多选题

1、下面哪些组件具备居中功能？（ABC）

A、Container B、Center C、Align D、Card

2、Textfield 需要设置哪些属性才能实现无限换行和扩展高度？（AD）

A、expands B、maxLength C、textAlign D、maxLines

3、如何让图片重复堆叠容器？（C）

A、fit: BoxFit.fill B、fit: BoxFit.cover C、repeat: ImageRepeat.repeat

4、Text 组件有哪些属性不能为空？（A）

A、data B、style C、overflow D、maxLines

5、哪些组件可以使用属性控制 child 的显隐（CD）？

A、Card B、Center C、Offstage D、Visibility

6、下面对 StatefulWidget 中 createState 方法描述正确的是？（ABC）

A、在树中的给定位置为此小部件创建可变状态；

B、框架可以在整个生命周期中多次调用此方法；

C、如果将小部件从树中删除，并且稍后再次插入树中，框架将调用 createState 再次创建一个新的 State 对象，从而简化了 State 对象；

D、用来看的，没啥实际作用；

7、下面哪些是正确调用 setstate 的姿势？（AC）

A、在点击某按钮时触发进行刷新类实例变量；

B、在点击某按钮时触发进行刷新方法实例变量；

C、使用前先判断 mounted 是否为 true；

D、dispose 生命周期中调用；

8、让 AppBar 的标题居中的方式有（AB）

A、centerTitle 属性设置为 true；

B、Center 组件包裹 title；

C、elevation 设置为 0；

D、primary 设置为 false；

9、controller 是一个动画控制器，如果 controller 声明的时候为空了，使用 controller?.forward()播放动画时会发生什么？（CD）

