Drawer Widget in Flutter

Drawer widget is used to provide access to different destinations and functionalities provided in your application. It is represented by three horizontal parallel lines on the upper end of the scaffold. It has a horizontal movement from the edge of the Scaffold that navigates the link to different routes in the flutter app.

In order to use the app drawer you need to import the material component package that is " package: flutter/material.dart."

The Navigating AppDrawer is divided into three sections namely header, body, and footer. The idea is about having a navigator with a couple of items as the drawer's child that will be navigated to different destinations on getting tapped. All children of a Drawer widget are usually in ListView and initially, only the DrawerHeader is present in the UI which when tapped extends horizontally.

Constructors:

Syntax:

```
Drawer({Key key, double elevation: 16.0, Widget child, Stri
ng semanticLabel})
```

Properties:

- **child:** The widgets below this widget in the tree.
- **hashCode:** The hash code for this object.
- **key:** Controls how one widget replaces another widget in the tree.
- **runtimeType:** A representation of the runtime type of the object.
- **elevation:** The z-coordinate at which to place this drawer relative to its parent.
- **semanticLabel:** The semantic label of the dialogue used by accessibility frameworks to announce screen transitions when the drawer is opened and closed.

Important Function:

- **build**(BuildContext context) → Widget: This method specifies the part of the UI rendered by the widget. This method is called when the *Drawer* widget is inserted into the tree in a given *BuildContext* and when the dependencies of the *Drawer* widget change.

Implementation:

```
@override
Widget build(BuildContext context) {
  assert(debugCheckHasMaterialLocalizations(context));
  String? label = semanticLabel;
  switch (Theme.of(context).platform) {
    case TargetPlatform.iOS:
    case TargetPlatform.macOS:
      break;
    case TargetPlatform.android:
    case TargetPlatform.fuchsia:
    case TargetPlatform.linux:
    case TargetPlatform.windows:
      label = semanticLabel ?? MaterialLocalizations.of(con
text).drawerLabel;
  }
  return Semantics(
    scopesRoute: true,
    namesRoute: true,
    explicitChildNodes: true,
    label: label,
    child: ConstrainedBox(
      constraints: const BoxConstraints.expand(width: _kWid
th),
      child: Material(
        elevation: elevation,
        child: child,
      ),
    ),
  );
}
```