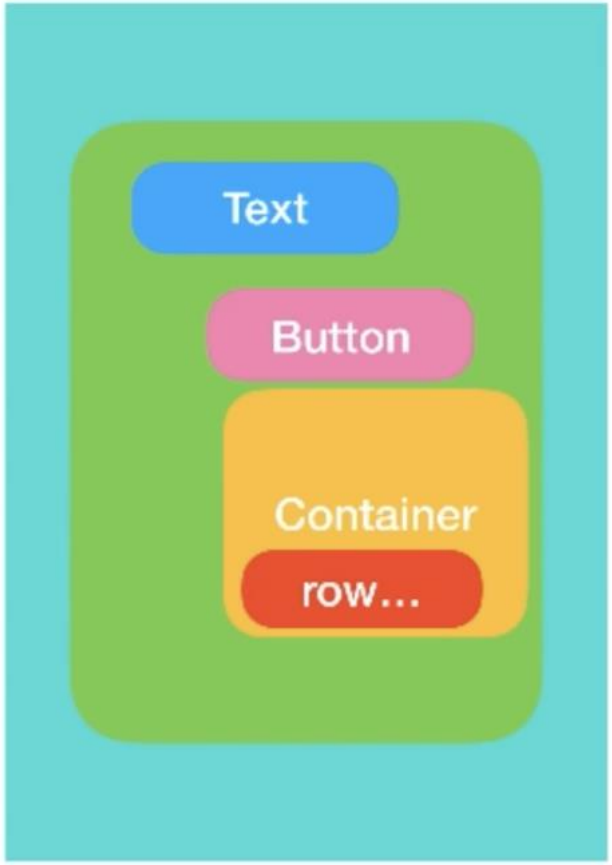
# Stream

```dart
Stream<int> CountStream() async* {
  for (int i = 1; i <= 10; i++) {
    print("SENT boat no. " + i.toString());
    await Future.delayed(Duration(seconds: 2));
    yield i;
  }
}
```

```dart
void main(List<String> args) async {
  Stream<int> stream = CountStream();

  stream.listen((receivedData) {
    print("RECEIVED boat no. " + receivedData.toString());
  });
}
```

# Flutter – introduction au Framework

## Anatomie d'une application :Tout est widget

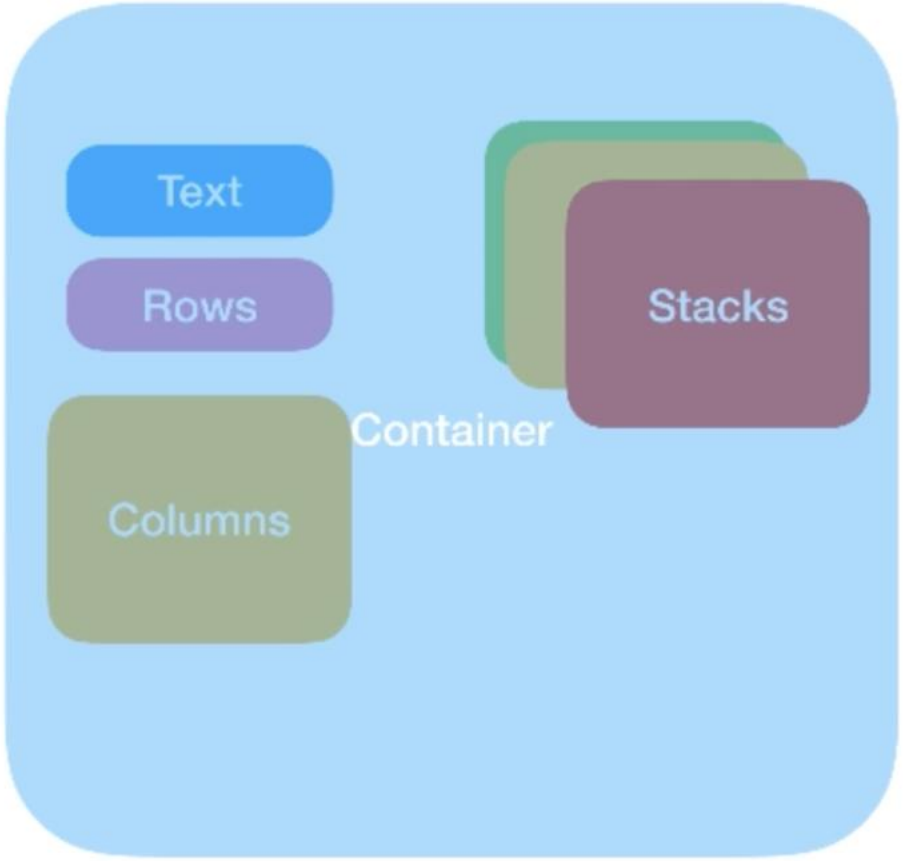


# Flutter – introduction au Framework

## Anatomie d'une application :Tout est widget

# Flutter – introduction au Framework

## Anatomie d'une application :Tout est widget



# Flutter – introduction au Framework

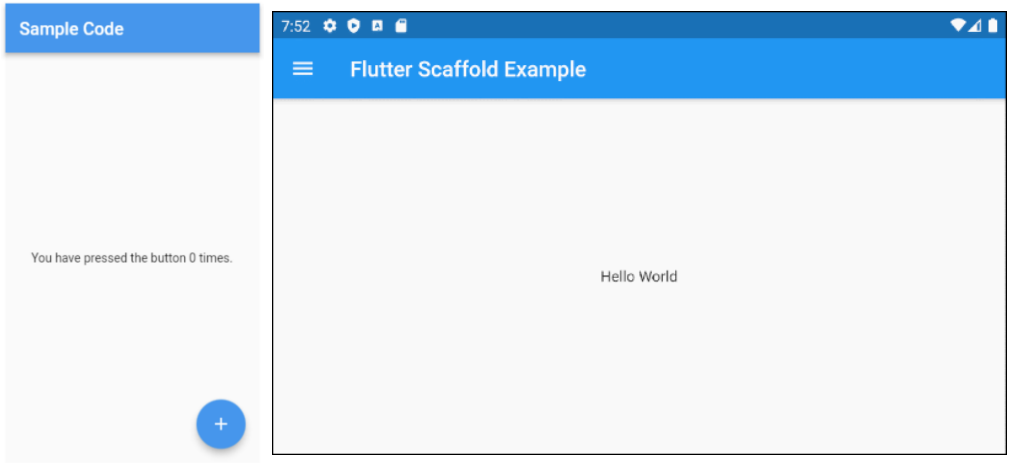
## Deux types de Widget : Statless et statfull

