1 【绘制开篇】纯粹的世界
学习时长: 28分31秒
2 【Paint篇】认识画笔的属性
学习时长: 22分20秒
3 【Canvas 上篇】画布绘制基础操作
学习时长: 44分6秒
4 【Canvas 下篇】画布绘制图片文字
学习时长: 41分39秒
5 【Path 上篇】路径的图形添加
学习时长: 27分49秒
6 【Path 下篇】路径的操作方法
学习时长: 36分14秒
7 【Color 上篇】熟悉而陌生的颜色
学习时长: 33分17秒
8 【Color 下篇】着色器和过滤器
学习时长: 45分10秒
9 【CustomPainter 篇】世界之基
学习时长: 19分6秒

本节目标:

```dart
[1]. 了解画笔的基础属性。
[2]. 了解画笔的线型属性。
[3]. 了解画笔的着色器效果。
[4]. 了解画笔的过滤器效果。
```

一、详细认识画笔 -- Level1

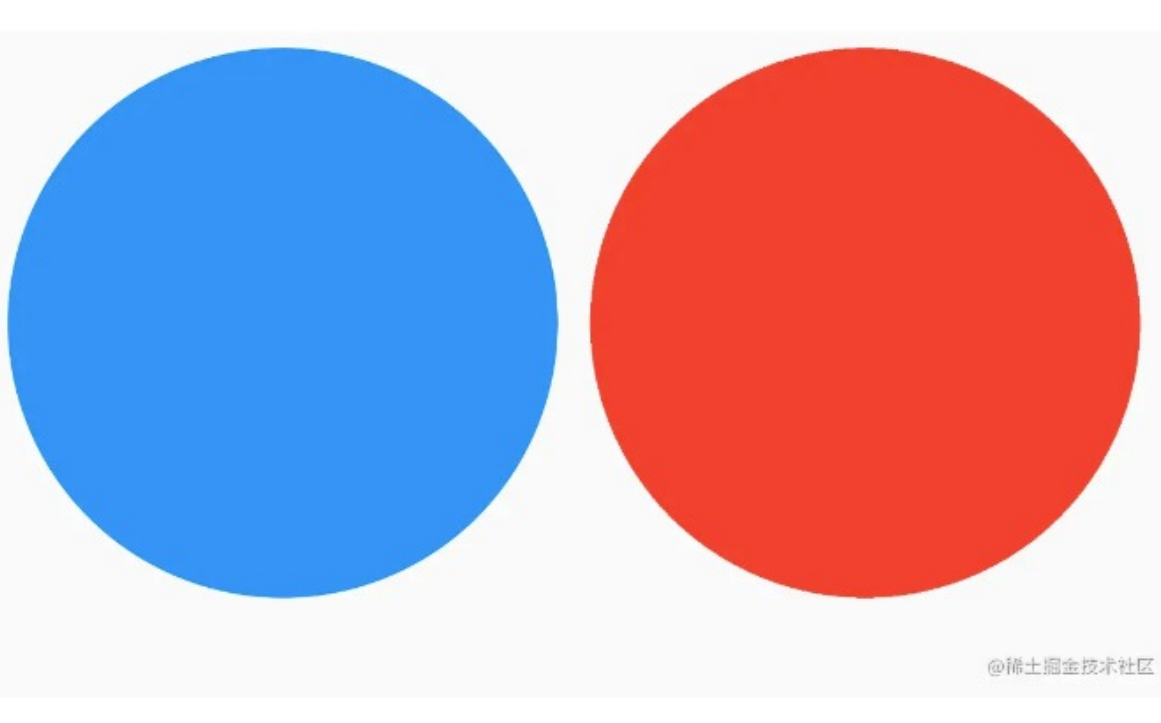
> 第一组基础属性:

属性	介绍	类型	默认值
isAntiAlias	是否抗锯齿	bool	true
style	画笔类型	PaintingStyle	PaintingStyle.fill
color	画笔颜色	Color	Color(0xff000000)
strokeWidth	线宽	double	0.0

