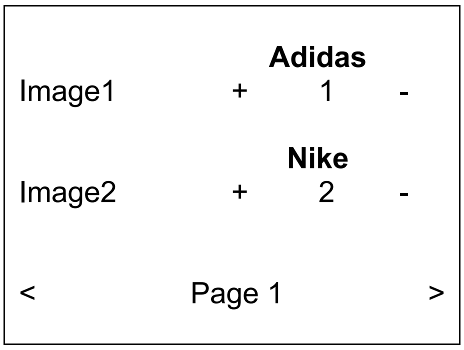
Introducing Flutter

Key Points	Notes

1. **How Flutter Works?**
    a. Flutter is reactive, declarative and composable view-layer library, which mean you can build a mobile UI by integrating together a few of smaller widgets (components).

    b. Flutter uses the concept of **everything is a widget**, which a widget is a Dart classes that represent their view.

    c. Example:



```
build(BuildContext context){

  return Column(
    //…
    Image(),
    Text("Nike"),

    //…
    IconButton(
     icon: Icon(Icons.chevron_left),
    ),

    Text("Page $page_num"),
    //…
  );

}
```

Summary

Key Points

Notes

d. Some widgets have state that keep data and update them when needed. For example, quantity of the above shopping cart, the state keep track number of quantity purchased or changed by the user.

e. When the widget's state changes, the framework is notified and compared with the new widget description to the previous description and make changes only to the related widgets that are necessary.

2. Everything is a Widget

a. A widget can define any aspect of an application and they consist of the following:
   i. Layout – `Rows`, `Column`, `Scaffold`, `Stack`
   ii. Structures – `Button`, `Toast`, `MenuDrawer`
   iii. Styles – `TextStyle`, `Color`
   iv. Animations – `FadeInPhoto`, tranformations
   v. Positioning and alignment – `Center`, `Padding`

3. Composing UI with Widgets

a. Flutter favours composition over class inheritance, which allows you to make your own unique widgets. A majority of widgets are combinations of smaller widgets.

b. Example:

```
class AddToCartButton extends StatelessWidget {
 // ... class members
  @override
 build() {
    return Center(
      child: Button(
        child: Text('Add to Cart'),
      ),
    );
  }
}
```

Summary

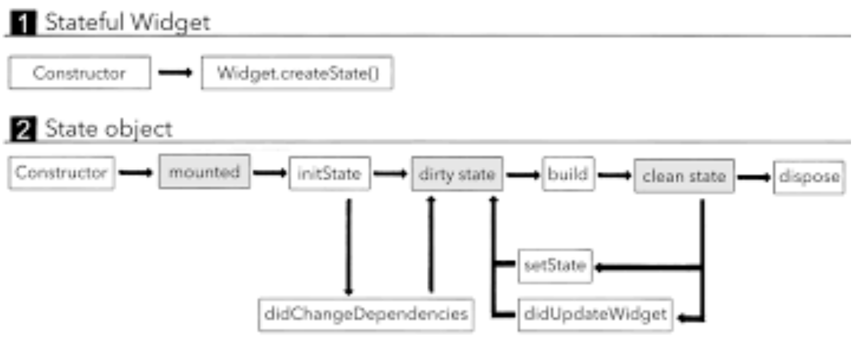
Key Points	Notes
	4. Widgets Types a. Most widgets fall under two categories: i. Stateless (StatelessWidget) 1. No information is kept in it and all widget's state or configuration is passed into it. 2. Its job is to display information and UI. 3. Example is AddToCartButton widget. ii. Stateful (StatefulWidget) 1. It manages a stateful data that keep changes in the widget. 2. It is associated with `State` object and using special method called `setState` to tell flutter to update or change or repaint the widget. b. Example:

```
Widget build(BuildContext context) {
  return Container(
    child: Row(
      children: List<Widget>[
         IconButton(
          icon: Icons.subtract,
          onPressed: () {
            setState(() {
               this.quantity--;
            });
          }),
        new Text("Qty: ${this.quantity}),
        new IconButton(
          icon: Icons.add,
          onPressed: () {
            setState(() {
              this.quantity++;
            });
          }),
      ],
    )
  );
}
```

Summary

Key Points	Notes
	5. Stateful Widget's Life Cycle

a. The process of building and updating widgets is called widget's life cycle.