1. Stateless Widget
    1. Ne dépend pas d'autres choses que des ses propres informations qui lui sont fournies lors de son build(par son parent)
    2. Aucun événement utilisateur ne relancera le build d'un stateless widget

2. Statefull Widget
    1. Possède un état représenté par ses « données internes ».
    2. Cet état changera  au cours de cycle de vie du widget en question
    3. Les données incluses dans ce type de widget forme un ensemble que l'on nome state
    4. Quand les données du state changent, le rendu du widget est régénéré


# Flutter – introduction au Framework

## Quelques Widgets prêts à être utilisés : Scaffold




```
Scaffold (
  appBar: AppBar(…
  ),
  body: Center(…
  ),
  drawer: Drawer(…
  ),
);
```

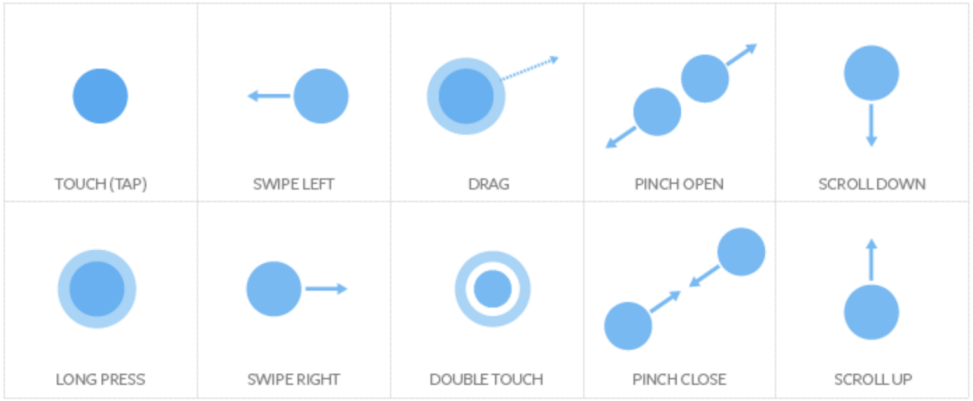
```
appBar: AppBar(
  title: Text(this.title),
),
```

```
body: Center(
    child:
    Text(
      'Hello World',
    )
),
```

```
drawer: Drawer(
  child: ListView(
    children: const <Widget> [
      DrawerHeader(
        decoration: BoxDecoration(
          color: Colors.green,
        ),
        child: Text(
          'Hello World',
          style: TextStyle(
            color: Colors.green,
            fontSize: 24,
          ),
        ),
      ),
      ListTile(
        title: Text('Gallery'),
      ),
      ListTile(
        title: Text('Slideshow'),
      ),
    ],
  ),
```

# Flutter – introduction au Framework

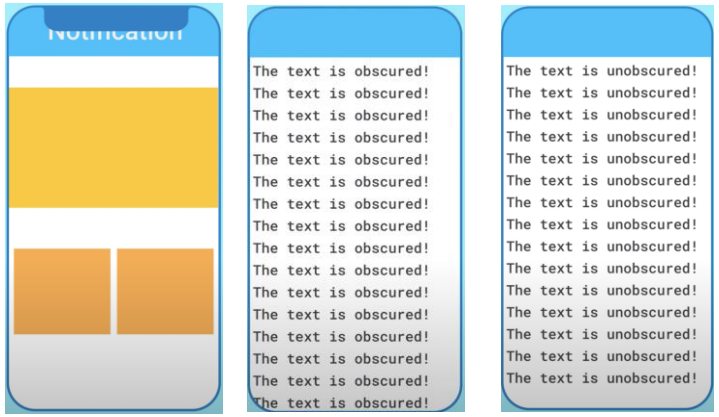
## Quelques Widgets prêts à être utilisés : GestureDetector




```
GestureDetector(
    onTap: () {
      print('onTap');
      Feedback.forTap(context);
    },
    onLongPress: () {
      print('onLongPress');
      Feedback.forLongPress(context);
    },
    child: RaisedButton(
      child: Text('Click'),
    ),
  )
```

# Flutter – introduction au Framework

## Quelques Widgets prêts à être utilisés : SafeArea




```
class SafeArea extends StatelessWidget {
  /// Creates a widget that avoids
     operating system interfaces.
```

```
@override
Widget build(BuildContext context) {
    assert(debugCheckHasMediaQuery(context));
    final EdgeInsets padding = MediaQuery.of(context)
    return new Padding(
      padding: new EdgeInsets.only(
        left: math.max(left ? padding.left : 0.0, min
        top: math.max(top ? padding.top : 0.0, minimu
        right: math.max(right ? padding.right : 0.0,
        bottom: math.max(bottom ? padding.bottom : 0.
      ),
      child: new MediaQuery.removePadding(
        context: contex t,
        removeLeft: left,
```

