Three axis transform and DRY principle

- Explain the DRY principle
- Have heard about “KISS” and “broken windows” in software development
- Know 2 ways how to avoid code duplication in Flutter
- Explain what “flutter clean” does and what is needed afterwards

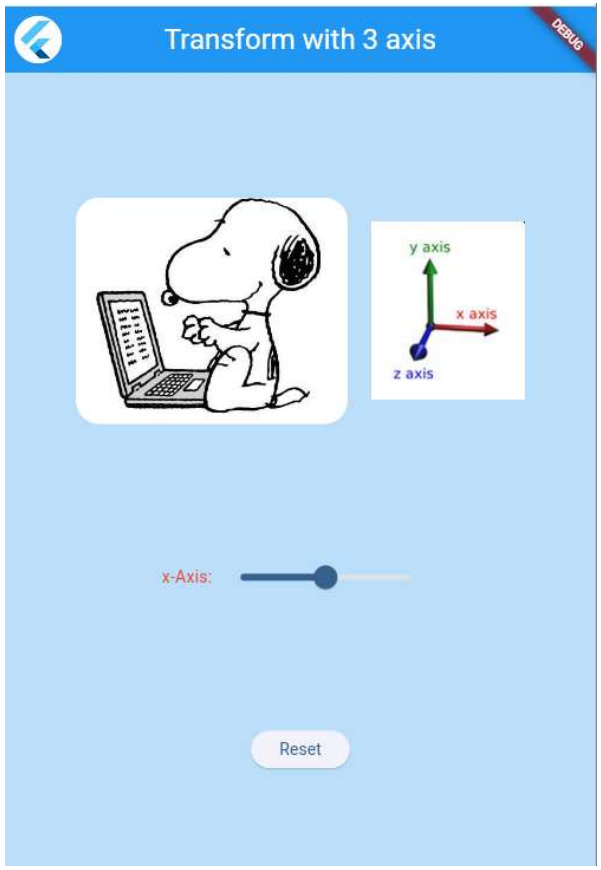
Transform with 3 axis

```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Transform(
      alignment: Alignment.center,
      transform: Matrix4.rotationZ(angleZ),
      child: Transform(
          alignment: Alignment.center,
          transform: Matrix4.rotationY(angleY),
          child: Transform(
            alignment: Alignment.center,
            transform: Matrix4.rotationX(angleX),
            child: ClipRRect(
                borderRadius:
                    const BorderRadius.all(Radius.circular(20)),
                child: Image.asset(
                    "assets/images/snoopy_laptop.jpg",
                    width: 230)), // Image.asset // ClipRRect
          )), // Transform // Transform
    ), // Transform
    const SizedBox(width: 20),
    Image.asset("assets/images/axis.jpg", width: 130),
  ],
), // Row
```

Three nested Transform widgets surrounding the Snoopy image.

First with a slider only for the x-axis

```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    const Text("x-Axis:", style: TextStyle(color: Colors.red)),
    Slider(
      value: angleX,
      min: -2 * pi,
      max: 2 * pi,
      onChanged: (value) {
        setState(() {
          angleX = value;
        });
      },
    ), // Slider
  ],
), // Row
```



Second slider added with copy/paste & adapted

```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    const Text("x-Axis:", style: TextStyle(color: Colors.red)),
    Slider(
      value: angleX,
      min: -2 * pi,
      max: 2 * pi,
      onChanged: (value) {
        setState(() {
          angleX = value;
        });
      },
    ), // Slider
  ],
), // Row
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    const Text("y-Axis:", style: TextStyle(color: Colors.green)),
    Slider(
      value: angleY,
      min: -2 * pi,
      max: 2 * pi,
      onChanged: (value) {
        setState(() {
          angleY = value;
        });
      },
    ), // Slider
  ],
), // Row
```

This code “smells”.

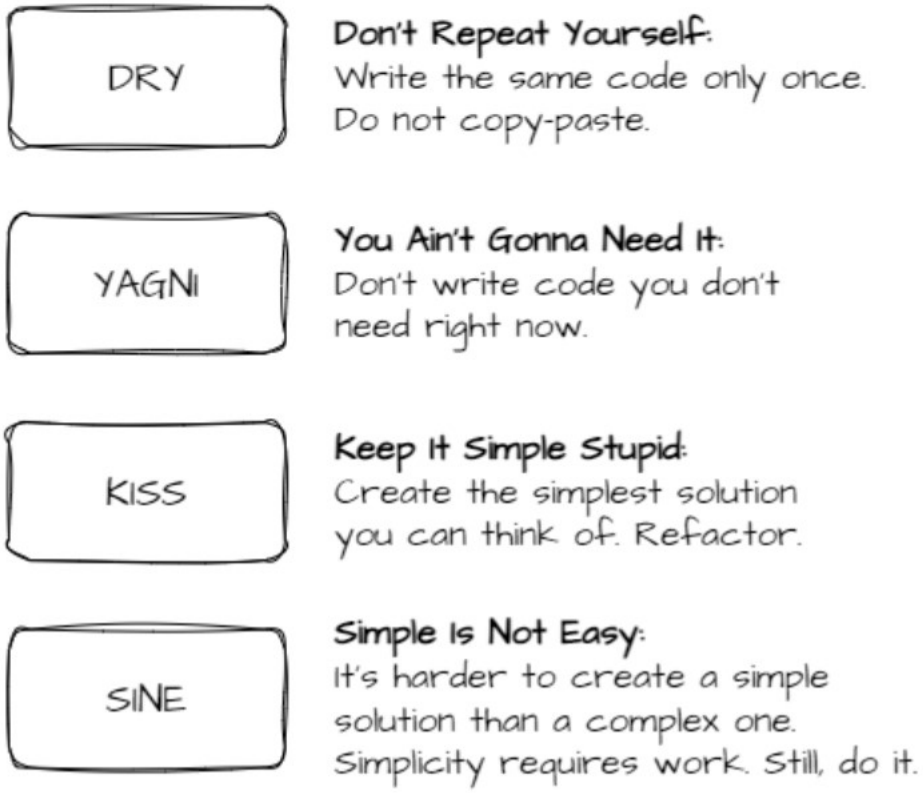
It violates the **DRY principle** (Don’t repeat yourself).