A、一片红的报错信息；

B、报错，但不影响布局；

C、不会报错；

D、因为是?.的调用，左边为空了则不会去调用播放方法；

10、在有状态类中编写一个按钮调用初始化生命周期（initState）方法，会发生什么？（BD）

A、一片红的报错信息；

B、报错，但不影响布局；

C、不会报错；

D、会报生命周期创建错误；

11、写布局时用 ListView 然后 children 并没有写间隔之类的，结果出现了上下有间距，为什么？（AB）

A、可能是 ListView 外层写了间隔之类的；

B、可能是 ListView 自带 Padding；

C、肯定是官方的问题；

D、人的问题，换个人来运行可能就好了；

12、属于滑动组件的有（AB）

A、ListView B、GridView C、Container D、Center

13、属于 Sliver 系列的组件有（ABC）

A、SliverToBoxAdapter

B、SliverAppBar

C、SliverAnimatedOpacity

D、Image

14、属于不可滑动的组件有（ACD）

A、Text B、SingleChildScrollView C、Card D、FlatButton

15、能操作对齐方式（Alignment）的组件有（CD）

A、FlatButton B、RaisedButton C、Align D、Container

16、当我给变量赋值的时候想为空的时候才赋值应该怎么操作（ABC）

A、使用??=来指定当左边为空时才执行等于；

B、使用三元运算符判断是否等于空才执行赋值；

C、使用 if 判断语句，为空时才等于；

D、直接使用等于来赋值；

17、去掉按钮水波纹效果的方式有（ABC）

A、全局设置水波纹颜色为透明；

B、局部设置水波纹颜色为透明然后包裹需要透明的组件；

C、使用按钮的水波纹颜色属性设置为透明；

D、发呆就对了，阿巴阿吧阿吧；

18、在什么情况下类可以混入 TickerProviderStateMixin？（CD）

A、用手写代码的情况下；

B、在不摸鱼的情况下；

C、混入 TickerProviderStateMixin 必须是继承 State 的情况下；

D、具备有状态类的生命周期下；

19、在 Android 和 IOS 情况下 ListView 的滑动表现描述正确的是（AC）

A、每个平台都有判断，效果会出现不同点；

B、Android 和 IOS 怎么滑效果都一样；

C、Android 滑动到无法滑动之后会出现焦点特殊动画，而 IOS 是回弹效果；

D、我又没滑过，我懒个晓得咯；

20、容器设置了约束盒子最小宽度为 200，但内部的宽度设置为了 50，则会发生（A）

A、以约束盒子的 200 为准；

B、以内部设置的 50 为准；

C、会造成混乱，导致无法显示；

D、有时候会 200，有时候会 50；

赋值操作符

1、AA ?? “999”表示：AA 如果为空，则显示 999

2、AA ??= “999”表示：AA 如果为空，则给值设置为 999

3、AA ~/999 表示：AA 对于 999 整除

4、A?.a 表示：A 如果为空则不调用，如果不为空则调用 a 值

5、A?.a??”999”表示：A 如果为空则不调用，如果不为空则调用 a 值，然后 a 值为空则显示 999

判断题

1、WidgetsFlutterBinding.ensureInitialized 在 runApp 之后调用（错）；

2、MaterialApp 中 router 内有“/”可以和 home 属性共存（错）；

3、构造方法中“Key key”必写（错）；

4、“=>”操作符是大括号的简写方式（对）；

5、使用 overlay 写全局提示框不需要上下文（错）；

6、MaterialApp 组件中 home 属性有时可以为空（对）；

7、setState 必须是在有状态类中调用（对）；

8、final 修饰的变量无法被 setState 刷新（对）；

9、方法内声明的变量可以被 setState 刷新（错）；

10、NestedScrollView 的内控制器可以从树上找（对）；

11、约束盒子最小宽度为 10，包裹容器设置宽度为 5，结果为 5（错）；

12、ListView 的 children 写容器可以直接设置容器宽度并生效（错）；

13、在 initState 生命周期可以拿到上下文干和 build 生命周期一样的事（错）；

14、Scaffold 的左右抽屉可以并存（对）；

15、整 AppBar 高度是由状态栏高度加 kToolbarHeight 组成的（对）；

16、媒体查询是获取上下文宽高数值的方法（错）；

17、SizedBox.expand 内宽和高都是无限值（对）；

18、Container 使用了 decoration 属性外面还可以设置 color 属性值（错）；

19、一个 App 项目存在多个 MaterialApp 组件是合理的（错）；

20、Platform.isAndroid 可以在 web 平台中判断当前是什么系统（错）；

21、showDialog 是一个组件（错）；

22、CupertinoAlertDialog 只能在 CupertinoApp 情况下使用（错）；

23、Divider 组件其实就是 SizedBox+Center+Container 组合出来的（对）；

24、ScrollConfiguration 组件可以自定义 behavior 然后去掉滑动的焦点（对）；

25、PageView 有自动轮播功能（错）；

26、debugShowCheckedModeBanner 属性 false 可以去掉右上 DEBUG 标签（对）；

27、AnimatedList 中 key 是必填的（错）；

28、AnimatedSwitcher 中 child 属性 key 可以不填但可能没效果（对）；

29、Spacer 组件可以放在任何可以放组件的地方（错）；

30、容器宽高未知的情况下可以使用 Expanded 组件（错）；

31、BottomNavigationBar 具备保存页面状态功能（错）；

32、Card 组件不会默认带阴影效果（错）；

32、Center 组件可以让 child 位于容器下方（错）；

33、LinearProgressIndicator 可以直接设置 Color 类型的 valueColor（错）；

34、VerticalDivider 和 Divider 用法一样，前者垂直分割线，后者水平分割线（对）；

35、RaisedButton 中 onPressed 为 null 或不设置时，按钮是禁用状态（对）；

36、BackButton 组件默认 Android 和 IOS 效果不一致（对）；

37、CupertinoButton 只能在 IOS 风格 App 下使用（错）；

38、Hero 飞行动画无需用到 tag（错）；

39、Row 中放 Textfield 无需知道 Textfield 所占宽度（错）；

40、TabBar 只能放在 AppBar 的 bottom 属性（错）；

问答题

1、说下 Flutter 的优缺点

优点：1、高性能，2、高保真，3、相对易学，4、热重载；

缺点：1、不支持热更新，2、生态需完善；

2、Flutter 跨平台原理是怎么样的？

自写 UI 渲染引擎实现跨平台

3、说下移动端跨平台技术划分

1、web 技术，2、原生渲染，3、自渲染技术；

4、简单的说下移动端跨平台技术演进

A：

5、蓝湖设计图有一张轮播图，宽度是 335 高度是 120，左右间隔是 10， 如何使用屏幕算法适配全机型屏幕宽和高？

分析：

- 左右间隔：设置 margin 然后左右 10 个间隔；
- 宽度：整宽减 20，20 就是左右的间隔；

- 高度：(宽度) * 120 / 335；

代码：

```
new Container(
  height: (winWidth(context) - 20) * 120 / 335,
  width: winWidth(context) - 20,
  margin: EdgeInsets.symmetric(horizontal: 10.0),
  alignment: Alignment.center,
  decoration: BoxDecoration(
    borderRadius: BorderRadius.all(Radius.circular(4.0)),
    color: Colors.amber.withOpacity(0.5),
  ),
  child: new Text('模拟图片'),
),
```

复制代码

效果


11:27
Flutter高级
模拟图
Flutter高级
模拟图片


11:27
Flutter高级
模拟图
Flutter高级
模拟图片

6、未知数据数量有规则的列表视图，要求一行显示 5 个，每个间隔为 10（含上下），最外边距 margin 左右都为 20，高度为 50，多出的数据继续往下排并向左对齐，适配任何机型，怎么做？

分析：

- 左右间隔：设置 margin 然后左右 20 个间隔；
- 间隔和高：除最外边左右，内边都为 10 间隔，并包含上下，高度固定 50；
- 对齐方式：对齐方式默认都为向左对齐；
- 组件：推荐 Wrap，动态数据，依次撑开；