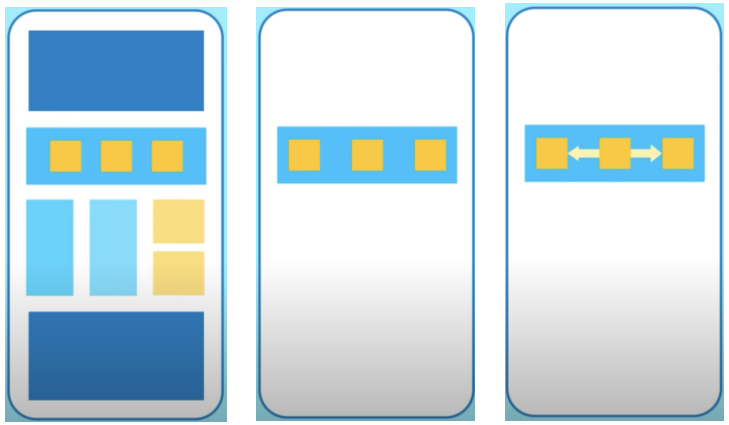
```
ListView(
   children: List.generate(
     100,
     (i) => Text('This is some text'),
   ),
)
```

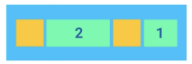
```
SafeArea(
 child: ListView(),
 top: true,
 bottom: true,
 left: false,
 right: true,
)
```

```
@override
Widget build(BuildContext context) {
 return Scaffold(
   body: SafeArea(
     child: TonsOfOtherWidgets(),
   ),
 );
}
```

# Flutter – introduction au Framework

## Quelques Widgets prêts à être utilisés : Expanded

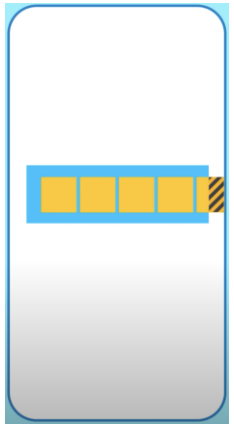


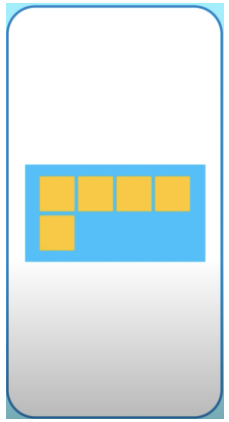


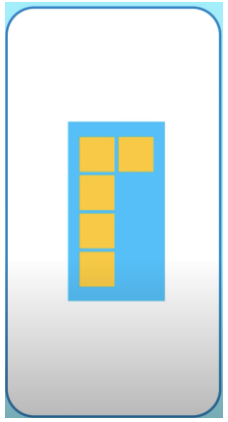

```
Row(
  children: [
    MyWidget(),
    Expanded(
      child: MyWidget()
    ),
    MyWidget(),
  ],
)
```

```
Expanded(
  flex: 2,
  child: Container()
),
```

# Quelques Widgets prêts à être utilisés : Wrap







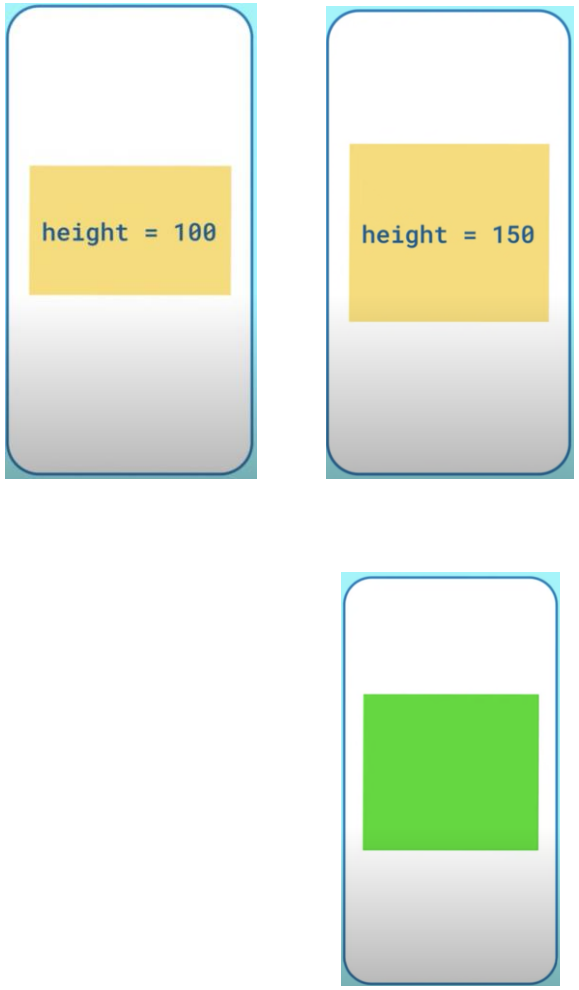
```
Wrap(
  children: [
    MyWidget(),
    MyWidget(),
    MyWidget(),
  ],
)
```

```
Wrap(
  direction: Axis.vertical,
  children: [
    MyWidget(),
    MyWidget(),
    MyWidget(),
    MyWidget(),
    MyWidget(),
  ],
)
```

```
Wrap(
  alignment: WrapAlignment.end,
  spacing: 10.0,
  runSpacing: 20.0,
  children: [
    MyWidget(),
    MyWidget(),
    MyWidget(),
    MyWidget(),
    MyWidget(),
  ],
)
```

# Flutter – introduction au Framework

## Quelques Widgets prêts à être utilisés : AnimatedContainer



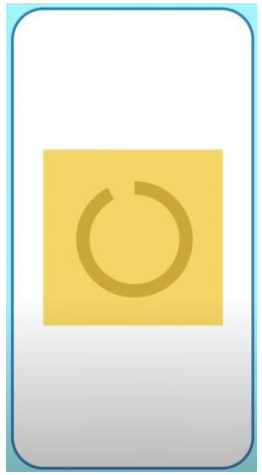

```
@override
Widget build(BuildContext context) {
 return AnimatedContainer(
     color: _color, // (0xFF00BB00)
     duration: _myDuration,
     child: SomeOtherWidget(),
   ),
 );
}
```

