# THE ULTIMATE FLUTTER CHEAT SHEET

## INPUTS & CONTROLS

`TextField` displays a keyboard input field

```
TextField(
  controller: TextEditingController,
  keyboardType: TextInputType.number,
  decoration: InputDecoration,
  onChanged: (newVal) => {},
)
```

`FlatButton` shows you a button that you can press

```
FlatButton(
  child: Widget, // Usually Text or Container
  onPressed: () => {},
  textColor: Colors.red,
)
```

Note: It's besties `OutlineButton` and `RaisedButton` accept similar parameters

Create your own dropdown using `DropDownButton`:

```
DropDownButton<String>(
  value: curValue, // selected option
  icon: Icon(Icons.arrow_drop_down),
  onChanged: (newVal) => {},
  items: [
    DropDownMenuItem(
      value: "opt1", child: Text("Option 1")
    ),
    ...
  ]
)
```

## RESPONSIVE DESIGN

Use `MediaQuery` to determine what device size you're using; adjust your output accordingly.

```
final width = MediaQuery.of(context).width;
if (width <= 600) {
  // mobile
} else if (width <= 960) {
  // tablet
} else {
  // desktop
}
```

## PLUGINS

These plugins come highly recommended. Try 'em!

| | | |
|---|---|---|
| flutter_svg | get | animations |
| provider | flutter_hooks | google_fonts |
| wiredash | video_player | dio |
| sentry | url_launcher | image_picker |
| http | shared_preferences | hive |

## IF YOU LIKED THIS..

Then check out my **Flutter Tricks email series** at yaz.in/flutter

## INTERACTIONS

Use `AlertDialog` to display alerts to the user

```
AlertDialog(
  title: Text,
  content: Text, // shown in the Alert body
  actions: <Widget>[], // usually FlatButton's
)
// trigger it..
showDialog(
  context: context,
  builder: (_) => AlertDialog(..),
  barrierDismissible: bool, // hide on background tap
);
```

Use a `SnackBar` to display short-lived messages

```
Scaffold.of(context).showSnackBar(
  SnackBar(
    content: Text('Greetings'),
    action: SnackBarAction
  )
);
```

To respond to user interactions, try a `GestureDetector`:

```
GestureDetector(
  onTap: () => childTapped(),
  child: Widget,
)
```

Finally, to navigate between screens use `Navigator`:

```
Navigator.of(context).push(
  MaterialPageRoute(
    builder: (_) => NextScreen // Widget
  )
)
```

Note: Go back to the previous screen using `.pop()`

## DART

Filter lists using `.where`, and don't forget `.toList()`

```
fruits = ['apple', 'avocado', 'banana']
fruits.where((f) => f.startsWith('a')).toList()
```

Common ways to iterate over a list:

```
fruits.forEach((f) => eat(f)) // returns null
fruits.map((f) => Text(f)).toList() // returns List
```

And if you need to access the index as well:

```
fruits.asMap().entries.map((f) =>
  Text('${f.key} - ${f.value}')).toList()
```

Use `...` to merge lists

```
Column(
  children: [
    Text('List of Items'),
    ...itemList() // method that returns <Widget>[]
  ])
)
```

..and `reduce` is a great tool to add to your belt:

```
numbers = [1, 2, 3, 5, 7, 9]
numbers.reduce((sum, num) => sum + num, 0)
```

# THE ULTIMATE FLUTTER CHEAT SHEET

## STRUCTURE

Use a `Scaffold` as the base layer for all your screens

```
Scaffold(
  body: Widget,
  appBar: AppBar,
)
```

A `Column` arranges things vertically quite nicely

```
Column(
  children: <Widget>[],
  mainAxisAlignment: MainAxisAlignment.start,
  crossAxisAlignment: CrossAxisAlignment.stretch,
)
```

and a `Row` is great for lining things up horizontally

```
Row(
  children: <Widget>[],
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  crossAxisAlignment: CrossAxisAlignment.center
)
```

If your `Row` or `Column` overflows, try replacing them with a `ListView` (to allow scrolling) or a `Wrap` to wrap to a new line

Use a `ListView` to hold a list of items (in any axis). Similar to Row/Column but also **supports scrolling**.

```
ListView(
  scrollDirection: Axis.horizontal,
  shrinkWrap: bool, // don't fill the main axis
  children: <Widget>[],
  // or lazy-load items using:
  itemCount: 10,
  itemBuilder: (context, index) => Widget,
)
```