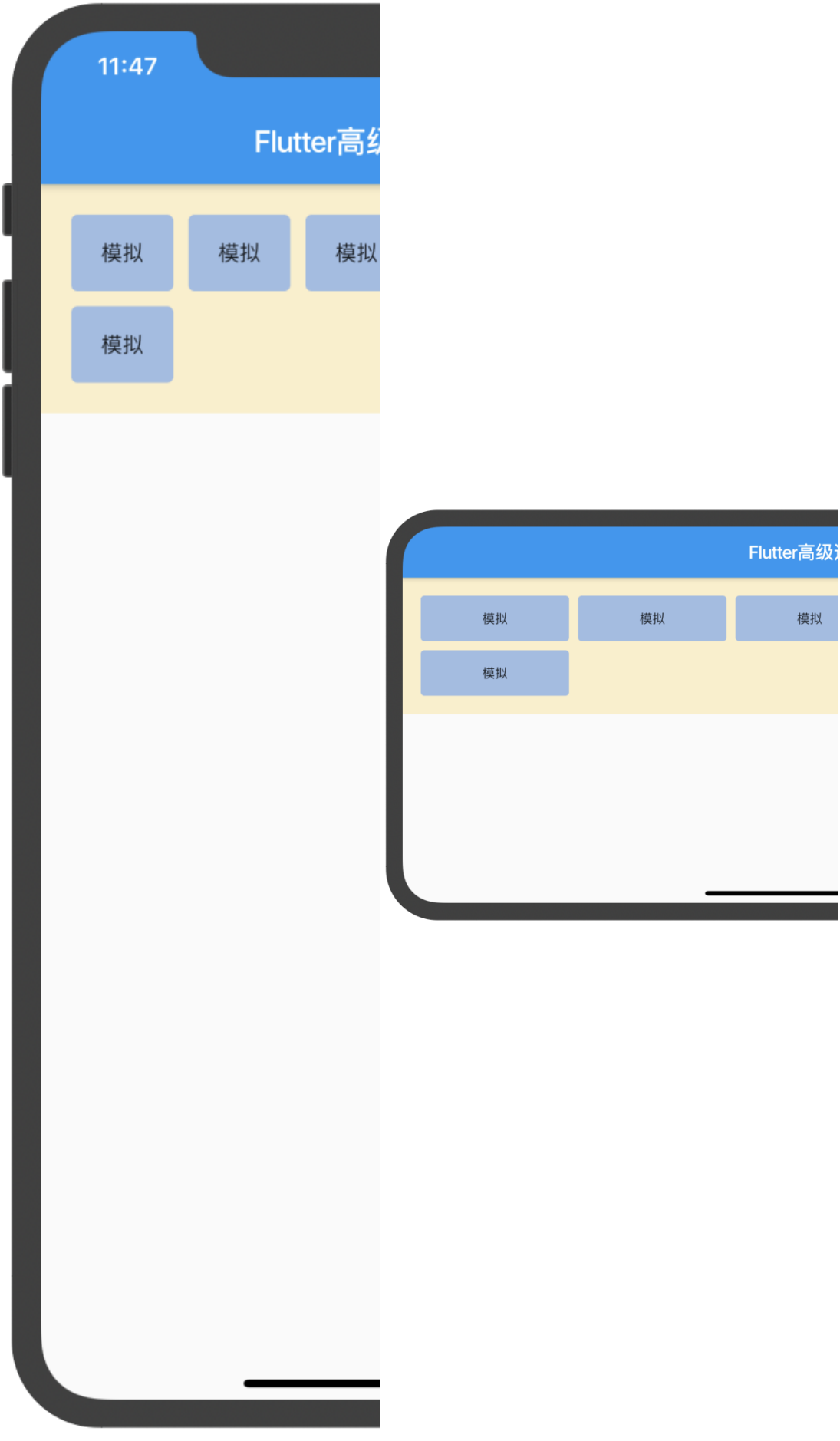
代码：

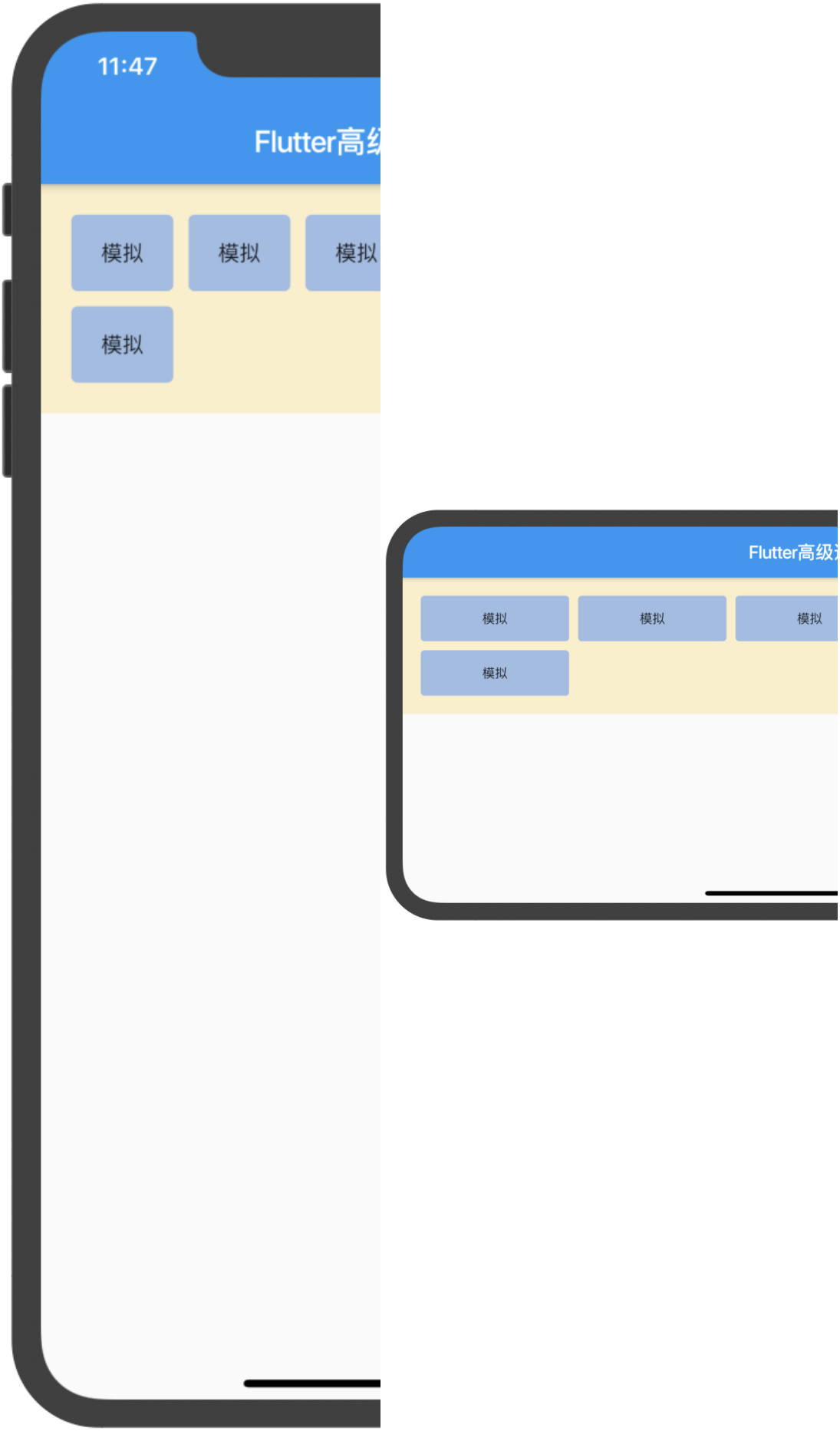
```
class MyHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return new Scaffold(
      appBar: new AppBar(
        title: new Text('Flutter 高级进阶'),
      ),
      body: new Container(
        padding: EdgeInsets.symmetric(vertical: 20.0),// 为了保持美观给了上下价格 20
        color: Colors.amber.withOpacity(0.2), // 为了验证动态撑开给了背景
        child: TestRoute(),// 主代码
      ),
    );
  }
}

class TestRoute extends StatefulWidget {
  @override
  _TestRouteState createState() => _TestRouteState();
}

class _TestRouteState extends State<TestRoute> {
  Widget buildItem(item) {
    return new Container(
      decoration: BoxDecoration(
        borderRadius: BorderRadius.all(Radius.circular(4.0)), // 圆角
```

```
        color: Colors.blueAccent.withOpacity(0.5),// item颜色
      ),
      height: 50.0, // 高度
      alignment: Alignment.center, // item文本剧中
      width: (winWidth(context) - 80) / 5, // 宽度
      child: new Text('模拟'),
    );
  }

  @override
  Widget build(BuildContext context) {
    return new Container(
      width: winWidth(context) - 40, // 宽度容器算法
      margin: EdgeInsets.symmetric(horizontal: 20.0),
      child: new Wrap(
        spacing: 10.0,
        runSpacing: 10.0,
        children: [0, 1, 2, 3, 4, 5].map(buildItem).toList(),
      ),
    );
  }
}
```

复制代码

效果


11:47
Flutter高级
模拟
模拟
模拟
模拟
Flutter高级
模拟
模拟
模拟
模拟


11:47
Flutter高级
模拟
模拟
模拟
模拟
Flutter高级
模拟
模拟
模拟
模拟

7、如何实现点击空白区域收起键盘？

触发代码:FocusScope.of(context).requestFocus(new FocusNode());

8、无需上下文进行路由跳转原理是怎么样的？

使用 GlobalKey 调用到 NavigatorState 的方法;

9、为什么我的容器无论设置宽高多少都是占整个的宽高？怎么解决？

1.被约束了；2.宽高无限了；

解决方式：

使用 UnconstrainedBox 包裹容器；

10、键盘弹出时底部溢出如何解决？

1.溢出部分增加滑动属性； 2.Scaffold 的 resizeToAvoidBottomPadding: false 让其遮挡布局；

11、Container 设置 borderRadius 不生效怎么解决？如何导致的？

原因：Container 设置 borderRadius 只对当前盒子生效；

解决方式：使用裁剪方式，如：ClipRRect 组件；

12、GestureDetector 设置 onTap 不生效怎么解决？

使用 GestureDetector 的 behavior: HitTestBehavior.translucent

13、如何实现动态更改 TabBar 长度？

1.控制器重新赋值；（使用 Ticker 不能是 Single 的）

2.使用 DefaultTabController；

14、为何写多个动画时动画控制器使用了 Ticker，类也混入了 Ticker 运行报错了？

有可能使用了单一的 Ticker（SingleTickerProviderStateMixin），只能使用一次就失效了；

15、如何实现键盘弹出后遮住布局，而不是顶起布局？

Scaffold 属性 resizeToAvoidBottomPadding: false 让其遮挡布局；

16、为何输入框输入内容之后返回到桌面，再进入 app 时内容被清空了？怎么解决？

可能是输入中没有做保存处理，可以使用生命周期判断，在程序暂停前让输入框取消焦点即可实现自动保存；

17、为何本地资源图片刚进入的时候切换到另一张出现白屏？怎么解决？

原因：切换之后才开始解析本地资源图片；

解决方案：在初始化的时候就加载指定 asset 图片，而不是在需要展示的时候才开始加载。

代码示例：

```
@override
  void initState() {
    super.initState();
    WidgetsBinding.instance.addPostFrameCallback((_) async {
     // _imageUrls 就是数组的 Asset 图片地址
      _imageUrls.forEach((image) {
        precacheImage(AssetImage(image), context);
      });
    });
  }
```