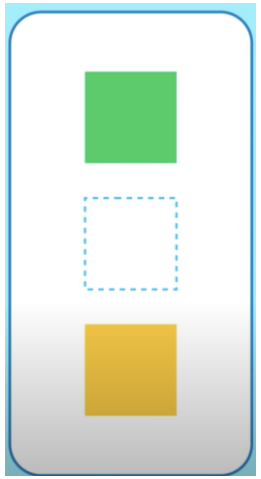
```
@override
Widget build(BuildContext context) {
 return AnimatedContainer(
     color: _color, // (0xFF00BB00)
     duration: _myDuration,
     child: SomeOtherWidget(),
   ),
 );
}
```

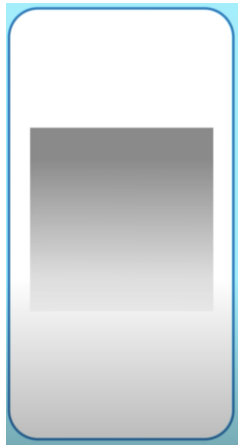
```
setState(() {
  _color = Color(0xFF0000FF);
})
```

# Flutter – introduction au Framework

## Quelques Widgets prêts à être utilisés : Opacity







```
class SomeWidget extends StatelessWidget {
 @override
 Widget build(BuildContext context) {
   final widgets = [
     MyWidget(Colors.green),
     MyWidget(Colors.blue),
     MyWidget(Colors.yellow),
   ];

   return Column(
     children: widgets,
   );
 }
}
```

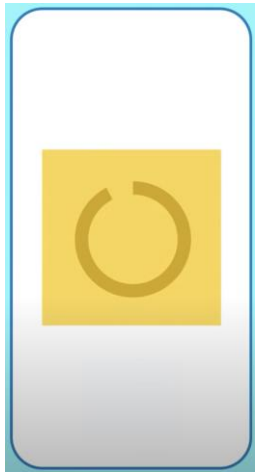
```
final widgets = [
 MyWidget(Colors.green),
Opacity(
   opacity:0.0,
   child:MyWidget(Colors.blue),
 ),
 MyWidget(Colors.yellow),
];
```

```
Stack(
 children: [
   MyImageWidget(),
   Opacity(
     opacity: 0.25,
     child:MyGradientWidget(),
   )
 ],
)
```

```
Stack(
 children: [
   MyImageWidget(),
   AnimatedOpacity(
     duration: _myDuration,
     opacity: _myOpacity,
     child: MyGradientWidget(),
   ),
 ],
)
```

```
setState(() =>
 _myOpacity = 0.0);
```

# Quelques Widgets prêts à être utilisés : FutureBuilder



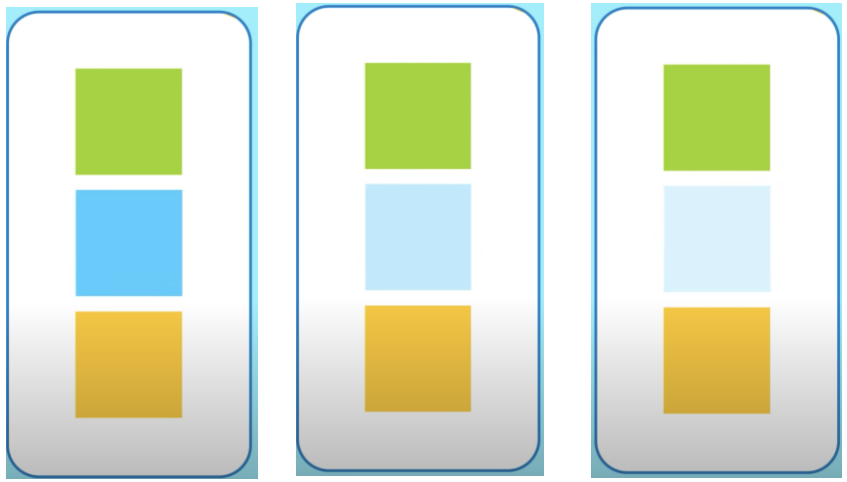
```
FutureBuilder(
  future: http.get('http://awesome.data'),
  builder: _myBuilderFunction,
)
```

```
FutureBuilder(
  future: http.get('http://awesome.data'),
    builder: (context, snapshot) {
    if (snapshot.connectionState ==
        ConnectionState.done) {
    return AwesomeData(snapshot.data);
  } else {
    return CircularProgressIndicator();
  }
}
```

```
if (snapshot.connectionState ==
    ConnectionState.done) {
  if (snapshot.hasError) {
    return SomethingWentWrong();
  }
  …
}
```

```
ConnectionState.none;
ConnectionState.waiting;
ConnectionState.active;
ConnectionState.done;
```

# Quelques Widgets prêts à être utilisés : FadeTransition



```
FadeTransition(
  opacity: animation,
  child: Text(widget.text));
```

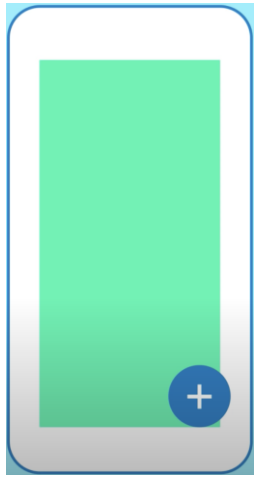
```
final controller = AnimationController(
  vsync: this,
  duration: Duration(seconds: 2),
);
```

```
controller.forward();
```

