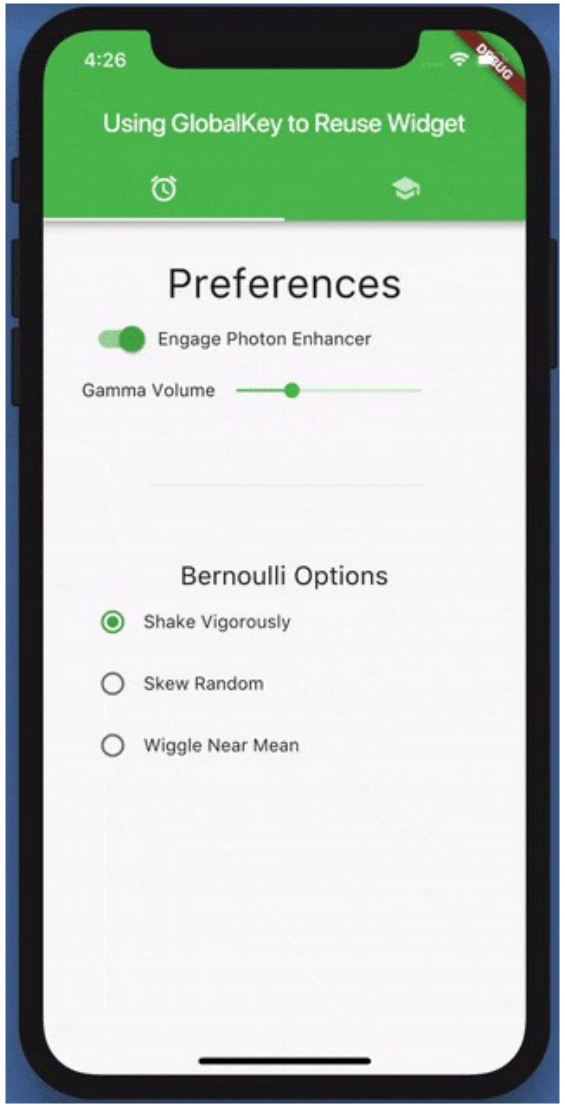
Page Storage Key

- **PageStorageKey**s are specialized keys that store a user's scroll location so that the app can preserve it for later.



Global Key

- **GlobalKeys** have two uses:
  - They allow widgets to change parents anywhere in yo ur app without losing state.
  - They can be used to access information about another widget in a completely different part of the widget tre e.
- The Form widget uses the GlobalKey, and this is t he only one recommended case using GlobalKey.



Team Project

Main goal of team project

- To have an experience of making a complete app with your team membe rs!
    - It does not need to a creative and unique app that do not exist before.
    - A specialized app for specific target is enough for novelty in this team project, ev en if this app is very similar to other apps in terms of functions.

What you can

- Referring to other existing apps for designs or functions.
- Use some parts of Flutter codes in the documentation or other sources for **gene ral use** in the Web.
  - e.g., Blog post explaining some widgets or concepts or Stack exchange
- Using other advanced widgets not dealt with in the class. (Recommended!)
- Using other frameworks as a database or a server.
  - The main framework should be Flutter.
- Submitting the output app to some other competitions **after the mid-term exa m**.

What you cannot

- Use other Flutter codes for development of some apps.
- Use other SDK for mobile development as the main framework.
- Use your pre-developed code used in other classes or in other external co mpetitions.

Assessment

- Assessment in this class
  - Attendance (10%)
  - Mid-term exam (30%)
  - Final exam (30%)
  - Assignment (30%)
    - Assignments in the class (5%)
    - Team project (25%)

- There are two area for assessment for team project.
  - Assessment for minimum requirements (10%)
  - Peer Review (15%)

Assessment

- Minimum Requirement (10%)
    - Submission of proposal (2%)
    - Presentations (4%)
    - Meeting the minimum specifications (4%)
        - Whether the app uses the core widgets dealt with in this class.
- Peer Review (15%)
    - Novelty(3%): Whether this app is new. (to some specific targets)
    - Completeness(4%): Whether each function works well.
    - Variety(4%): Whether the app includes various functions related to the main goal.
    - Convenience(4%): Whether the app is convenient to use. (Documentation may be i mportant.)

Proposal Presentation

- The presentation material must be in English, but you may give a presenta tion in any language.
- The proposal presentation is important in many cases of development.
- Each team will have 5 minutes for the presentation, and the presentation will be in the class of 10/27.
- Main goal: Introduce your plans to your classmates and make a good impr ession!