复制代码

18、如何拦截 App 返回事件，用什么组件？

WillPopScope 组件，用返回的 bool 来操作是否允许返回；

19、如何监听 App 暂停运行或不可见状态事件？

使用 WidgetsBindingObserver 观察生命周期状态；

具体：book.flutterj.com/chapter1/li…

20、Text 的 TextOverflow.ellipsis 不生效如何解决？

让 Text 组件的所占宽度可知；

21、如何获取控件的大小和位置？

1、使用 Key 拿到上下文取得 findRenderObject 拿内容的尺寸数据；

2、使用 context 取得 findRenderObject 拿内容的尺寸数据；

22、类构造方法后面加个 super 表示什么意思？

调用父类的属性，可进行赋值传输；

23、assert(data != null, ‘no data’)是什么意思？

assert：断言；

data != null：data 不能为空，否则触发断言错误；

no data：如果触发断言则提示的内容；

24、const 修饰构造函数和放声明数值前分别有什么作用？

构造函数前：构造函数会在编译期和常量一起被编译；

声明数值前：一个不可变的常量，编译期就被初始化；

25、描述下 getter setter 和重写

Dart 中所有的基础类型、类等都继承 Object ，默认值是 NULL， 自带 getter 和 setter ，而如果是 final 或者 const 的话，那么它只有一个 getter 方法，Object 都支持 getter、setter 重写

26、Assert(断言)有什么作用？什么时候有效？

在 debug 的时候提示出断言错误让开发者知悉，只在 debug 有效；

Dart 相关

1、Dart 当中的 「..」表示什么意思？

级连操作符

“..” 和 “.” 不同：调用..后返回的相当于是 this，而.返回的则是该方法返回的值；

2、Dart 的作用域是怎么样的？

Dart 没有 public 和 private 等关键词，默认就是公开的，私有变量使用下划线开头；

4、dart 是多线程还是单线程执行？

单线程执行，多线程是使用异步来执行的；

5、阻塞式调用和非阻塞式调用是怎么样的？

阻塞：调用结果之前，当前线程会被挂起，调用线程只有在得到结果之后才会继续执行；

非阻塞：调用执行之后，当前线程不会停止运行，只需要过一段时间来检查有没有结果返回即可；

6、事件循环是什么？

将需要处理的一系列事件，放在一个事件队列（Event Queue）中，不断从事件队列中取出事件，并执行需要执行的代码块，直到事件被清空。

7、dart 是值传递还是引用传递？

dart 是值传递。我们每次调用函数，传递过去的都是对象的内存地址，而不是这个对象的复制。

8、Dart 语言有哪些重要的特性？

- Productive（生产力高，Dart 的语法清晰明了，工具简单但功能强大）
- Fast（执行速度快，Dart 提供提前优化编译，以在移动设备和 Web 上获得可预测的高性能和快速启动。）
- Portable（易于移植，Dart 可编译成 ARM 和 X86 代码，这样 Dart 移动应用程序可以在 iOS、Android 和其他地方运行）
- Approachable（容易上手，充分吸收了高级语言特性，如果你已经知道 C++，C 语言，或者 Java，你可以在短短几天内用 Dart 来开发）
- Reactive（响应式编程）

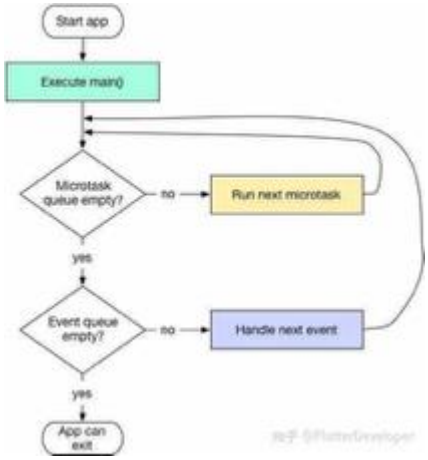
9、Dart 语言有哪些重要的概念？

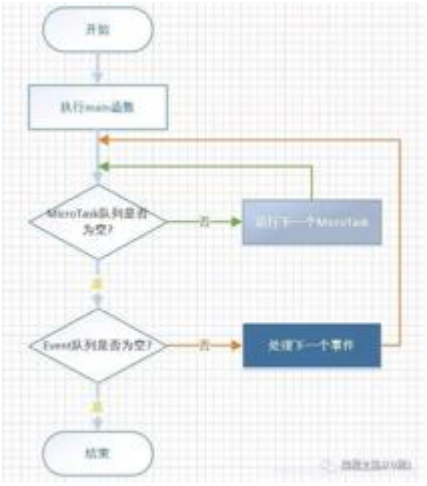
- 在 Dart 中，一切都是对象，所有的对象都是继承自 Object
- Dart 是强类型语言，但可以用 var 或 dynamic 来声明一个变量，Dart 会自动推断其数据类型,dynamic 类似 c#
- 没有赋初值的变量都会有默认值 null
- Dart 支持顶层方法，如 main 方法，可以在方法内部创建方法
- Dart 支持顶层变量，也支持类变量或对象变量
- Dart 没有 public protected

private 等关键字，如果某个变量以下划线（_）开头，代表这个变量在库中是私有的

10、Dart 线程模型是如何执行的？

Dart 是单线程模型，运行的的流程如下图。

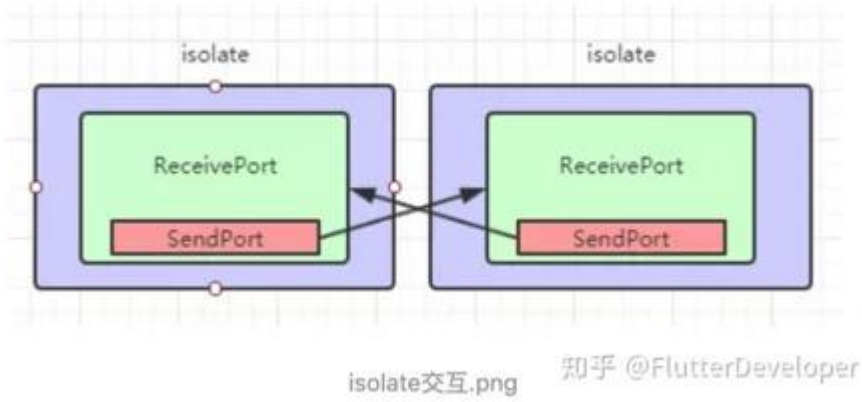




Dart 在单线程中是以消息循环机制来运行的，包含两个任务队列，一个是“微任务队列” microtask queue，另一个叫做“事件队列” event queue。 当 Flutter 应用启动后，消息循环机制便启动了。首先会按照先进先出的顺序逐个执行微任务队列中的任务，当所有微任务队列执行完后便开始执行事件队列中的任务，事件任务执行完毕后再去执行微任务，如此循环往复，生生不息。