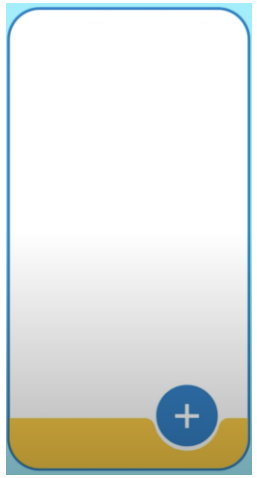
```
Final animation = Tween(
  begin: 0.0,
  end: 1.0,
).animate(controller);
```

```
class MyFadeIn extends StatefulWidget {
  final Widget child;

  MyFadeIn({@required this.child});

  @override
  createState() => _MyFadeInState();
}

class _MyFadeInState extends State<MyFadeIn>
    with SingleTickerProviderStateMixin {
  AnimationController _controller;
  Animation _animation;

  @override
  void initState() {
    _controller = AnimationController(
      vsync: this,
      duration: Duration(seconds: 2),
    );
@override
void initState() {
  _controller = AnimationController(
    vsync: this,
    duration: Duration(seconds: 2),
  );
  _animation = Tween(
    begin: 0.0,
    end: 1.0,
  ).animate(_controller);
}

@override
dispose() {
  _controller.dispose();
  super.dispose();
}
   @override
   Widget build(BuildContext context) {
     _controller.forward();
     return FadeTransition(
       opacity: _animation,
       child: widget.child,
     );
   }
}
```

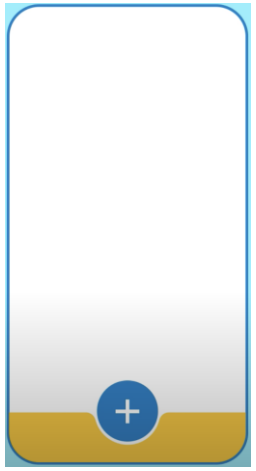
# Quelques Widgets prêts à être utilisés : FloatingActionButton



```
Scaffold(
  floatingActionButton: FloatingActionButton(
    child: Icon(Icons.add),
    onPressed: () {},
  ),
);
```

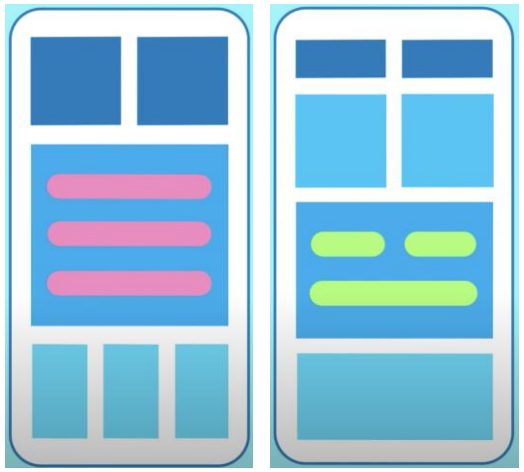


```
Scaffold(
  floatingActionButton: ...
  bottomNavigationBar: BottomAppBar
    color: Colors.yellow,
    child: Container(height: 50.0),
  ),
);
```



```
Scaffold(
  floatingActionButton: ...
  bottomNavigationBar: ...
  floatingActionButtonLocation:
    FloatingActionButtonLocation.endDocked,
);
```

# Quelques Widgets prêts à être utilisés : PageView



```
final controller = PageController(
  initialPage: 1,
);
```

```
Final pageView = PageView(
  controller: controller,
  children: [
    MyPage1Widget(),
    MyPage2Widget(),
    MyPage3Widget(),
  ],
);
```

```
Final pageView = PageView(
    controller: controller,
    scrollDirection: Axis.vertical,
    children: [
      Page1(),
      Page2(),
      Page3(),
    ],
);
```

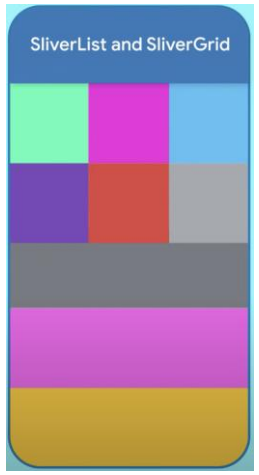
# Quelques Widgets prêts à être utilisés : SilverAppBar


```
CustomScrollView(
    slivers: <Widget>[
      SliverAppBar(
        title: Text('SliverAppBar'),
      ),
      _oneSliver,
      _anotherSliver,
      _yetAnotherSliver
    ],
  );
```

```
SliverAppBar(
        expandedHeight: 200.0,
        flexibleSpace: FlexibleSpaceBar(
          background: _expandedImage,
        ),
      ),

    slivers: <Widget>[
```

```
SliverAppBar(floating: true)
```

# Quelques Widgets prêts à être utilisés : SilverList & SilverGrid




```
SliverList(
  delegate: SliverChildListDelegate(
    [
      widget,
      anotherWidget,
      yetAnotherWidget,
    ],
  ),
```

```
SliverList(
  delegate: SliverChildBuilderDelegate(
      (BuildContext context, int index) {
    return aWidget;
  }),
);
```

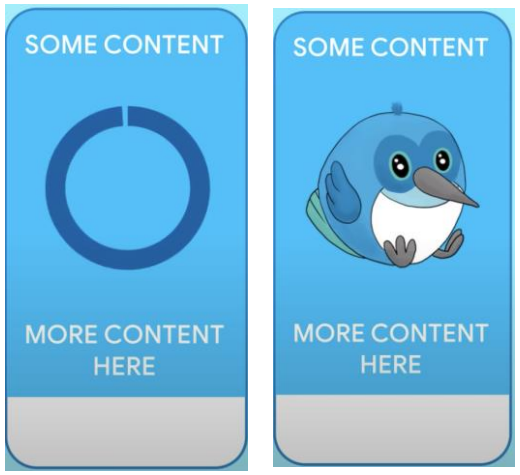
```
SliverGrid.count(
  children: scrollItems,
  ...
);
```

```
SliverGrid.count(
  children: scrollItems,
  crossAxisCount: 4,
);
```

```
SliverGrid.extent(
  crossAxisExtent: 90.0,
    ...
);
```