Code-Smell (Code smell)

Unter Code-Smell, kurz Smell oder deutsch übelriechender Code versteht man in der Programmierung ein Konstrukt, das eine Überarbeitung des Programm-Quelltextes nahelegt. Dem Vernehmen nach stammt die Metapher Smell von Kent Beck und erlangte weite Verbreitung durch das Buch Refactoring von Martin Fowler. Wikipedia ›

Copied from https://de.wikipedia.org/wiki/Code-Smell

Some principles in software development



Copied from https://mattilehtinen.com/articles/4-most-important-software-development-principles-dry-yagni-kiss-and-sine/

KISS - Keep it simple, stupid!

Ursprünglich steht die Abkürzung *KISS* für „Keep it simple, stupid!“, übersetzt in etwa „Halte es einfach, Dummkopf!“ bzw. sinngemäß „Mach’s doch so einfach wie möglich.“ Die Interjektion „Stupid!“ ist hier als scherzhafte Anrede zu verstehen, die dem Satz eine flapsige, aber wohlmeinende Bedeutung gibt: „Sei nicht so blöd, dir den Kopf zu zerbrechen, wenn es auch einfach geht.“ Ein ähnliches Beispiel aus der Englischen Sprache ist der Slogan „It’s the economy, stupid“ während der Präsidentschaftswahl in den USA 1992 für die zweite Amtszeit von George Bush senior.

Dennoch wurde vermutlich bereits früh die Anrede gerne umgedeutet und durch andere, weniger beleidigend auffassbare Worte ersetzt, beispielsweise in der Form „**Keep it simple and straightforward**“[*], übersetzt in etwa „Halte es einfach und unkompliziert.“

Copied from https://de.wikipedia.org/wiki/KISS-Prinzip

[*] alternative: „**Keep it short and simple**“

Broken Windows (when principles keep to be violated)

Don’t leave windows broken, don’t leave software issues unaddressed

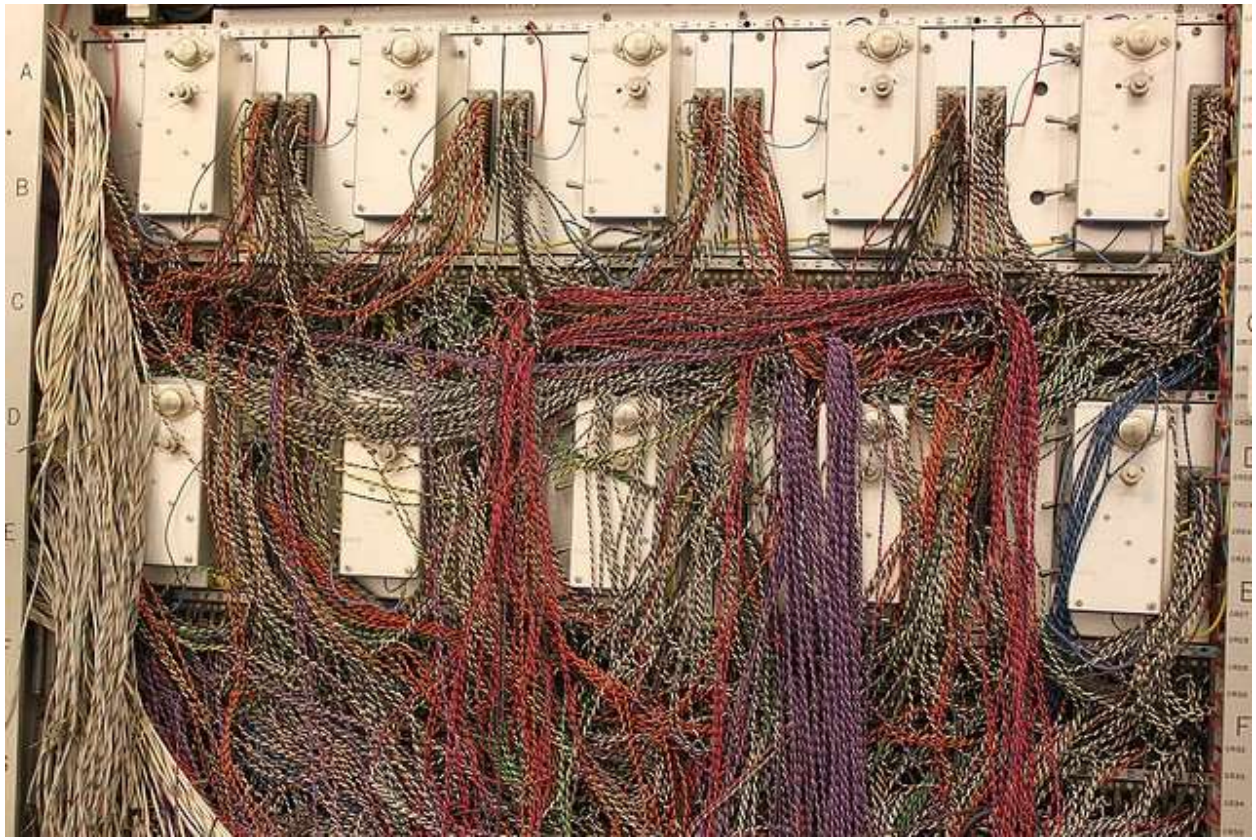
The most crucial lesson from the broken window theory in the context of software development is simple: don’t leave broken windows unrepaired. Whenever you encounter an issue, be it a bad design, incorrect decision, or poor code, address it immediately.