11、Dart 是如何实现多任务并行的？

Dart 是单线程的，不存在多线程，那如何进行多任务并行的呢？其实，Dart 的多线程和前端的多线程有很多的相似之处。Flutter 的多线程主要依赖 Dart 的并发编程、异步和事件驱动机制。



简单的说，在 Dart 中，一个 Isolate 对象其实就是一个 isolate 执行环境的引用，一般来说我们都是通过当前的 isolate 去控制其他的 isolate 完成彼此之间的交互，而当我们想要创建一个新的 Isolate 可以使用 Isolate.spawn 方法获取返回的一个新的 isolate 对象，两个 isolate 之间使用 SendPort 相互发送消息，而 isolate 中也存在了一个与之对应的 ReceivePort 接受消息用来处理，但是我们需要注意的是，ReceivePort 和 SendPort 在每个 isolate 都有一对，只有同一个 isolate 中的 ReceivePort 才能接受到当前类的 SendPort 发送的消息并且处理。

12、await for 如何使用?

await for 是不断获取 stream 流中的数据，然后执行循环体中的操作。它一般用在直到 stream 什么时候完成，并且必须等待传递完成之后才能使用，不然就会一直阻塞。

```
Stream<String> stream = new Stream<String>.fromIterable(
    ['不开心', '面试', '没', '过',]);

main() async {
  print('上午被开水烫了脚');
  await for (String s in stream) {
    print(s);
  }
  print('晚上还没吃饭');
}
```

13、Stream 有哪两种订阅模式？分别是怎么调用的？

单订阅(single)和多订阅（broadcast）。

单订阅就是只能有一个订阅者，而广播是可以有多个订阅者。这就有点类似于消息服务（Message Service）的处理模式。单订阅类似于点对点，在订阅者出现之前会持有数据，在订阅者出现之后就才转交给它。而广播类似于发布订阅模式，可以同时有多个订阅者，当有数据时就会传递给所有的订阅者，而不管

当前是否已有订阅者存在。 Stream 默认处于单订阅模式，所以同一个 stream 上的 listen 和其它大多数方法只能调用一次，调用第二次就会报错。但 Stream 可以通过 transform() 方法（返回另一个 Stream）进行连续调用。通过 Stream.asBroadcastStream() 可以将一个单订阅模式的 Stream 转换成一个多订阅模式的 Stream，isBroadcast 属性可以判断当前 Stream 所处的模式。

14、dart 中 mixin 机制是怎么样的？

mixin 是 Dart 2.1 加入的特性，以前版本通常使用 abstract class 代替。简单来说，mixin 是为了解决继承方面的问题而引入的机制，Dart 为了支持多重继承，引入了 mixin 关键字，它最大的特殊处在于：mixin 定义的类不能有构造方法，这样可以避免继承多个类而产生的父类构造方法冲突。 mixins 的对象是类，mixins 绝不是继承，也不是接口，而是一种全新的特性，可以 mixins 多个类，mixins 的使用需要满足一定条件。

15、JIT 与 AOT 分别是什么？

借助于先进的工具链和编译器，Dart 是少数同时支持 JIT（Just In Time，即时编译）和 AOT（Ahead of Time，运行前编译）的语言之一。那，到底什么是 JIT 和 AOT 呢？语言在运行之前通常都需要编译，JIT 和 AOT 则是最常见的两种编译模式。JIT 在运行时即时编译，在开发周期中使用，可以动态下发和执行代码，开发测试效率高，但运行速度和执行性能则会因为运行时即时编译受到影响。AOT 即提前编译，可以生成被直接执行的二进制代码，运行速度快、执行性能表现好，但每次执行前都需要提前编译，开发测试效率低。

16、Dart 的内存分配与垃圾回收是怎么样的？

Dart VM 的内存分配策略比较简单，创建对象时只需要在堆上移动指针，内存增长始终是线性的，省去了查找可用内存的过程。在 Dart 中，并发是通过 Isolate 实现的。Isolate 是类似于线程但不共享内存，独立运行的 worker。这样的机制，就可以让 Dart 实现无锁的快速分配。Dart 的垃圾回收，则是采用了多生代算法。新生代在回收内存时采用“半空间”机制，触发垃圾回收时，Dart 会将当前半空间中的“活跃”对象拷贝到备用空间，然后整体释放当前空间的所有内存。回收过程中，Dart 只需要操作少量的“活跃”对象，没有引用的大量“死亡”对象则被忽略，这样的回收机制很适合 Flutter 框架中大量 Widget 销毁重建的场景。

17、使用 mixins 的条件是什么？

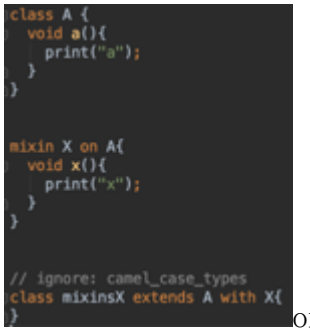
因为 mixins 使用的条件，随着 Dart 版本一直在变，这里讲的是 Dart2.1 中使用 mixins 的条件：

- mixins 类只能继承自 object mixins 类不能有构造函数
- 一个类可以 mixins 多个 mixins 类
- 可以 mixins 多个类，不破坏 Flutter 的单继承

18、mixin 怎么指定异常类型？

on 关键字可用于指定异常类型。 on 只能用于被 mixins 标记的类，例如 mixins X on A，意思是要 mixins X 的话，得先接口实现或者继承 A。这里 A 可以是类，也可以是接口，但是在 mixins 的时候用法有区别.

on 一个类：



on 的是一个接口： 得首先实现这个接口，然后再用 mix

19、main future mirotask 的执行顺序是怎样的?

普通代码都是同步执行的，结束后会开始检查 microtask 中是否有任务，若有则执行，执行完继续检查 microtask，直到 microtask 列队为空。最后会去执行 event 队列（future）。

20、Future 和 Isolate 有什么区别？

future 是异步编程，调用本身立即返回，并在稍后的某个时候执行完成时再获得返回结果。在普通代码中可以使用 await 等待一个异步调用结束。 isolate 是并发编程，Dart 有并发时的共享状态，所有 Dart 代码都在 isolate 中运行，包括最初的 main()。每个 isolate 都有它自己的堆内存，意味着其中所有内存数据，包括全局数据，都仅对该 isolate 可见，它们之间的通信只能通过传递消息的机制完成，消息则通过端口(port)收发。isolate 只是一个概念，具体取决于如何实现，比如在 Dart VM 中一个 isolate 可能会是一个线程，在 Web 中可能会是一个 Web Worker。