# Flutter – introduction au Framework

## Quelques Widgets prêts à être utilisés : fadeInImage




```
FadeInImage.assetNetwork(
    placeholder: 'assets/waiting.png',
    image: 'https://example.com/image.png',
)
```

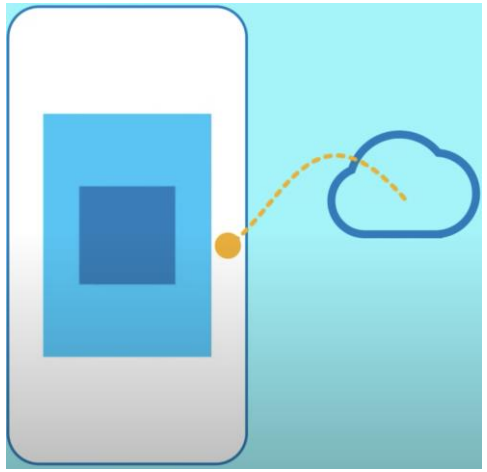
```
FadeInImage.memoryNetwork(
    placeholder: localImageBytes,
    image: 'https://example.com/image.png',
)
```

```
FadeInImage.memoryNetwork(
    height: 300.0,
    placeholder: localImageBytes,
    image: 'https://example.com/image.png',
)
```

```
FadeInImage.assetNetwork(
    fadeInDuration:
      const Duration(seconds: 1),
    placeholder: 'waiting.png',
    image: 'loaded.png',
)
```

```
FadeInImage.assetNetwork(
    fadeInCurve: Curves.bounceIn,
    placeholder: 'waiting.png',
    image: 'loaded.png',
)
```

# Quelques Widgets prêts à être utilisés : StreamBuilder



```
Stream<int> count() async* {
  int i = 1;
  while (true) {
    yield i++;
  }
}
```

```
StreamBuilder(
  stream: _myStream,
  builder: (context, snapshot) {
    return MyWidget(snapshot.data);
  },
);
```

```
StreamBuilder(
  stream: _myStream,
  initialData: 42,
  builder: (context, snapshot) {
    return MyWidget(snapshot.data);
  },
);
```

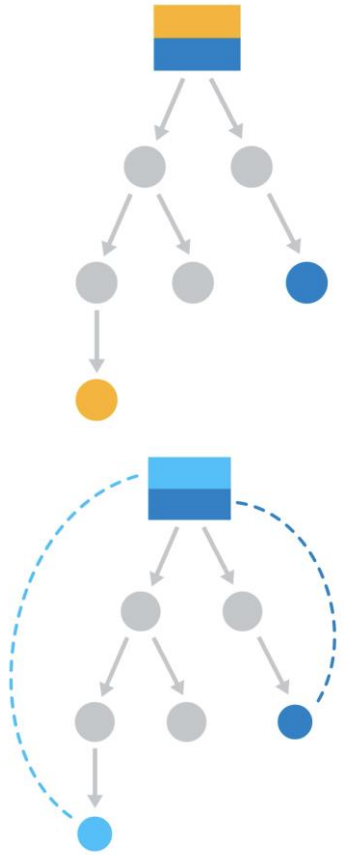
```
StreamBuilder(
  stream: _myStream,
  builder: (context, snapshot) {
    if (!snapshot.hasData) {
      return CircularProgressIndicator();
    }

    return MyWidget(snapshot.data);
  },
);
```

```
StreamBuilder(
  stream: _myStream,
  builder: (context, snapshot) {
    switch (snapshot.connectionState) {
      case ConnectionState.waiting:
      case ConnectionState.none:
        return LinearProgressIndicator();
      case ConnectionState.active:
        return MyWidget(snapshot.data);
      case ConnectionState.done:
        return MyFinalWidget(snapshot.data);
    }
  },
);
```

```
StreamBuilder(
  stream: _myStream,
  builder: (context, snapshot) {
    if (snapshot.hasError) {
      return UhOh(snapshot.error);
    }
    ...
  },
);
```

# Flutter – introduction au Framework

## Quelques Widgets prêts à être utilisés : InheritedModel



```
class MyAncestor extends InheritedWidget {
  const MyAncestor(
    this.colorOne,
    this.colorTwo,
    Widget child,
  ) : super(child: child);

  final Color colorOne;
  final Color colorTwo;

  @override
  bool updateShouldNotify(
        MyAncestor oldWidget) {
    return colorOne != oldWidget.colorOne ||
          colorTwo != oldWidget.colorTwo;
  }
}
```

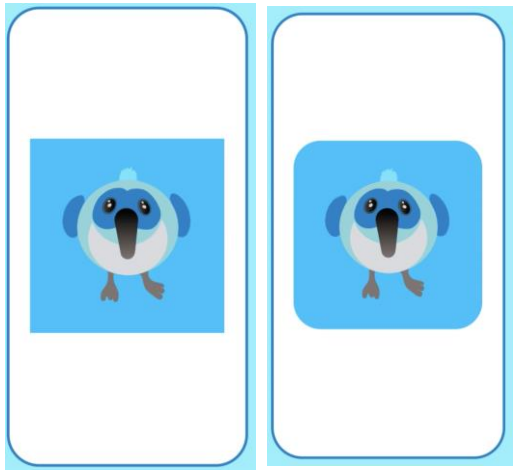
```
class ColorTwoWidget extends StatelessWidget
  @override
  Widget build(BuildContext context) {
    final ancestor = context
      .inheritFromWidgetOfExactType(
        MyAncestor);

    return Container(
      color: ancestor.colorTwo;
      height: 50.0,
      width: 50.0,
    );
  }
```

```
class MyAncestor
    extends InheritModel<String> {
  const MyAncestor(
    this.colorOne,
    this.colorTwo,
    Widget child,
  ) : super(child: child);

  final Color colorOne;
  final Color colorTwo;

  @override
  bool updateShouldNotify(MyAncestor oldWidg
    return colorOne != oldWidget.colorOne ||
        colorTwo != oldWidget.colorTwo;
  }
```