If time constraints prevent a full repair, take action to prevent further damage; board the window up. This could be anything from commenting out the code to throwing an error when it is used. The key is to show that you’re attentive to the situation and committed to maintaining software quality.

Picture copied from https://news.northeastern.edu/2019/05/15/northeastern-university-researchers-find-little-evidence-for-broken-windows-theory-say-neighborhood-disorder-doesnt-cause-crime/

Text copied from https://archerpoint.com/the-broken-windows-theory-the-key-to-tackling-software-entropy/
See also https://www.kungfudev.com/blog/2024/05/18/broken-windows

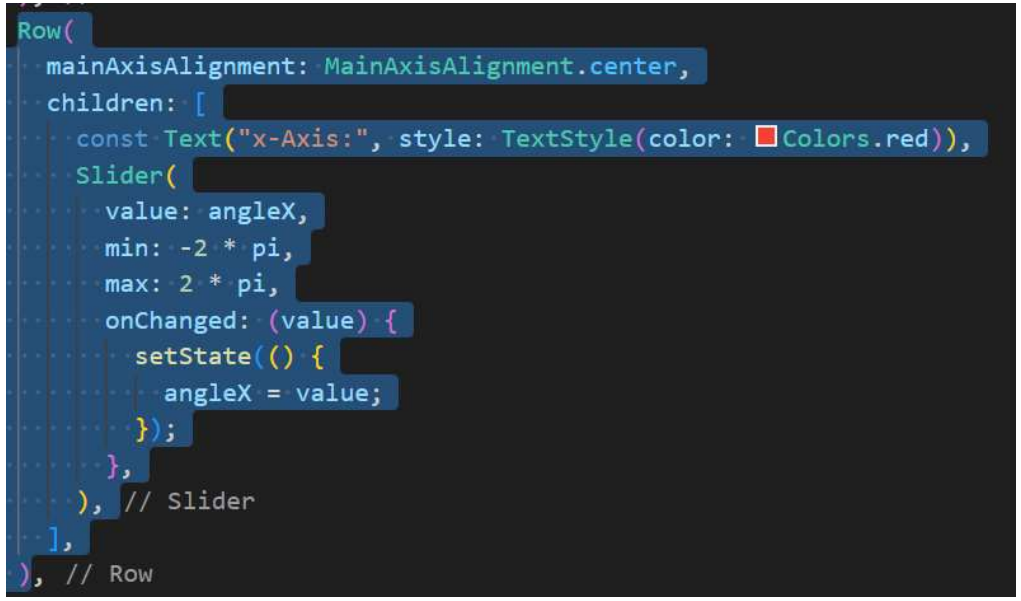
You may end up like this



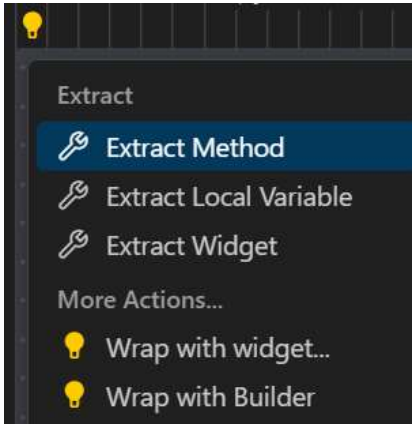
If you know something is broken but never make time to fix it, other bugs and issues will creep into it, until it's a mess that no one dares touch.

Copied from https://www.freecodecamp.org/news/pragmatic-programmer-broken-windows-6916998eecbe/

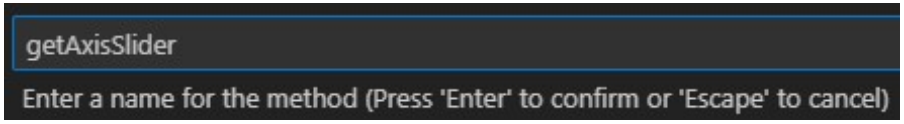
Avoid code duplication by extracting a method



```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    const Text("x-Axis:", style: TextStyle(color: Colors.red)),
    Slider(
      value: angleX,
      min: -2 * pi,
      max: 2 * pi,
      onChanged: (value) {
        setState(() {
          angleX = value;
        });
      },
    ), // Slider
  ],
), // Row
```



Enter the name of the method to be created:

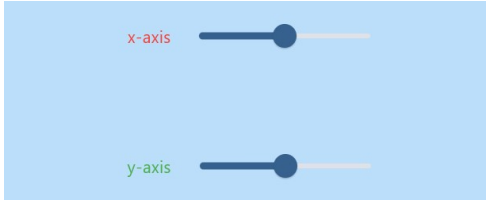


Important: select only the Row widget, that you want to extract, do not include e.g. the following comma in your selection !

Define parameters for your method

```
Row getAxisSlider() {
  return Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      const Text("x-Axis:", style: TextStyle(color: Colors.red)),
      Slider(
        value: angleX,
        min: -2 * pi,
        max: 2 * pi,
        onChanged: (value) {
          setState(() {
            angleX = value;
          });
        },
      ), // Slider
    ],
  ); // Row
}
```

```
Row getAxisSlider(String title, Color color, double angle) {
  return Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text(title, style: TextStyle(color: color)),
      Slider(
        value: angle,
        min: -2 * pi,
        max: 2 * pi,
        onChanged: (value) {
          setState(() {
            angle = value;
          });
        },
      ), // Slider
    ],
  ); // Row
}
```




```
getAxisSlider("x-axis", Colors.red, angleX),
getAxisSlider("y-axis", Colors.green, angleY),
```

Problem: the sliders no longer work

```
getAxisSlider("x-axis", Colors.red, angleX),
getAxisSlider("y-axis", Colors.green, angleY),
```

```
Row getAxisSlider(String title, Color color, double angle) {
  return Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text(title, style: TextStyle(color: color)),
      Slider(
        value: angle,
        min: -2 * pi,
        max: 2 * pi,
        onChanged: (value) {
          setState(() {
            angle = value;
          });
        },
      ), // Slider
    ],
  ); // Row
}
```

This assignment changes the local variable “angle” in method getAxisSlider, but not the values of angleX and angleY in the caller (call-by-value).

