جامعة حضرموت
Hadhramout University

قسم تقنية المعلومات
Information Technology Department

كلية الحاسبات وتقنية المعلومات
College of Computers & Information Technology

STUDENT NAME: AHMED MUBARAK ABBAD

LECTURER: MUHAMMAD KAADAN.

ASSIGNMENT: THE DIFFERENT BETWEEN:

1) CROSS AND NATIVE PLATFORM.
2) STATELESS AND SATAEDUL WIDGET.

SCHOOL: HADRAMOUT UNIVERSITY
COURSE TITLE: MOBILE APPLICATION

Definition

Stateless and stateful widgets are two different approaches to creating widgets in Flutter. According to the official documentation, stateless widgets are widgets that do not have any internal state and do not change their appearance or behavior during the runtime of an application. They are immutable and only depend on the configuration information and the build context. Stateful widgets are widgets that have an internal state and can change their appearance or behavior during the runtime of an application. They are mutable and can be updated by calling the setState method. They can access and modify their state and rebuild themselves whenever the state changes.

Examples

Stateless and stateful widgets can be used to create different types of user interface elements in Flutter. Some examples of stateless widgets are text, icons, buttons, etc. Stateless widgets are useful for creating static parts of the user interface that do not need to respond to user interactions or data changes. For example, the following code snippet shows how to create a stateless widget that displays a text message:

```
class MessageWidget extends StatelessWidget {
  final String message;

  MessageWidget(this.message);

  @override
  Widget build(BuildContext context) {
    return Text(message);
  }
```

Some examples of stateful widgets are checkboxes, sliders, forms, etc. Stateful widgets are useful for creating dynamic and interactive parts of the user interface that need to respond to user interactions or data changes. For example, the following code snippet shows how to create a stateful widget that displays a checkbox and a text message that changes according to the checkbox value:

```
class CheckboxWidget extends StatefulWidget {
  @override
  _CheckboxWidgetState createState() => _CheckboxWidgetState();
}

class _CheckboxWidgetState extends State<CheckboxWidget> {
  bool isChecked = false;
```

```
  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        Checkbox(
          value: isChecked,
          onChanged: (value) {
            setState(() {
              isChecked = value;
            });
          },
        ),
        Text(isChecked ? 'Checked' : 'Unchecked'),
      ],
    );
  }
}
```

Advantages and Disadvantages

Stateless and stateful widgets have their own advantages and disadvantages, depending on the requirements and preferences of the developers and the users. Some of the advantages and disadvantages are:

- **Advantages of stateless widgets**:
    - They are faster and easier to build and render than stateful widgets, as they do not require any state management or update logic.
    - They are more reusable and modular, as they can be easily composed with other widgets and used in different contexts.
    - They are more testable and debuggable, as they do not have any side effects or dependencies on external factors.

- **Disadvantages of stateless widgets**:
    - They cannot provide dynamic and interactive features that require state management, such as animations, transitions, gestures, etc.
    - They cannot store any information that can be reused or updated later, such as user preferences, form data, etc.
    - They may not reflect the latest data or state of the application, as they do not rebuild themselves when the data or state changes.

- **Advantages of stateful widgets**:
    - They can provide dynamic and interactive features that require state management, such as animations, transitions, gestures, etc.
    - They can store any information that can be reused or updated later, such as user preferences, form data, etc.
    - They can reflect the latest data or state of the application, as they rebuild themselves when the data or state changes.
- **Disadvantages of stateful widgets**:
    - They are more complex and harder to build and maintain than stateless widgets, as they require state management and update logic.
    - They are less reusable and modular, as they may have specific dependencies or assumptions on the context or the data.
    - They are less testable and debuggable, as they may have side effects or dependencies on external factors.

Conclusion

Stateless and stateful widgets are two different types of widgets in Flutter that have different characteristics, use cases, and implications. Stateless widgets are widgets that do not have any internal state and do not change their appearance or behavior during the runtime of an application. Stateful widgets are widgets that have an internal state and can change their appearance or behavior during the runtime of an application. Developers should use the appropriate type of widget for the appropriate purpose, and avoid unnecessary stateful widgets that can make the code more difficult to understand and the app less efficient.

Source(s)

1. Stateless and Stateful Widgets in Flutter: Examples and Best Practices ...
2. dart - What is the relation between stateful and stateless widgets in ...
3. Exploring Stateless and Stateful Widgets | by Olvine george - Medium
4. Stateless vs Stateful Widgets in Flutter | by Günseli Ünsal - Medium
5. Flutter Basics: The differences between Stateless Widget and Stateful ...