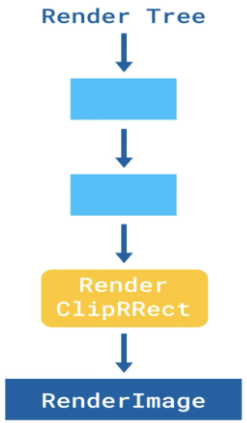
```
class ColorOneWidget extends StatelessWidget
  @override
  Widget build(BuildContext context) {
    final ancestor = InheritedModel
        .inheritFrom<MyAncestor>(
          context,
          aspect: 'one',
        );

    return Container(
      color: ancestor.colorOne;
      height: 50.0,
      width: 50.0,
    );
  }
}
```

```
  @override
  bool updateShouldNotifyDependent(
      MyAncestor oldWidget,
      Set<String> dependencies) {
    if (dependencies.contains('one') &&
        colorOne != oldWidget.colorOne) {
      return true;
    }
    if (dependencies.contains('two') &&
        colorTwo != oldWidget.colorTwo) {
      return true;
    }
```

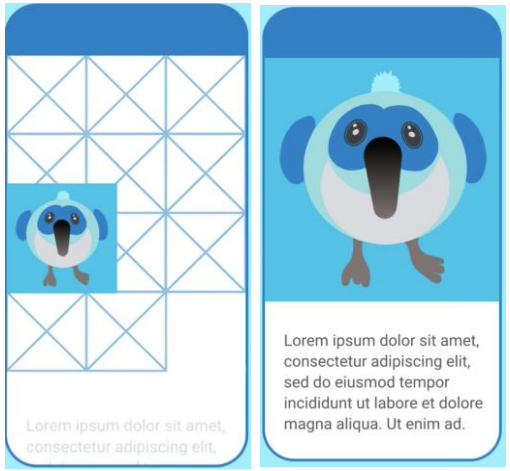
# Quelques Widgets prêts à être utilisés : clipRRect



```
ClipRRect(
  borderRadius:
      BorderRadius.circular(15.0),
  child: MyDashPicWidget(),
);
```



# Quelques Widgets prêts à être utilisés : Hero




```
class MyHomePage extends StatelessWidget {
  @override
  Widget build(context) {
    ...
    Hero(
      tag: 'dash',
      child: Image.asset('images/dash.jpg'),
    )
    ...
  }
}
```

```
class MyDetailPage extends StatelessWidget {
  @override
  Widget build(context) {
    ...
    Image.asset('images/dash.jpg')
    ...
  }
}
```

# Flutter – introduction au Framework

## Quelques Widgets prêts à être utilisés : customPaint

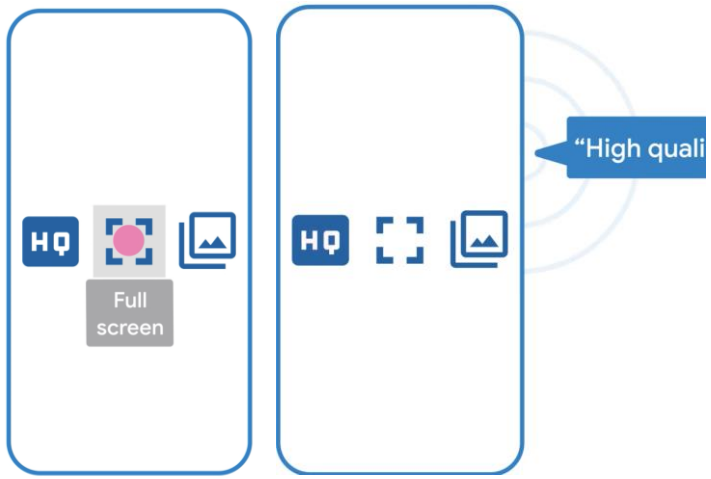
```
class MyPainter extends CustomPainter {
  @override
  void paint(Canvas canvas, Size size) {
    // ...
  }

  @override
  bool shouldRepaint(CustomPainter old) {
    // ...
  }
}
```

```
@override
void paint(Canvas canvas, Size size) {
  canvas.drawLine(...);
}
```

```
@override
bool shouldRepaint(CustomPainter old) {
  return old.myParameter != myParameter;
}
```

```
canvas.drawLine()
canvas.drawRect()
canvas.drawCircle()
canvas.drawArc()
canvas.drawPath()
canvas.drawImage()
canvas.drawImageNine()
canvas.drawParagraph()
```

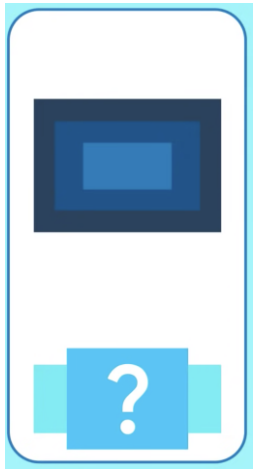
# Quelques Widgets prêts à être utilisés : tooltip




```
Tooltip(
  message: 'Dash',
  verticalOffset: 48,
  height: 24,
  child: MyVisualWidget(),
)
```

```
IconButton(
  icon: Icon(Icons.high_quality),
  tooltip: 'High quality',
)
```

