Yandex

Yandex

Widgets

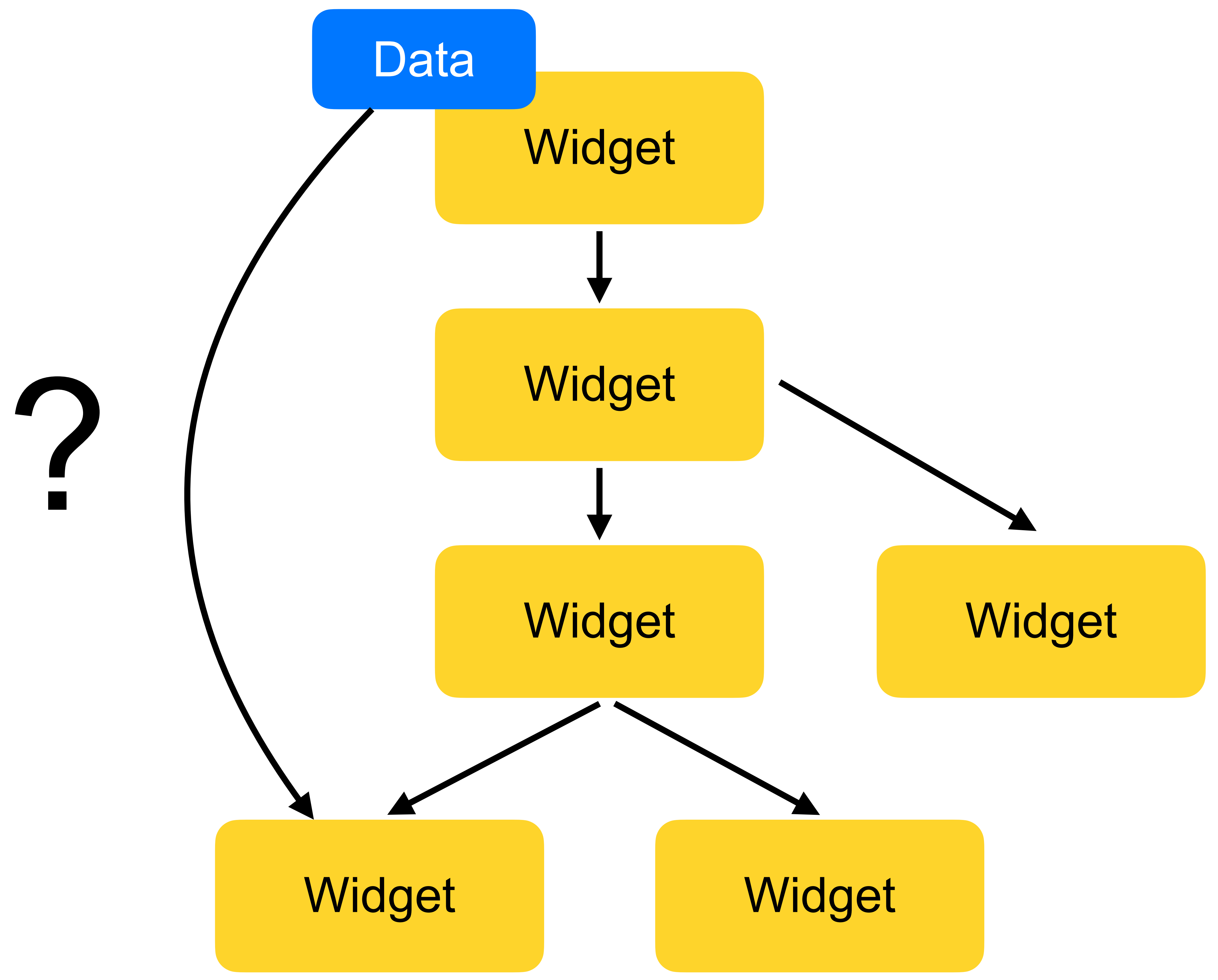
Sergey Koltsov, Yandex.Pro Team Lead

Table of Content

1. InheritedWidget
2. Scrollable widgets
3. Builders

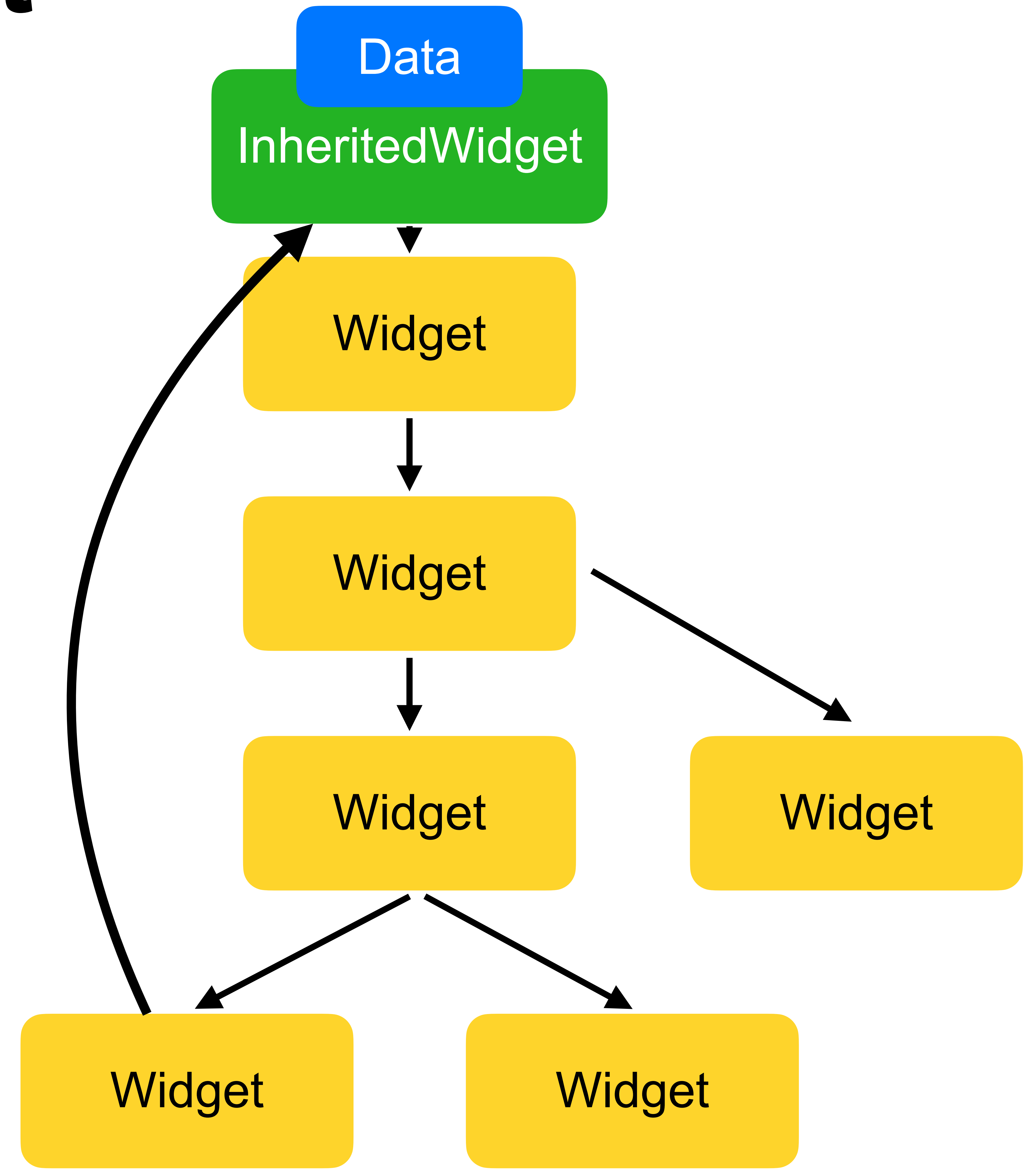
InheritedWidget

InheritedWidget



https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html

InheritedWidget



https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html

InheritedWidget

```
class DataProvider extends InheritedWidget {
  const DataProvider({
    Key? key,
    required this.data,
    required Widget child,
  }) : super(key: key, child: child);

  final String data;

  static String? of(BuildContext context) {
    final DataProvider? result =
        context.dependOnInheritedWidgetOfExactType<DataProvider>();
    return result?.data;
  }

  @override
  bool updateShouldNotify(DataProvider oldWidget) => data != oldWidget.data;
}
```