---

1. 颜色 `color` 和是否抗锯齿 `isAntiAlias`

> 为了避免冗余，后面的代码中，只贴出 核心绘制代码 ,代码头部有对应的文件地址。
> 下面通过绘制两个圆形来测试 `color` 和 `isAntiAlias` 两个属性，仔细观察可发现:
> 左侧蓝色是抗锯齿的，边缘比较圆润，右侧红色是非抗锯齿的，边缘比较粗糙。

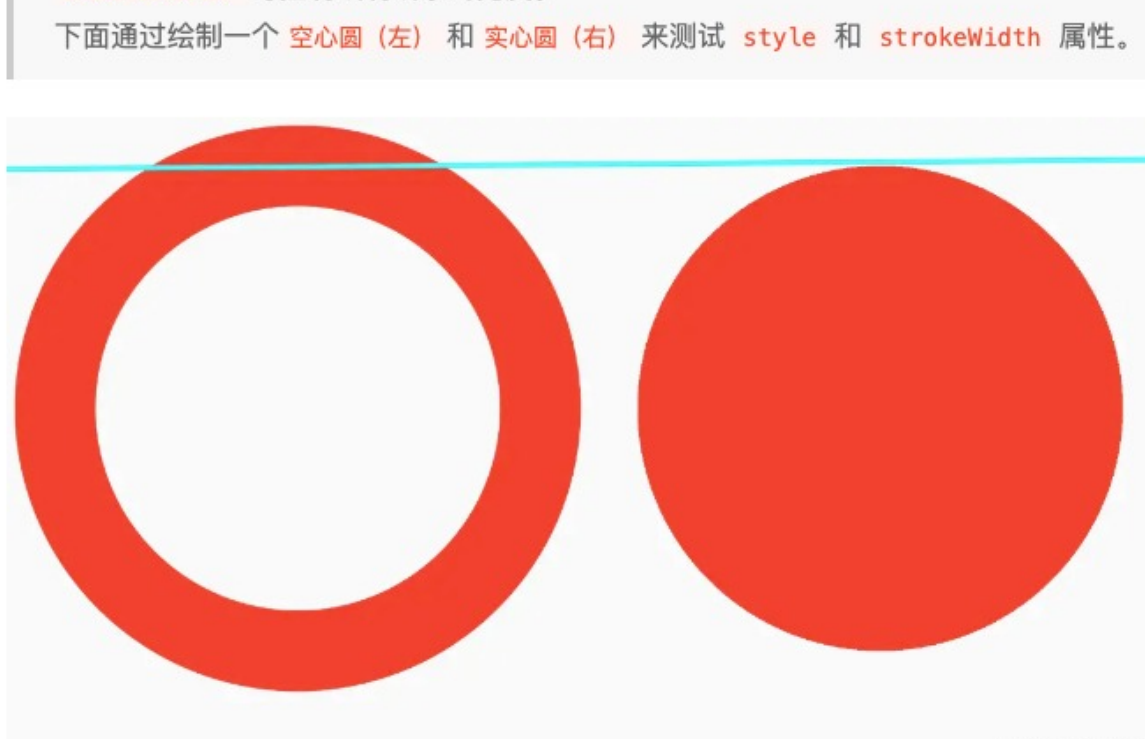


```Dart
---->[p02/level1/paper.dart]----
// 测试 isAntiAlias 和 color属性
void drawIsAntiAliasColor(Canvas canvas) {
  Paint paint = Paint();
  canvas.drawCircle(
      Offset(180, 180),
      170,
      paint
        ..color = Colors.blue
        ..strokeWidth = 5);
  canvas.drawCircle(
      Offset(180 + 360.0, 180),
      170,
      paint
        ..isAntiAlias = false
        ..color = Colors.red);
}
```

---

2. 画笔类型 `style` 和线宽 `strokeWidth`

> 画笔类型有填充 `PaintingStyle.fill` 和线条 `PaintingStyle.stroke` 。
> `strokeWidth` 可控制绘制线条的宽度。
> 下面通过绘制一个 空心圆（左） 和 实心圆（右） 来测试 `style` 和 `strokeWidth` 属性。