Finally, say hello to the most popular widget™, `Container`

```
Container(
  child: Widget,
  alignment: Alignment.topLeft,
  padding: EdgeInsets.all(4),
  decoration: BoxDecoration(), // see below
  width: 150,
  height: 100,
)
```

`BoxDecoration` has some pretty nifty properties:

```
BoxDecoration(
  color: Colors.red, // background
  border: Border.all(width: 2),
  borderRadius: BorderRadius.circular(5),
  boxShadow: <BoxShadow>[],
  gradient: LinearGradient(colors: <Color>[]),
)
```

## IF YOU LIKED THIS..

Then check out my **Flutter Tricks email series** at yaz.in/flutter

## ALIGNMENT & LAYOUT

Order brings joy. We could all use some extra joy.

`Center` does pretty much what you'd expect:

```
Center(child: Widget)
```

`Align` allows you to place things wherever you want:

```
Align(
  child: Widget,
  alignment: Alignment.topLeft,
)
```

Note: To specify position by pixels, use `Positioned`

`Expanded` causes the child to expand along the main axis

```
Expanded(
  child: Widget,
  flex: 1, // how big, relative to adjacent Expanded's
)
```

A `Stack` is perfect for drawing things on top of each other:

```
Stack(
  children: <Widget>[],
  alignment: Alignment.topLeft,
)
```

Finally, `SizedBox` gives you precise control over the space available (often used as a spacer too)

```
SizedBox(
  child: Widget,
  width: 150,
  height: 100,
)
```

## DISPLAY

```
Text("Aloha", style: TextStyle)
```

```
TextStyle(
  fontSize: 18,
  color: Colors.red,
  fontWeight: FontWeight.bold,
  fontFamily: 'Raleway',
  height: 1.5, // 1.5 * fontSize
)
```

Use `RichText` to style words on the same line differently

```
RichText(
  text: TextSpan(children: [
    TextSpan(text: "Hi", style: TextStyle),
    TextSpan(text: "there", style: TextStyle),
  ])
)
```

Use `Icon` to quickly drop some commonly used icons

```
Icon(
  Icons.timer,
  size: 24, // dimensions, in px
  color: Colors.red
)
```

Finally, `Image` to display your own asset

```
Image.asset(
  "./assets/path.png",
  width: 24,
  height: 24,
)
```

# THE ULTIMATE FLUTTER CHEAT SHEET

## INPUTS & CONTROLS

`TextField` displays a keyboard input field

```
TextField(
  controller: TextEditingController,
  keyboardType: TextInputType.number,
  decoration: InputDecoration,
  onChanged: (newVal) => {},
)
```

`FlatButton` shows you a button that you can press

```
FlatButton(
  child: Widget, // Usually Text or Container
  onPressed: () => {},
  textColor: Colors.red,
)
```

Note: It's besties `OutlineButton` and `RaisedButton` accept similar parameters

Create your own dropdown using `DropDownButton`:

```
DropDownButton<String>(
  value: curValue, // selected option
  icon: Icon(Icons.arrow_drop_down),
  onChanged: (newVal) => {},
  items: [
    DropDownMenuItem(
      value: "opt1", child: Text("Option 1")
    ),
    ...
  ]
)
```

## RESPONSIVE DESIGN

Use `MediaQuery` to determine what device size you're using; adjust your output accordingly.

```
final width = MediaQuery.of(context).width;
if (width <= 600) {
  // mobile
} else if (width <= 960) {
  // tablet
} else {
  // desktop
}
```

## PLUGINS

These plugins come highly recommended. Try 'em!

| | | |
|---|---|---|
| flutter_svg | get | animations |
| provider | flutter_hooks | google_fonts |
| wiredash | video_player | dio |
| sentry | url_launcher | image_picker |
| http | shared_preferences | hive |

## IF YOU LIKED THIS..

Then check out my **Flutter Tricks email series** at yaz.in/flutter

## INTERACTIONS

Use `AlertDialog` to display alerts to the user

```
AlertDialog(
  title: Text,
  content: Text, // shown in the Alert body
  actions: <Widget>[], // usually FlatButton's
)
// trigger it..
showDialog(
  context: context,
  builder: (_) => AlertDialog(..),
  barrierDismissible: bool, // hide on background tap
);
```

Use a `SnackBar` to display short-lived messages

```
Scaffold.of(context).showSnackBar(
  SnackBar(
    content: Text('Greetings'),
    action: SnackBarAction
  )
);
```

To respond to user interactions, try a `GestureDetector`:

```
GestureDetector(
  onTap: () => childTapped(),
  child: Widget,
)
```