21、Stream 与 Future 是什么关系？

Stream 和 Future 是 Dart 异步处理的核心 API。Future 表示稍后获得的一个数据，所有异步的操作的返回值都用 Future 来表示。但是 Future 只能表示一次异步获得的数据。而 Stream 表示多次异步获得的数据。比如界面上的按钮可能会被用户点击多次，所以按钮上的点击事件（onClick）就是一个 Stream 。简单地说，Future 将返回一个值，而 Stream 将返回多次值。Dart 中统一使用 Stream 处理异步事件流。Stream 和一般的集合类似，都是一组数据，只不过一个是异步推送，一个是同步拉取。

Flutter

1、介绍下 Flutter 的 FrameWork 层和 Engine 层，以及它们的作用

Flutter 的 FrameWork 层是用 Dart 编写的框架（SDK），它实现了一套基础库，包含 Material（Android 风格 UI）和 Cupertino（iOS 风格）的 UI 界面，下面是通用的 Widgets（组件），之后是一些动画、绘制、渲染、手势库等。这个纯 Dart 实现的 SDK 被封装为了一个叫作 dart:ui 的 Dart 库。我们在使用 Flutter 写 App 的时候，直接导入这个库即可使用组件等功能。 Flutter 的 Engine 层是 Skia 2D 的绘图引擎库，其前身是一个向量绘图软件，Chrome 和 Android 均采用 Skia 作为绘图引擎。Skia 提供了非常友好的 API，并且在图形转换、文字渲染、位图渲染方面都提供了友好、高效的表现。Skia 是跨平台的，所以可以被嵌入到 Flutter 的 iOS SDK 中，而不用去研究 iOS 闭源的 Core Graphics / Core Animation。Android 自带了 Skia，所以 Flutter Android SDK 要比 iOS SDK 小很多。

2、介绍下 Widget、State、Context 概念

- Widget：在 Flutter 中，几乎所有东西都是 Widget。将一个 Widget 想象为一个可视化的组件（或与应用可视化方面交互的组件），当你需要构建与布局直接或间接相关的任何内容时，你正在使用 Widget。

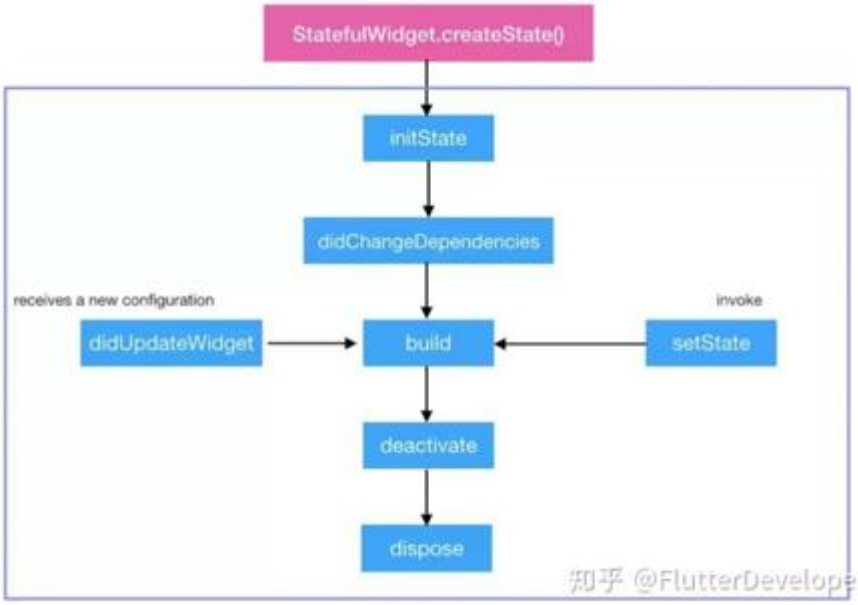
- Widget 树：Widget 以树结构进行组织。包含其他 Widget 的 widget 被称为父 Widget(或 widget 容器)。包含在父 widget 中的 widget 被称为子 Widget。
- Context：仅仅是已创建的所有 Widget 树结构中的某个 Widget 的位置引用。简而言之，将 context 作为 widget 树的一部分，其中 context 所对应的 widget 被添加到此树中。一个 context 只从属于一个 widget，它和 widget 一样是链接在一起的，并且会形成一个 context 树。
- State：定义了 StatefulWidget 实例的行为，它包含了用于”交互/干预“Widget 信息的行为和布局。应用于 State 的任何更改都会强制重建 Widget。

3、介绍下 StatelessWidget 和 StatefulWidget 两种状态组件类

- StatelessWidget: 一旦创建就不关心任何变化，在下次构建之前都不会改变。它们除了依赖于自身的配置信息（在父节点构建时提供）外不再依赖于任何其他信息。比如典型的 Text、Row、Column、Container 等，都是 StatelessWidget。它的生命周期相当简单：初始化、通过 build()渲染。
- StatefulWidget: 在生命周期内，该类 Widget 所持有的数据可能会发生变化，这样的数据被称为 State，这些拥有动态内部数据的 Widget 被称为 StatefulWidget。比如复选框、Button 等。State 会与 Context 相关联，并且此关联是永久性的，State 对象将永远不会改变其 Context，即使可以在树结构周围移动，也仍将与该 context 相关联。当 state 与 context 关联时，state 被视为已挂载。StatefulWidget 由两部分组成，在初始化时必须要在 createState()时初始化一个与之相关的 State 对象。

4、StatefulWidget 的生命周期是怎么样的？

Flutter 的 Widget 分为 StatelessWidget 和 StatefulWidget 两种。其中，StatelessWidget 是无状态的，StatefulWidget 是有状态的，因此实际使用时，更多的是 StatefulWidget。StatefulWidget 的生命周期如下图



- initState()：Widget 初始化当前 State，在当前方法中是不能获取到 Context 的，如想获取，可以试试 Future.delayed()
- didChangeDependencies()：在 initState() 后调用，State 对象依赖关系发生变化的时候也会调用。
- deactivate()：当 State 被暂时从视图树中移除时会调用这个方法，页面切换时也会调用该方法，和 Android 里的 onPause 差不多。
- dispose()：Widget 销毁时调用。

didUpdateWidget：Widget 状态发生变化的时候调用。

5、说下 Widgets、RenderObjects 和 Elements 的关系

首先看一下这几个对象的含义及作用。

- Widget ：仅用于存储渲染所需要的信息。
- RenderObject ：负责管理布局、绘制等操作。
- Element ：才是这颗巨大的控件树上的实体。

Widget 会被 inflate（填充）到 Element，并由 Element 管理底层渲染树。Widget 并不会直接管理状态及渲染,而是通过 State 这个对象来管理状态。Flutter 创建 Element 的可见树，相对于 Widget 来说，是可变的，通常界面开发中，我们不用直接操作 Element,而是由框架层实现内部逻辑。就如一个 UI 视图树中，可能包含有多个 TextWidget(Widget 被使用多次)，但是放在内部视图树的视角，这些 TextWidget 都是填充到一个个独立的 Element 中。Element 会持有 renderObject 和 widget 的实例。记住，Widget 只是一个配置，RenderObject 负责管理布局、绘制等操作。 在第一次创建 Widget 的时候，会对应创建一个 Element， 然后将该元素插入树中。如果之后 Widget 发生了