注意：只有画笔是 stroke 类型线宽才会起作用。 而且线条的宽度有一半在外侧，必要时需要进行校正

```Dart
---->[p02/level1/paper.dart]----
// 测试 style 和 strokeWidth 属性
void drawStyleStrokeWidth(Canvas canvas) {
  Paint paint =  Paint()..color=Colors.red;
  canvas.drawCircle(
      Offset(180, 180),
      150,
      paint
        ..style = PaintingStyle.stroke
        ..strokeWidth =50);
  canvas.drawCircle(
      Offset(180 + 360.0, 180),
      150,
      paint
        ..strokeWidth =50
        ..style = PaintingStyle.fill);
}
```

二、详细认识画笔 -- Level2

> 第二组线条属性:

属性	介绍	类型	默认值
strokeCap	线帽类型	StrokeCap	StrokeCap.butt
strokeJoin	线接类型	StrokeJoin	StrokeJoin.miter
strokeMiterLimit	斜接限制	double	0.0

---

1. 线帽类型 `strokeCap`

> 下图的效果依次是:
> `StrokeCap.butt - 不出头`
> `StrokeCap.round - 圆头`
> `StrokeCap.square - 方头`

```Dart
---->[p02_paint/level2/paper.dart]----
void drawStrokeCap(Canvas canvas) {
  Paint paint =  Paint();
  paint
    ..style = PaintingStyle.stroke
    ..color = Colors.blue
    ..strokeWidth = 20;
  canvas.drawLine(
      Offset(50, 50), Offset(50, 150), paint..strokeCap = StrokeCap.butt);
  canvas.drawLine(Offset(50 + 50.0, 50), Offset(50 + 50.0, 150),
      paint..strokeCap = StrokeCap.round);
  canvas.drawLine(Offset(50 + 50.0 * 2, 50), Offset(50 + 50.0 * 2, 150),
      paint..strokeCap = StrokeCap.square);
}
```

---

2. 线接类型 `strokeJoin`

线接类型只适用于Path的线段绘制。它不适用于用【Canvas.drawPoints】绘制的线。

> 下图的效果依次是:
> `StrokeJoin.bevel - 斜角、`
> `StrokeJoin.miter - 锐角、`
> `StrokeJoin.round - 圆角`

```Dart
---->[p02_paint/level2/paper.dart]----
void drawStrokeJoin(Canvas canvas) {
  Paint paint =  Paint();
  Path path =  Path();
  paint
    ..style = PaintingStyle.stroke
    ..color = Colors.blue
    ..strokeWidth = 20;
  path.moveTo(50, 50);
  path.lineTo(50, 150);
  path.relativeLineTo(100, -50);
  path.relativeLineTo(0, 100);
  canvas.drawPath(path, paint..strokeJoin = StrokeJoin.bevel);

  path.reset();
  path.moveTo(50 + 150.0, 50);
  path.lineTo(50 + 150.0, 150);
  path.relativeLineTo(100, -50);
  path.relativeLineTo(0, 100);
  canvas.drawPath(path, paint..strokeJoin = StrokeJoin.miter);

  path.reset();
  path.moveTo(50 + 150.0 * 2, 50);
  path.lineTo(50 + 150.0 * 2, 150);
  path.relativeLineTo(100, -50);
  path.relativeLineTo(0, 100);
  canvas.drawPath(path, paint..strokeJoin = StrokeJoin.round);
}
```

---

3. 斜接限制 `strokeMiterLimit`

strokeMiterLimit只适用于【StrokeJoin.miter】。
它是一个对斜接的限定，如果超过阈值，会变成【StrokeJoin.bevel】。
因为考虑到锐角太尖，会影响视觉。