Solution A: work with “boxed values”

```
class _MainAppState extends State<MainApp> {
  var boxedAngleX = BoxedValue<double>(0);
  var boxedAngleY = BoxedValue<double>(0);
```

```
class BoxedValue<T> {
  BoxedValue (this.value);
  T value;
}
```

```
child: Transform(
    alignment: Alignment.center,
    transform: Matrix4.rotationY(boxedAngleY.value),
    child: Transform(
      alignment: Alignment.center,
      transform: Matrix4.rotationX(boxedAngleX.value),
      child: ClipRRect(
          borderRadius:
              const BorderRadius.all(Radius.circular(20)),
          child: Image.asset(
              "assets/images/snoopy_laptop.jpg",
              width: 230)), // Image.asset // ClipRRect
    )), // Transform // Transform
```

```
Row getAxisSlider(String title, Color color, BoxedValue<double> boxedAngle) {
  return Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text(title, style: TextStyle(color: color)),
      Slider(
        value: boxedAngle.value,
        min: -2 * pi,
        max: 2 * pi,
        onChanged: (value) {
          setState(() {
            boxedAngle.value = value;
          });
        },
      ), // Slider
    ],
  ); // Row
}
```

```
getAxisSlider("x-axis", Colors.red, boxedAngleX),
getAxisSlider("y-axis", Colors.green, boxedAngleY),
```

Solution B: work with a callback

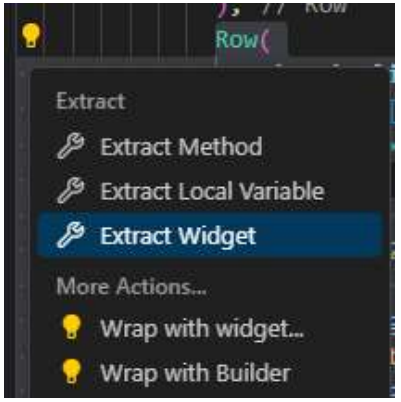
```
class _MainAppState extends State<MainApp> {
  double angleX = 0;
  double angleY = 0;
```

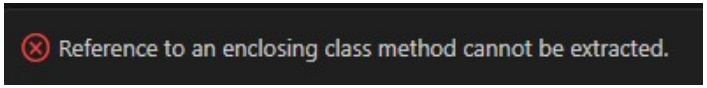
```
getAxisSlider("x-axis", Colors.red, angleX, (value) {
  setState(() {
    angleX = value;
  });
}),
getAxisSlider("y-axis", Colors.green, angleY, (value) {
  setState(() {
    angleY = value;
  });
}),
```

```
Row getAxisSlider(String title, Color color, double angle,
    Function(double value) callback) {
  return Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Text(title, style: TextStyle(color: color)),
      Slider(
        value: angle,
        min: -2 * pi,
        max: 2 * pi,
        onChanged: callback,
      ), // Slider
    ],
  ); // Row
}
```

Solution C: extract a widget

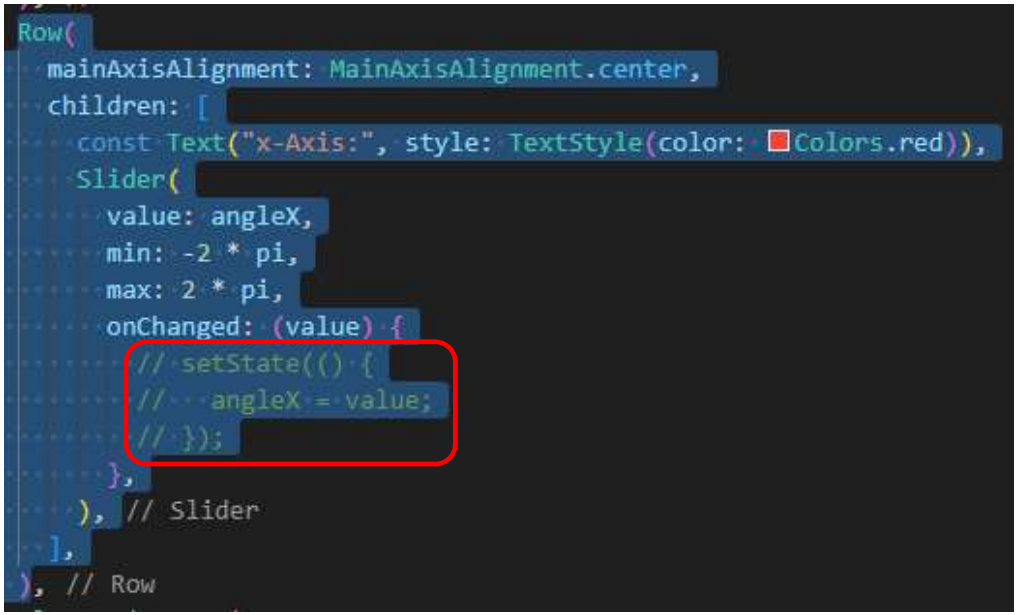
```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    const Text("x-Axis:", style: TextStyle(color: Colors.red)),
    Slider(
      value: angleX,
      min: -2 * pi,
      max: 2 * pi,
      onChanged: (value) {
        setState(() {
          angleX = value;
        });
      },
    ), // Slider
  ],
), // Row
```