https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html

InheritedWidget

```dart
class DataProvider extends InheritedWidget {
  const DataProvider({
    Key? key,
    required this.data,
    required Widget child,
  }) : super(key: key, child: child);

  final String data;

  static String? of(BuildContext context) {
    final DataProvider? result =
        context.dependOnInheritedWidgetOfExactType<DataProvider>();
    return result?.data;
  }

  @override
  bool updateShouldNotify(DataProvider oldWidget) => data != oldWidget.data;
}
```

https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html

InheritedWidget

```
class DataProvider extends InheritedWidget {
  const DataProvider({
    Key? key,
    required this.data,
    required Widget child,
  }) : super(key: key, child: child);

  final String data;

  static String? of(BuildContext context) {
    final DataProvider? result =
        context.dependOnInheritedWidgetOfExactType<DataProvider>();
    return result?.data;
  }

  @override
  bool updateShouldNotify(DataProvider oldWidget) => data != oldWidget.data;
}
```

https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html

InheritedWidget

```
class DataProvider extends InheritedWidget {
  const DataProvider({
    Key? key,
    required this.data,
    required Widget child,
  }) : super(key: key, child: child);

  final String data;

  static String? of(BuildContext context) {
    final DataProvider? result =
        context.dependOnInheritedWidgetOfExactType<DataProvider>();
    return result?.data;
  }

  @override
  bool updateShouldNotify(DataProvider oldWidget) => data != oldWidget.data;
}
```

https://api.flutter.dev/flutter/widgets/InheritedWidget-class.html

Scrollable widgets

Scrollable widgets

- ListView
- SingleChildScrollView
- PageView
- Slivers

ListView

- `ListView(children: [])`
- `ListView.builder(builder: (context, index) => Widget(), itemCount: 0)`
- `ListView.separated`
- `SingleChildScrollView`

https://api.flutter.dev/flutter/widgets/ListView-class.html
https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html

ListView

- ScrollController
- ScrollPhysics
- Axis scrollDirection

https://api.flutter.dev/flutter/widgets/ListView-class.html
https://api.flutter.dev/flutter/widgets/SingleChildScrollView-class.html

PageView

- PageController
- ScrollPhysics
- onPageChanged

https://api.flutter.dev/flutter/widgets/PageView-class.html

Slivers

In the next episode!

Builders

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: Text(
        'hasAppBar: ${Scaffold.of(context).hasAppBar}',
      ),
    ),
  );
}
```

Builder

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: Builder(
        builder: (context) {
          return Text(
            'hasAppBar: ${Scaffold.of(context).hasAppBar}',
          );
        }
      ),
    ),
  );
}
```

https://api.flutter.dev/flutter/widgets/Builder-class.html

Builder

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Center(
      child: Builder(
        builder: (context) {
          return Text(
            'hasAppBar: ${Scaffold.of(context).hasAppBar}',
          );
        }
      ),
    ),
  );
}
```

https://api.flutter.dev/flutter/widgets/Builder-class.html

LayoutBuilder

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: LayoutBuilder(
      builder: (BuildContext context, BoxConstraints constraints) {
        if (constraints.maxWidth > 600) {
          return _buildWideContainers();
        } else {
          return _buildNormalContainer();
        }
      },
    ),
  );
}
```

https://api.flutter.dev/flutter/widgets/LayoutBuilder-class.html

LayoutBuilder

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: LayoutBuilder(
      builder: (BuildContext context, BoxConstraints constraints) {
        if (constraints.maxWidth > 600) {
          return _buildWideContainers();
        } else {
          return _buildNormalContainer();
        }
      },
    ),
  );
}
```

https://api.flutter.dev/flutter/widgets/LayoutBuilder-class.html

FutureBuilder

Resolves `Future<T>` into UI

Uses `AsyncSnapshot<T>`

> `connectionState` (none, waiting, active, done)

> data

> error

> stacktrace

https://api.flutter.dev/flutter/widgets/FutureBuilder-class.html

StreamBuilder

Resolves `Stream<T>` into UI

Uses `AsyncSnapshot<T>`

> `connectionState` (none, waiting, active, done)

> data

> error

> stacktrace

https://api.flutter.dev/flutter/widgets/StreamBuilder-class.html

Yandex

Thank you for attention

Sergey Koltsov, Yandex.Pro Team Lead

ringov@yandex-team.ru