> 第一行strokeMiterLimit = 2;
> 第二行strokeMiterLimit = 3;
> 数字越大，允许出现的尖角就可以越尖 。

```dart
---->[p02_paint/level2/paper.dart]----
void drawStrokeMiterLimit(Canvas canvas) {
  Paint paint =  Paint();
  Path path =  Path();
  paint
    ..style = PaintingStyle.stroke
    ..color = Colors.blue
    ..strokeJoin = StrokeJoin.miter
    ..strokeWidth = 20;
  for (int i = 0; i < 4; i++) {
    path.reset();
    path.moveTo(50 + 150.0 * i, 50);
    path.lineTo(50 + 150.0 * i, 150);
    path.relativeLineTo(100, -(40.0 * i + 20));
    canvas.drawPath(path, paint..strokeMiterLimit = 2);
  }
  for (int i = 0; i < 4; i++) {
    path.reset();
    path.moveTo(50 + 150.0 * i, 50 + 150.0);
    path.lineTo(50 + 150.0 * i, 150 + 150.0);
    path.relativeLineTo(100, -(40.0 * i + 20));
    canvas.drawPath(path, paint..strokeMiterLimit = 3);
  }
}
```

三、详细认识画笔 -- Level3

> 第三组颜色属性:

属性	介绍	类型	默认值
shader	着色器	Shader	null
blendMode	混合模式	BlendMode	BlendMode.srcOver
invertColors	是否反色	bool	false

---

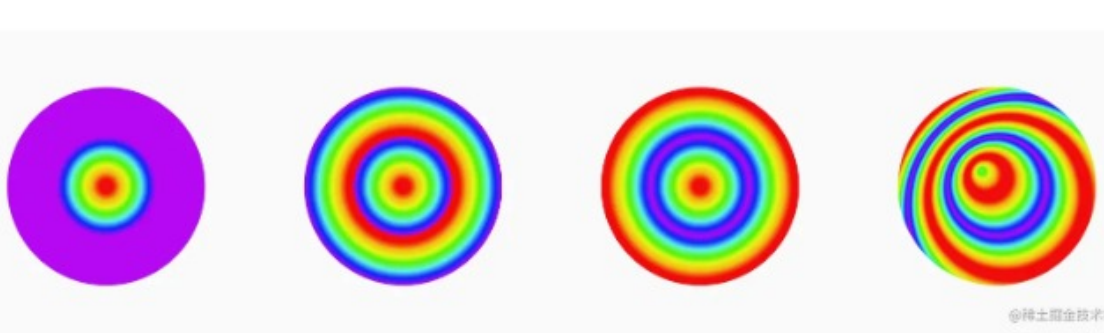
1. 着色器shader

```lua
Shader  抽象类
|---- Gradient 渐变着色
               |--- linear 线性渐变
               |--- radial 径向渐变
               |--- sweep  扫描渐变
|--- ImageShader 图片着色
```

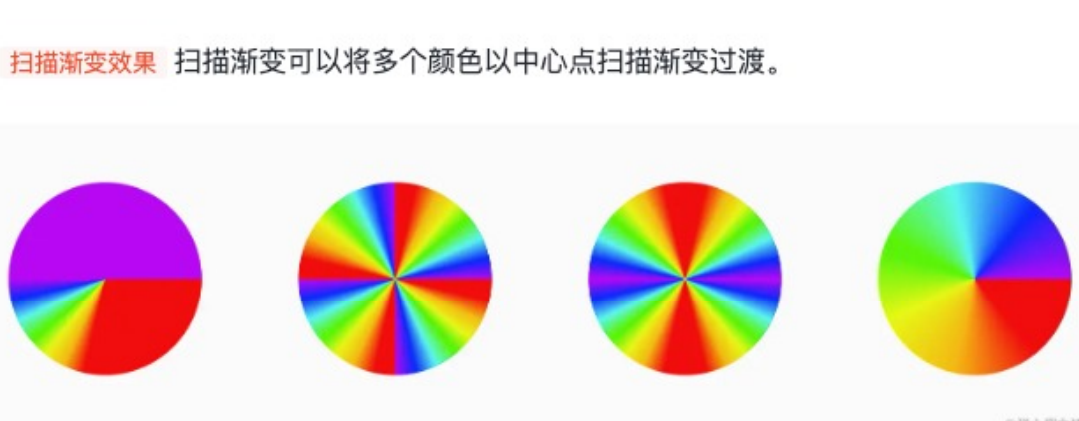
> 由于这部分比较复杂，为了不让本篇过长，将第 8 篇进行详细讲述，本篇会展示使用 shader 实现的效果。