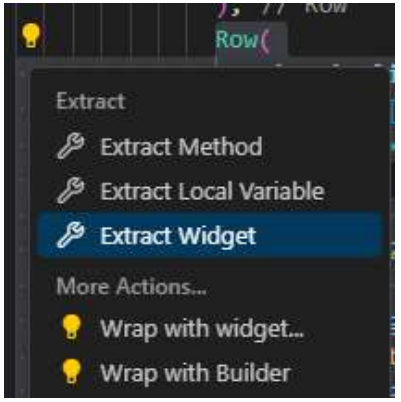




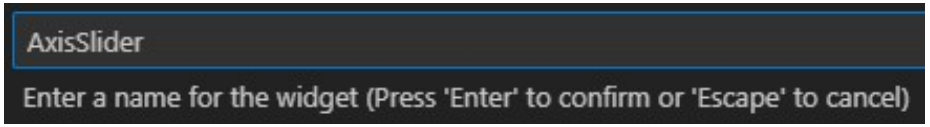
The mentioned “enclosing class method” is this call to “setState”, which is a method of `class _MainAppState extends State<MainApp>`

Workaround: comment out the call to setState





Now “Extract widget” works:



Define parameters for the extracted widget

```dart
class AxisSlider extends StatelessWidget {
  const AxisSlider({
    super.key,
    required this.angleX,
  });

  final double angleX;

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        const Text("x-Axis:", style: TextStyle(color: Colors.red)),
        Slider(
          value: angleX,
          min: -2 * pi,
          max: 2 * pi,
          onChanged: (value) {
            // setState(() {
            //   angleX = value;
            // });
          },
        ), // Slider
      ],
    ); // Row
  }
}
```

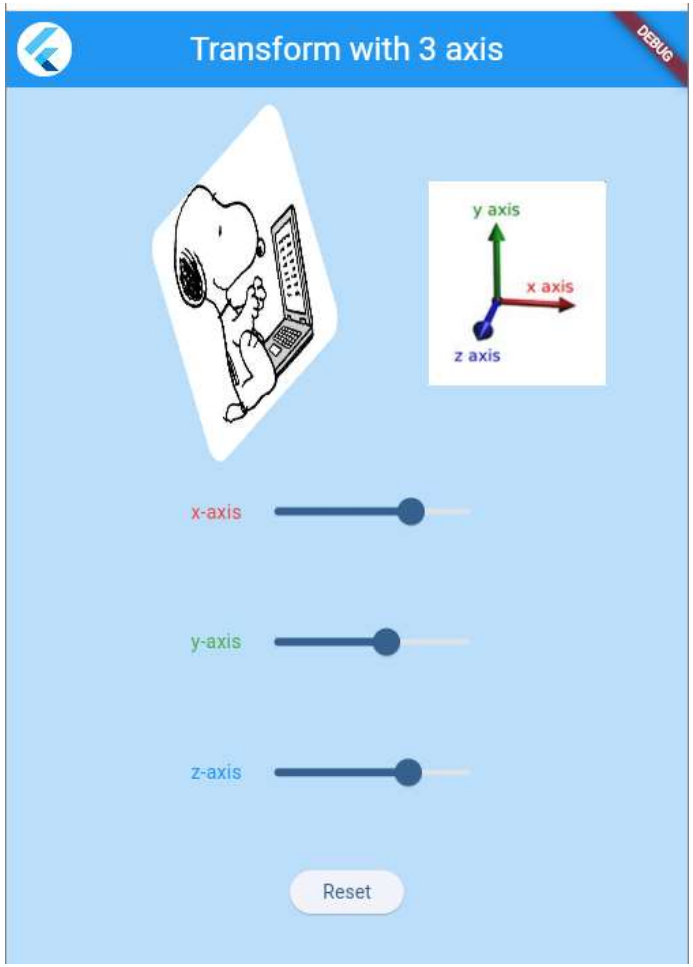
```dart
class AxisSlider extends StatelessWidget {
  const AxisSlider({
    super.key,
    required this.title,
    required this.color,
    required this.angle,
    required this.callback,
  });

  final String title;
  final Color color;
  final double angle;
  final Function(double value) callback;

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Text(title, style: TextStyle(color: color)),
        Slider(
          value: angle,
          min: -2 * pi,
          max: 2 * pi,
          onChanged: callback,
        ), // Slider
      ],
    ); // Row
  }
}
```