Flutter Widget's Life Cycle (Windmill, 2020)

i. When app is loaded, Flutter creates the object, which creates the `State` object associated with the widget.

ii. As soon as the widget is mounted, Flutter calls `initState`

iii. After the state is initialized, Flutter builds the widget.

iv. The widget is sitting and waiting for three (3) possible events:

1. User navigate to different part of the application, which the state can be disposed.

2. Other widget have updated and changed the configuration of the widget (`didUpdateWidget`).

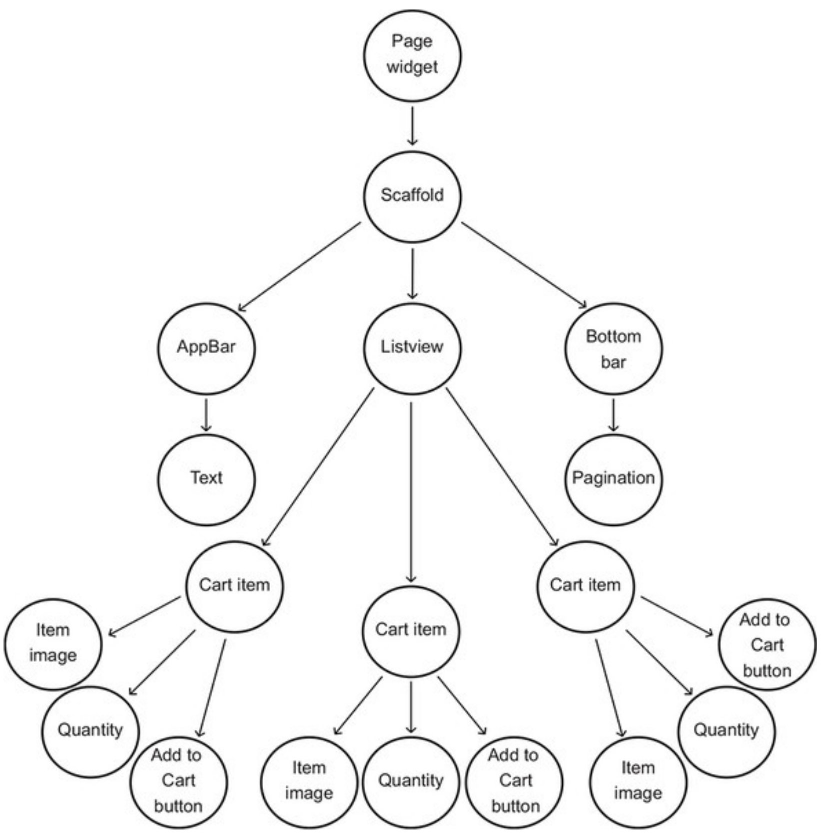
3. If the event `onPressed` is called the `setState` is executed and update the internal state and rebuild and re-render.

Summary

Key Points

Notes

6. **Flutter Rendering: Under the Hood**
    a. Flutter application is composed of a giant widget tree.



    b. Refer to one the `CartItem` widgets. This widget is Stateful and each of its children relies on the state of that widget. When the state of the `CartItem` widget is updated, the rendering process in the subtree from this point begins.

    c. Flutter widgets are reactive because they respond to new information from an outside source or `setState` and Flutter rebuilds what it needs to.

Summary

Key Points	Notes
	7. References a. Windmill, E. (2020). Flutter in Action (1st Ed.). USA: Manning Publications. b. Flutter Official Documentation. Retrieved on 1 December 2020 from https://flutter.dev/docs
Summary	