Finally, to navigate between screens use `Navigator`:

```
Navigator.of(context).push(
  MaterialPageRoute(
    builder: (_) => NextScreen // Widget
  )
)
```

Note: Go back to the previous screen using `.pop()`

## DART

Filter lists using `.where`, and don't forget `.toList()`

```
fruits = ['apple', 'avocado', 'banana']
fruits.where((f) => f.startsWith('a')).toList()
```

Common ways to iterate over a list:

```
fruits.forEach((f) => eat(f)) // returns null
fruits.map((f) => Text(f)).toList() // returns List
```

And if you need to access the index as well:

```
fruits.asMap().entries.map((f) =>
  Text('${f.key} - ${f.value}')).toList()
```

Use `...` to merge lists

```
Column(
  children: [
    Text('List of Items'),
    ...itemList() // method that returns <Widget>[]
  ])
)
```

..and `reduce` is a great tool to add to your belt:

```
numbers = [1, 2, 3, 5, 7, 9]
numbers.reduce((sum, num) => sum + num, 0)
```

# THE ULTIMATE FLUTTER CHEAT SHEET

## STRUCTURE

Use a `Scaffold` as the base layer for all your screens

```
Scaffold(
  body: Widget,
  appBar: AppBar,
)
```

A `Column` arranges things vertically quite nicely

```
Column(
  children: <Widget>[],
  mainAxisAlignment: MainAxisAlignment.start,
  crossAxisAlignment: CrossAxisAlignment.stretch,
)
```

and a `Row` is great for lining things up horizontally

```
Row(
  children: <Widget>[],
  mainAxisAlignment: MainAxisAlignment.spaceBetween,
  crossAxisAlignment: CrossAxisAlignment.center
)
```

If your `Row` or `Column` overflows, try replacing them with a `ListView` (to allow scrolling) or a `Wrap` to wrap to a new line

Use a `ListView` to hold a list of items (in any axis). Similar to Row/Column but also **supports scrolling**.

```
ListView(
  scrollDirection: Axis.horizontal,
  shrinkWrap: bool, // don't fill the main axis
  children: <Widget>[],
  // or lazy-load items using:
  itemCount: 10,
  itemBuilder: (context, index) => Widget,
)
```

Finally, say hello to the most popular widget™, `Container`

```
Container(
  child: Widget,
  alignment: Alignment.topLeft,
  padding: EdgeInsets.all(4),
  decoration: BoxDecoration(), // see below
  width: 150,
  height: 100,
)
```

`BoxDecoration` has some pretty nifty properties:

```
BoxDecoration(
  color: Colors.red, // background
  border: Border.all(width: 2),
  borderRadius: BorderRadius.circular(5),
  boxShadow: <BoxShadow>[],
  gradient: LinearGradient(colors: <Color>[]),
)
```

## IF YOU LIKED THIS..

Then check out my **Flutter Tricks email series** at yaz.in/flutter

## ALIGNMENT & LAYOUT

Order brings joy. We could all use some extra joy.

`Center` does pretty much what you'd expect:

```
Center(child: Widget)
```

`Align` allows you to place things wherever you want:

```
Align(
  child: Widget,
  alignment: Alignment.topLeft,
)
```

Note: To specify position by pixels, use `Positioned`

`Expanded` causes the child to expand along the main axis

```
Expanded(
  child: Widget,
  flex: 1, // how big, relative to adjacent Expanded's
)
```

A `Stack` is perfect for drawing things on top of each other:

```
Stack(
  children: <Widget>[],
  alignment: Alignment.topLeft,
)
```

Finally, `SizedBox` gives you precise control over the space available (often used as a spacer too)

```
SizedBox(
  child: Widget,
  width: 150,
  height: 100,
)
```

## DISPLAY

```
Text("Aloha", style: TextStyle)
```

```
TextStyle(
  fontSize: 18,
  color: Colors.red,
  fontWeight: FontWeight.bold,
  fontFamily: 'Raleway',
  height: 1.5, // 1.5 * fontSize
)
```

Use `RichText` to style words on the same line differently

```
RichText(
  text: TextSpan(children: [
    TextSpan(text: "Hi", style: TextStyle),
    TextSpan(text: "there", style: TextStyle),
  ])
)
```

Use `Icon` to quickly drop some commonly used icons

```
Icon(
  Icons.timer,
  size: 24, // dimensions, in px
  color: Colors.red
)
```

Finally, `Image` to display your own asset

```
Image.asset(
  "./assets/path.png",
  width: 24,
  height: 24,
)
```