变化，则将其与旧的 Widget 进行比较，并且相应地更新 Element。重要的是，Element 不会被重建，只是更新而已。

6、Flutter 是如何与原生 Android、iOS 进行通信的？

Flutter 通过 PlatformChannel 与原生进行交互，其中 PlatformChannel 分为三种：

- BasicMessageChannel：用于传递字符串和半结构化的信息。
- MethodChannel：用于传递方法调用。Flutter 主动调用 Native 的方法，并获取相应的返回值。
- EventChannel：用于数据流（event streams）的通信。

关于原理：www.jianshu.com/p/39575a90e…

7、简述下 Flutter 的热重载

Flutter 的热重载是基于 JIT 编译模式的代码增量同步。由于 JIT 属于动态编译，能够将 Dart 代码编译成生成中间代码，让 Dart VM 在运行时解释执行，因此可以通过动态更新中间代码实现增量同步。

热重载的流程可以分为 5 步，包括：扫描工程改动、增量编译、推送更新、代码合并、Widget 重建。

Flutter 在接收到代码变更后，并不会让 App 重新启动执行，而只会触发 Widget 树的重新绘制，因此可以保持改动前的状态，大大缩短了从代码修改到看到修改产生的变化之间所需要的时间。

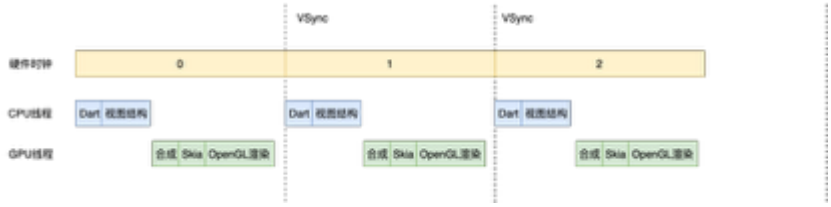
另一方面，由于涉及到状态的保存与恢复，涉及状态兼容与状态初始化的场景，热重载是无法支持的，如改动前后 Widget 状态无法兼容、全局变量与静态属性的更改、main 方法里的更改、initState 方法里的更改、枚举和泛型的更改等。

可以发现，热重载提高了调试 UI 的效率，非常适合写界面样式这样需要反复查看修改效果的场景。但由于其状态保存的机制所限，热重载本身也有一些无法支持的边界。

8、说下 Flutter 和其他跨平台方案的本质区别

React Native 之类的框架，只是通过 JavaScript 虚拟机扩展调用系统组件，由 Android 和 iOS 系统进行组件的渲染；

Flutter 则是自己完成了组件渲染的闭环。那么，Flutter 是怎么完成组件渲染的呢？这需要从图像显示的基本原理说起。在计算机系统中，图像的显示需要 CPU、GPU 和显示器一起配合完成：CPU 负责图像数据计算，GPU 负责图像数据渲染，而显示器则负责最终图像显示。CPU 把计算好的、需要显示的内容交给 GPU，由 GPU 完成渲染后放入帧缓冲区，随后视频控制器根据垂直同步信号（VSync）以每秒 60 次的速度，从帧缓冲区读取帧数据交由显示器完成图像显示。操作系统在呈现图像时遵循了这种机制，而 Flutter 作为跨平台开发框架也采用了这种底层方案。下面有一张更为详尽的示意图来解释 Flutter 的绘制原理。



Flutter 绘制原理可以看到，Flutter 关注如何尽可能快地在两个硬件时钟的 VSync 信号之间计算并合成视图数据，然后通过 Skia 交给 GPU 渲染：UI 线程使用 Dart 来构建视图结构数据，这些数据会在 GPU 线程进行图层合成，随后交给 Skia 引擎加工成 GPU 数据，而这些数据会通过 OpenGL 最终提供给 GPU 渲染。

9、Widget 唯一标识 Key 有哪几种？

在 flutter 中，每个 widget 都是被唯一标识的。这个唯一标识在 build 或 rendering 阶段由框架定义。该标识对应于可选的 Key 参数，如果省略，Flutter 将会自动生成一个。

在 flutter 中，主要有 4 种类型的 Key：GlobalKey（确保生成的 Key 在整个应用中唯一，是很昂贵的，允许 element 在树周围移动或变更父节点而不会丢失状态）、LocalKey、UniqueKey、ObjectKey。

10、什么是 Navigator? MaterialApp 做了什么？

Navigator 是在 Flutter 中负责管理维护页面堆栈的导航器。

MaterialApp 在需要的时候，会自动为我们创建 Navigator。

Navigator.of(context)，会使用 context 来向上遍历 Element 树，找到 MaterialApp 提供的_NavigatorState 再调用其 push/pop 方法完成导航操作。

11、Flutter 动画类型有哪些？

- 补间动画：给定初值与终值，系统自动补齐中间帧的动画
- 物理动画：遵循物理学定律的动画，实现了弹簧、阻尼、重力三种物理效果

在应用使用过程中常见动画模式：

- 动画列表或者网格：例如元素的添加或者删除操作；
- 转场动画 Shared element transition：例如从当前页面打开另一页面的过渡动画；
- 交错动画 Staggered animations：比如部分或者完全交错的动画。

12、Flutter 是怎么完成组件渲染的?

A:

13、Flutter 绘制流程是怎么样的？

A:

14、如何统一管理错误页面？

在 main 方法修改 ErrorWidget.builder 来自定义一个属于自己的 Widget；

如：

```
  /// 自定义报错页面
  ErrorWidget.builder = (FlutterErrorDetails flutterErrorDetails) {
    debugPrint(flutterErrorDetails.toString());
    return new Center(child: new Text("App 错误，快去反馈给作者!"));
  };
```

复制代码

15、Flutter 中存在哪四大线程？

Flutter 中存在四大线程，分别为 UI Runner、GPU Runner、IO Runner，Platform Runner （原生主线程），同时在 Flutter 中可以通过 isolate 或者 compute 执行真正的跨线程异步操作。

16、PlatformView 的作用有哪些？

Flutter 中通过 PlatformView 可以嵌套原生 View 到 Flutter UI 中；

17、PlatformView 使用了哪些东西来实现？

Presentation、VirtualDisplay 、 Surface 等；

18、PlatformView 大致原理是怎么样的？

使用了类似副屏显示的技术，VirtualDisplay 类代表一个虚拟显示器，调用 DisplayManager 的 createVirtualDisplay() 方法，将虚拟显示器的内容渲染在一个 Surface 控件上，然后将 Surface 的 id 通知给 Dart，让 engine 绘制时，在内存中找到对应的 Surface 画面内存数据，然后绘制出来。 实时控件截图渲染显示技术。