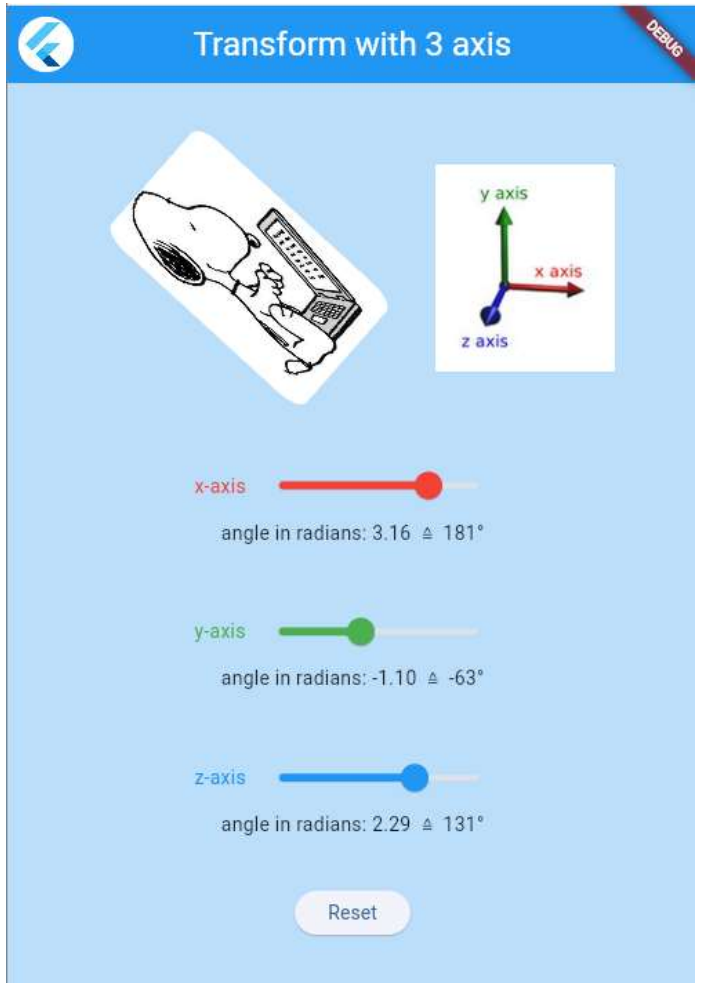
Instantiate the widget 3 times

```
AxisSlider(
  title: "x-axis",
  color: Colors.red,
  angle: angleX,
  callback: (value) {
    setState(() {
      angleX = value;
    });
  },
), // AxisSlider
AxisSlider(
  title: "y-axis",
  color: Colors.green,
  angle: angleY,
  callback: (value) {
    setState(() {
      angleY = value;
    });
  },
), // AxisSlider
AxisSlider(
  title: "z-axis",
  color: Colors.blue,
  angle: angleZ,
  callback: (value) {
    setState(() {
      angleZ = value;
    });
  },
), // AxisSlider
```



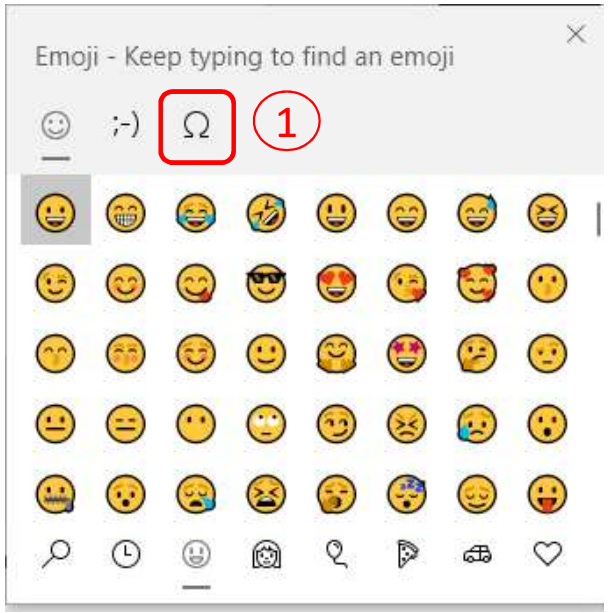
Beautify by code changes at one place only

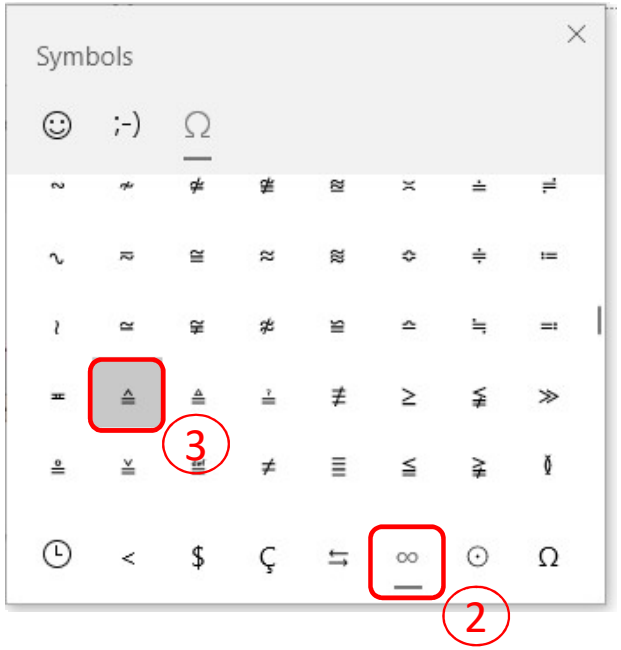
```dart
@override
Widget build(BuildContext context) {
  return Column(
    children: [
      Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Text(title, style: TextStyle(color: color)),
          Slider(
            value: angle,
            min: -2 * pi,
            max: 2 * pi,
            activeColor: color,
            onChanged: callback,
          ), // Slider
        ],
      ), // Row
      Text("angle in radians: ${angle.toStringAsFixed(2)} "
          " ≙  ${((angle / pi) * 180).toStringAsFixed(0)}°") // Text
    ],
  ); // Column
}
```