- 线性渐变效果 线性渐变可以将多个颜色进行单向渐变过渡。

- 径向渐变效果 径向渐变可以将多个颜色以一点沿径向渐变过渡。



- 扫描渐变效果 扫描渐变可以将多个颜色以中心点扫描渐变过渡。



- 图片着色器ImageShader 使用 `ImageShader` 可以加载一张图片，绘制时使用图片对图形进行着色。

---

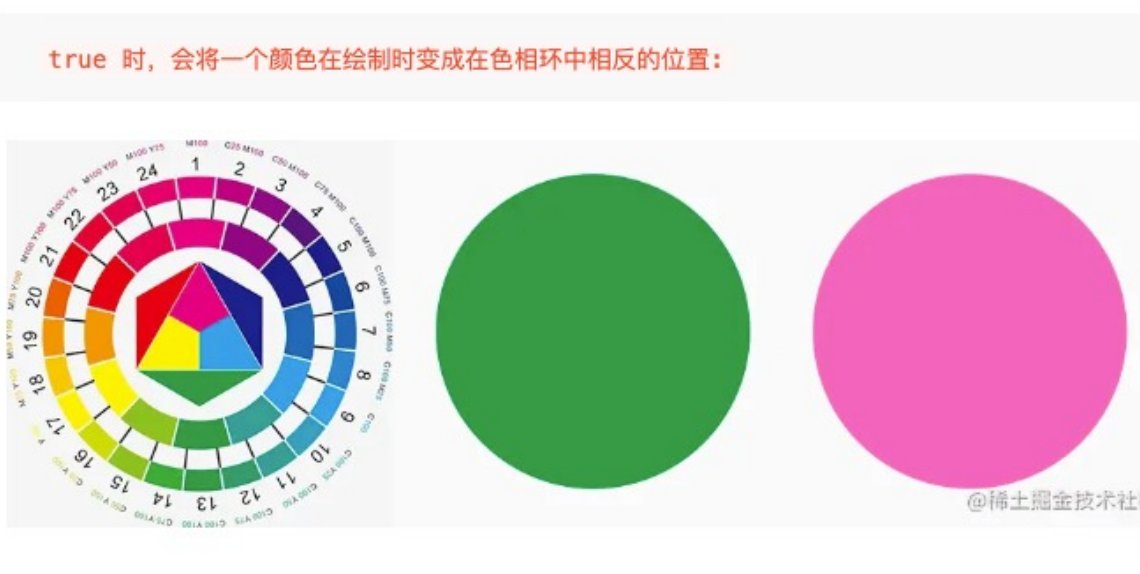
3. 颜色叠合模式 `blendMode`

> `BlendMode` 在组件中的应用有 `Image 组件` 和 `ColorFilter 组件`
> 用于将目标与一个颜色叠合,一共有如下 29 种叠合模式 ，这里看一下效果。

---

4. 是否反色 `invertColors`

> `true` 时，会将一个颜色在绘制时变成在色相环中相反的位置:



```Dart
void drawInvertColors(Canvas canvas) {
  _paint..color = Color(0xff009A44);
  canvas.drawCircle(Offset(100, 100), 50, _paint..invertColors = false);
  canvas.drawCircle(Offset(100+120.0, 100), 50, _paint..invertColors = true);
}
```

四、详细认识画笔 -- Level4

> 第四组滤镜属性:

属性	介绍	类型	默认值
colorFilter	颜色滤镜	Shader	null
maskFilter	遮罩滤镜	MaskFilter	null
imageFilter	图片滤镜	ImageFilter	null
filterQuality	滤镜质量	FilterQuality	FilterQuality.none