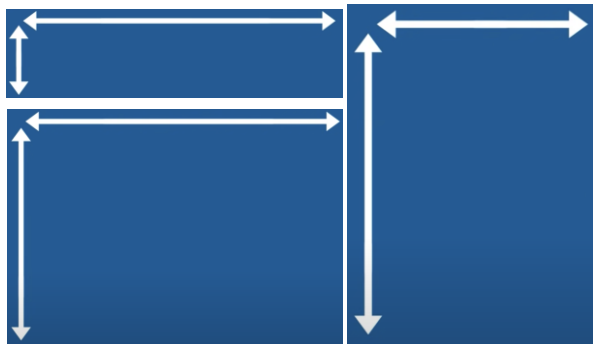
# Quelques Widgets prêts à être utilisés : FittedBox




```
MyBlueRect(
  child: FittedBox(
    alignment: Alignment.centerLeft,
    fit: BoxFit.contain,
    child: MyCatPic(),
  ),
)
```

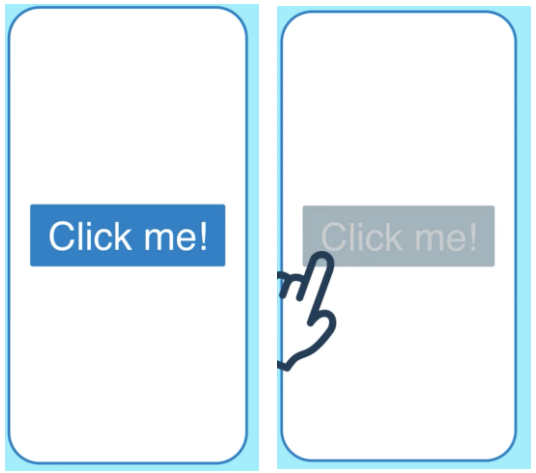
# Flutter – introduction au Framework

## Quelques Widgets prêts à être utilisés : LayoutBuilder



```
Widget build(BuildContext context) {
  return LayoutBuilder(
    builder: (context, constraints) {
      if (constraints.maxWidth < 600) {
        return MyOneColumnLayout();
      } else {
        return MyTwoColumnLayout();
      }
    },
  );
}
```

# Quelques Widgets prêts à être utilisés : AbsorbPointer



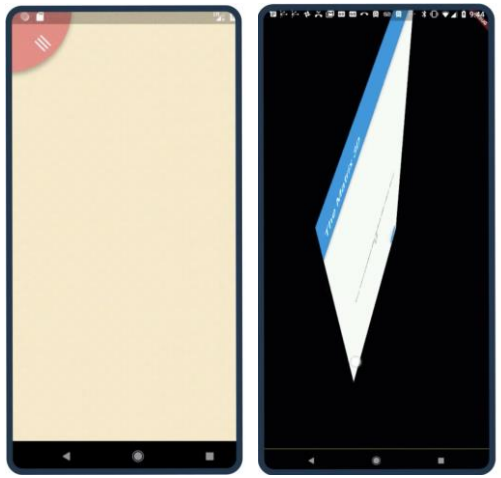

```
class MyHomeScreen extends
    StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return AbsorbPointer(
      child: ABunchOfWidgets(),
    );
  }
}
```

```
Widget build(BuildContext context) {
  return AbsorbPointer(
    absorbing: false,
    child: ABunchOfWidgets(),
  );
}
```

```
Widget build(BuildContext context) {
  return AbsorbPointer(
    ignoringSemantics: false,
    child: ABunchOfWidgets(),
  );
}
```

# Flutter – introduction au Framework

## Quelques Widgets prêts à être utilisés : Transform



```
Transform.rotate(
   angle: pi/4, // 45 deg
   child: MyIcon(),
);
```

```
Transform.translate(
   offset: Offset(50, 50),
   child: MyIcon(),
);
```

```
Transform.scale(
   scale: 1.5,
   child: MyIcon(),
);
```

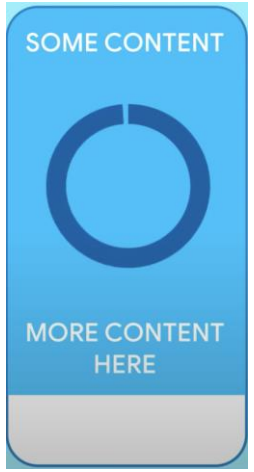
---

```
Transform(
 transform: Matrix4.skewX(0.3),
 child: MyIcon(),
);
```

# Quelques Widgets prêts à être utilisés : BackDropFilter

# Quelques Widgets prêts à être utilisés : Align

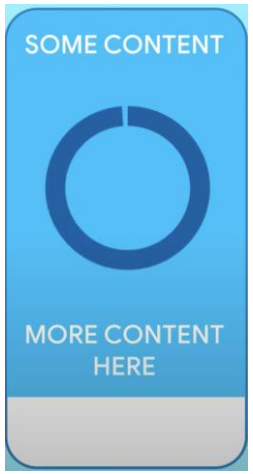
# Quelques Widgets prêts à être utilisés : positioned



# Quelques Widgets prêts à être utilisés : AnimatedBuilder

# Quelques Widgets prêts à être utilisés : Dismissible

# Quelques Widgets prêts à être utilisés : SizedBox



# Flutter – Architecture d'une application

```dart
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
@override
  Widget build(BuildContext context) {
    return (MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: Text('First
App'),backgroundColor: Colors.orange,),
        body: Center(
            child: Text(
          'Hello',style: TextStyle(fontSize: 30),
          textAlign: TextAlign.center,
        )),
      ),
    ));
  }
}
```