How to enter emojis or special characters like „≙“

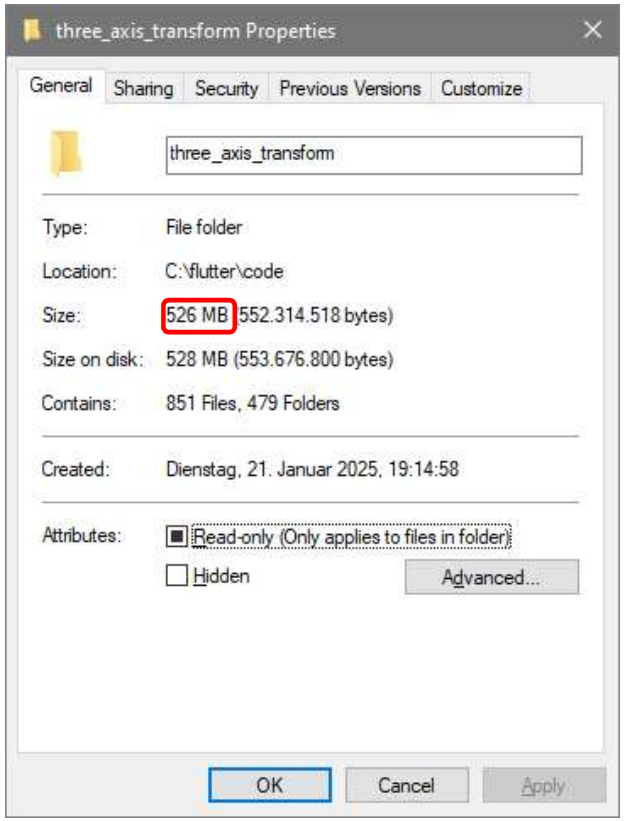
Press Windows-Key and “.”:

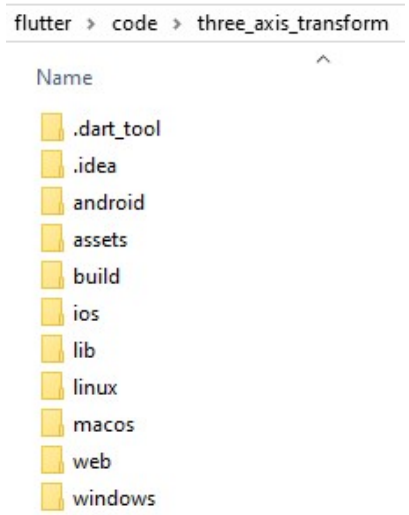


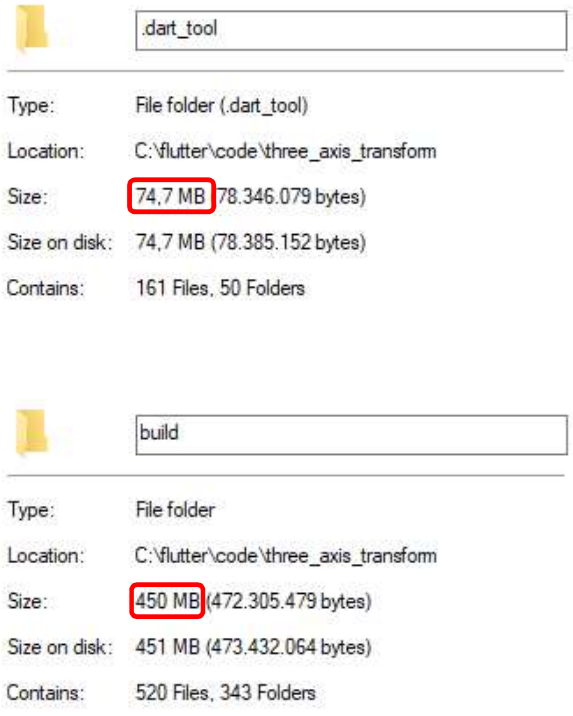


Flutter projects need disc space

When a flutter project is built, it takes a lot of MB on your disk:



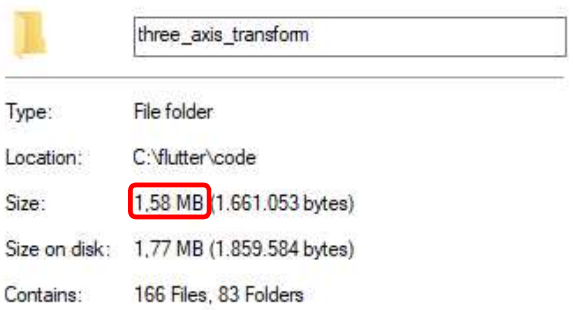


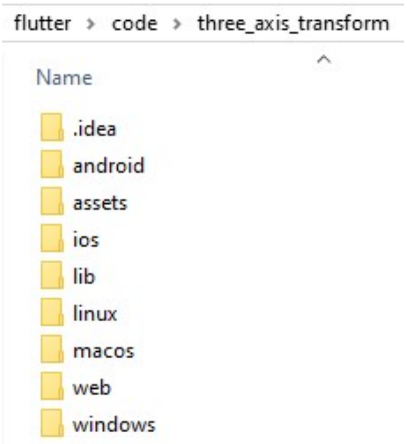


Cleaning Flutter projects

Enter “flutter clean” in Terminal:

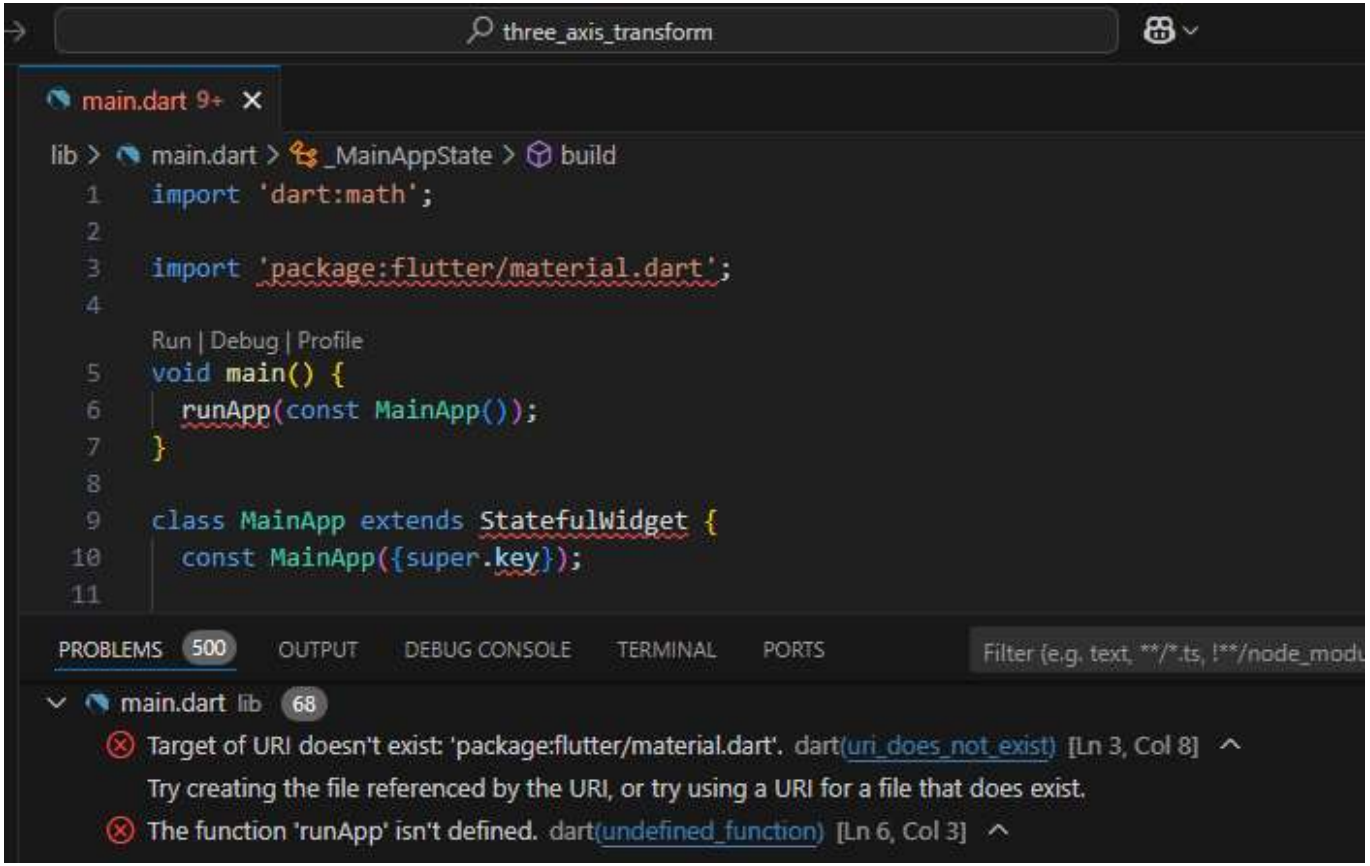
```
PS C:\flutter\code\three_axis_transform> flutter clean
Deleting build...                                  129ms
Deleting .dart_tool...                              30ms
Deleting Generated.xcconfig...                       0ms
Deleting flutter_export_environment.sh...            0ms
Deleting ephemeral...                                0ms
```



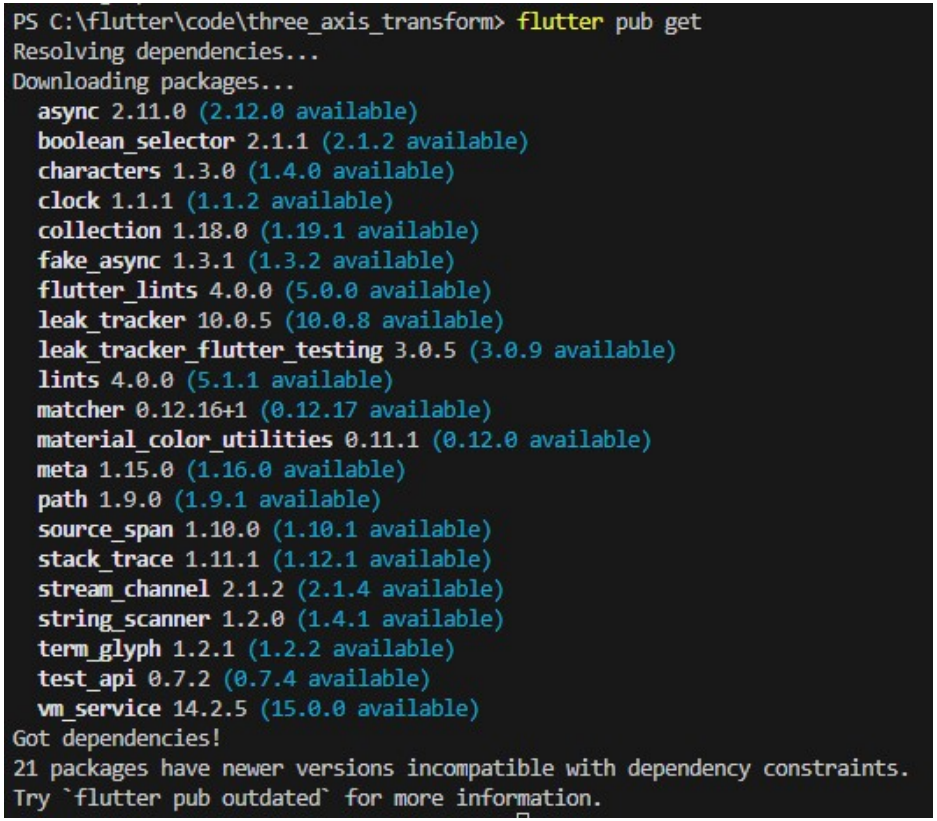


The folders “build” and “.dart_tool” have been deleted.

After cleaning, the Flutter project has a lot errors



To solve this run “flutter pub get” in Terminal



We will discuss on “flutter pub get” in more detail later in this training when we speak about packages in flutter.