> 这部分实现的代码比较复杂，将在第 8 篇进行详解，这里先看一下效果，进行简单的认识。

---

1. 颜色滤镜 `colorFilter`

> `ColorFilter` 对象可以使用变换矩阵或颜色叠合模式对绘制的对象进行滤色处理。

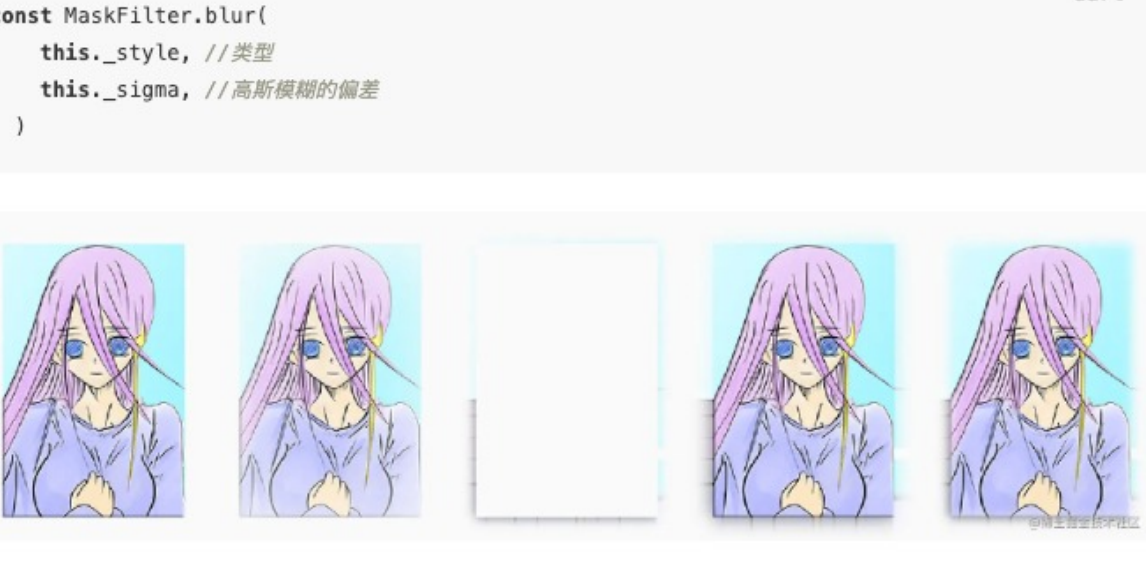
```dart
const ColorFilter.mode(Color color, BlendMode blendMode) #颜色模式
const ColorFilter.matrix(List<double> matrix) #颜色矩阵变换
```

---

2. 遮罩滤镜 `maskFilter`

> 使图片进行模糊，可以指定模糊的类型, 只有一个 `MaskFilter.blur` 构造

```dart
const MaskFilter.blur(
    this._style, //类型
    this._sigma, //高斯模糊的偏差
  )
```



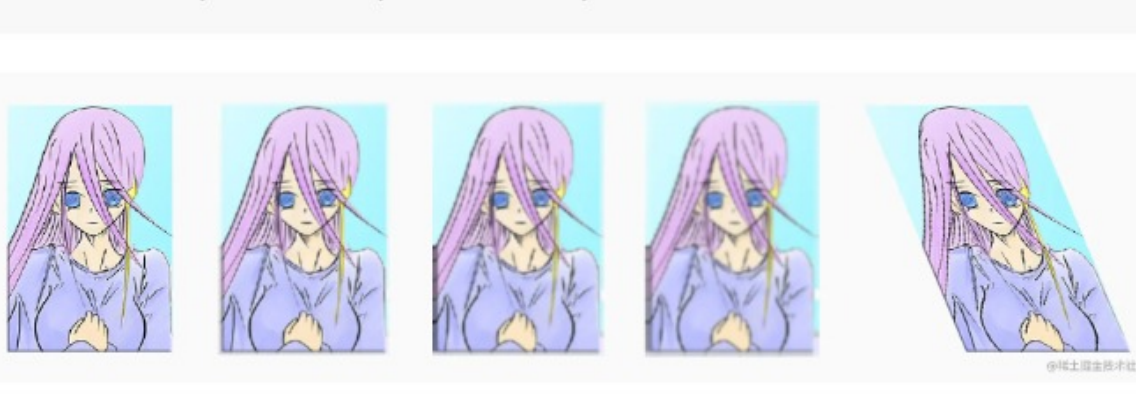
---

3. 图片滤镜 `imageFilter`

> 可以通过 `ImageFilter.blur` 来让图片模糊，
> 或通过 `ImageFilter.matrix` 进行变换

```dart
ImageFilter.blur(
  { double sigmaX = 0.0,
    double sigmaY = 0.0
  })

ImageFilter.matrix(
  Float64List matrix4,
  { FilterQuality filterQuality = FilterQuality.low })
```

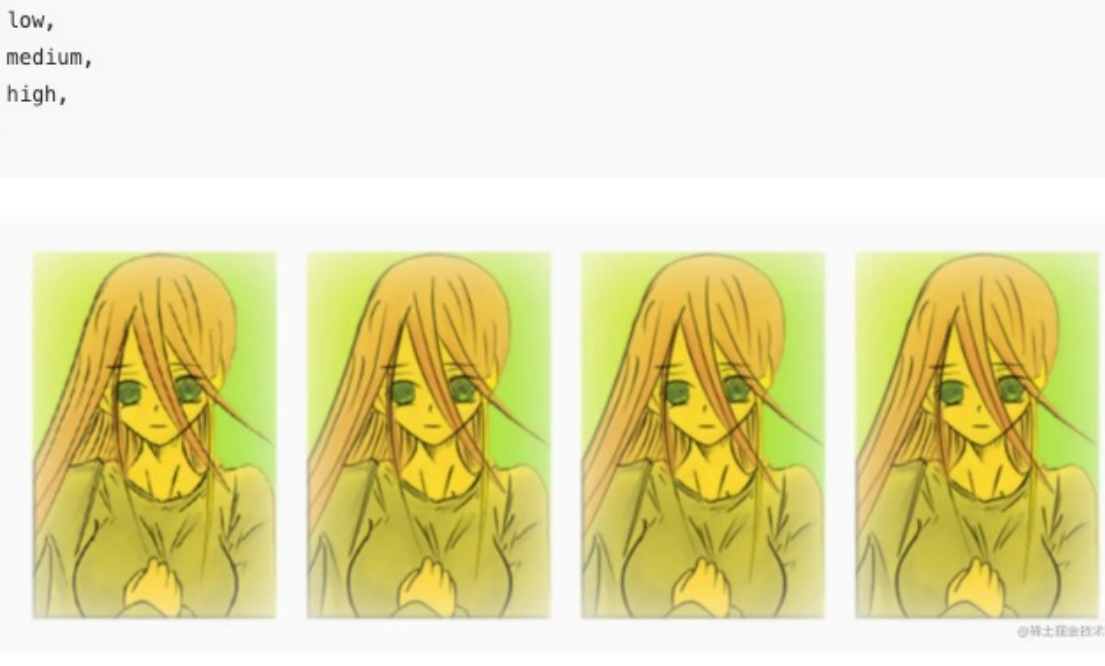


---

4. 滤镜质量 `filterQuality`

> 一共四种类型: 表现依次如下

```dart
enum FilterQuality {
  none,
  low,
  medium,
  high,
}
```



> 至此，关于 Paint 画笔的所有属性就全部介绍完毕。一些用法较复杂的属性，将会在后面进行详解。现在你应该对 Paint 可以实现哪些效果有一个大概的认识。既然已经有了笔，那下面看看画布 `Canvas` 吧！

留言

输入评论（Enter换行，⌘ + Enter发送）

发表评论

全部评论（20）

风二中 LV.1 JY.4 | Android&Flutter | 3月前

打卡

点赞 回复

衿璃 JY.4 | flutter | 4月前

第二章over 吃饭

点赞 回复