19、Flutter 的 Debug 和 release 分别是在什么模式下运行的？

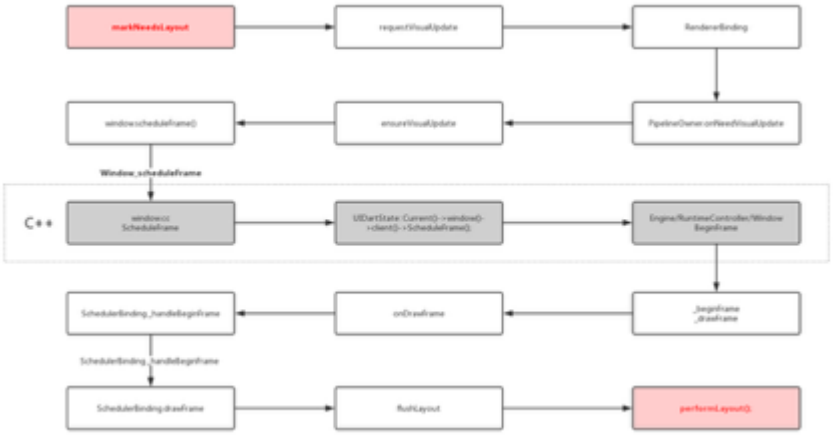
Flutter 的 Debug 下是 JIT 模式，release 下是 AOT 模式。

20、Platform Channel 有哪几种通信方式？分别是用于什么操作？

- BasicMessageChannel ：用于传递字符串和半结构化的信息。
- MethodChannel ：用于传递方法调用（method invocation）。
- EventChannel: 用于数据流（event streams）的通信。

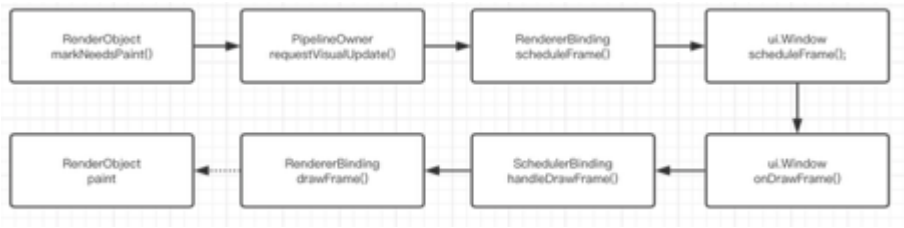
21、RenderObject 布局相关方法调用顺序是怎么样的？

layout -> performResize -> performLayout -> markNeedsPaint ，但是用户一般不会直接调用 layout，而是通过 markNeedsLayout ，具体流程如下：



22、RenderObject 如何使得页面重绘？流程是怎么样的？

RenderObject 在 attch/layout 之后会通过 markNeedsPaint(); 使得页面重绘，流程大概如下：



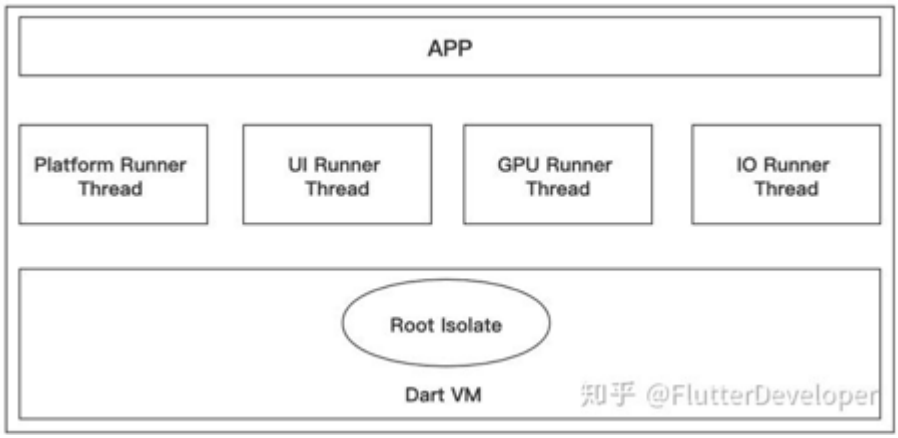
通过 isRepaintBoundary 往上确定了更新区域，通过 requestVisualUpdate 方法触发更新往下绘制。

23、Flutter 存在哪几棵树？他们有什么关系？

Flutter 中存在 Widget 、 Element 、RenderObject 、Layer 四棵树，其中 Widget 与 Element 是一对多的关系 ，

24、简述下 Flutter 的线程管理模型

默认情况下，Flutter Engine 层会创建一个 Isolate，并且 Dart 代码默认就运行在这个主 Isolate 上。必要时可以使用 spawnUri 和 spawn 两种方式来创建新的 Isolate，在 Flutter 中，新创建的 Isolate 由 Flutter 进行统一的管理。事实上，Flutter Engine 自己不创建和管理线程，Flutter Engine 线程的创建和管理是 Embeder 负责的，Embeder 指的是将引擎移植到平台的中间层代码，Flutter Engine 层的架构示意图如下图所示。



在 Flutter 的架构中，Embeder 提供四个 Task Runner，分别是 Platform Task Runner、UI Task Runner Thread、GPU Task Runner 和 IO Task Runner，每个 Task Runner 负责不同的任务，Flutter Engine 不在乎 Task Runner 运行在哪个线程，但是它需要线程在整个生命周期里面保持稳定

状态管理 【来自老友：i 校长】

1、状态管理是什么？

程序=算法+数据结构 数据是程序的中心。数据结构和算法两个概念间的逻辑关系贯穿了整个程序世界，首先二者表现为不可分割的关系。其实 Flutter 不就是一个程序吗，那我们面临的最底层的问题还是算法和数据结构，所以我们推导出

Flutter=算法+数据结构 那状态管理是什么？我也用公式来表达一下，如下：

Flutter 状态管理=算法+数据结构+UI 绑定

2、为什么需要状态管理？

用于解决状态更新问题，不需要 WidgetState 被全局化，保证组件隐私，使得代码可扩展，易维护，可以动态替换 UI 而不影响算法逻辑，安全可靠，保持数据的稳定伸缩，性能佳，局部优化；

3、说下状态管理基本分类

分为局部管理和全局管理；

- 局部管理：短暂的状态，这种状态根本不需要做全局处理；
- 全局管理：即应用状态，非短暂状态，您要在应用程序的许多部分之间共享，以及希望在用户会话之间保持的状态，就是我们所说的应用程序状态（有时也称为共享状态）

4、状态管理的底层逻辑一般是怎么样的？

- State：如 StatefulWidget、StreamBuilder 状态管理方式；
- InheritedWidget 专门负责 Widget 树中数据共享的功能型 Widget：如 Provider、scoped_model 就是基于它开发；

- Notification：与 InheritedWidget 正好相反，InheritedWidget 是从上往下传递数据，Notification 是从下往上，但两者都在自己的 Widget 树中传递，无法跨越树传递；
- Stream 数据流 ：如 Bloc、flutter_redux、fish_redux 等也都基于它来做实现；

5、状态管理的使用原则是怎么样的？

局部管理优于全局、保持数据安全性、考虑页面重新 build 带来的影响；

6、使用成熟状态管理库的弊端有哪些？

增加代码复杂性、框架 bug 修复需要时间等待、不理解框架原理导致使用方式不对，反而带来更多问题、选型错误导致不符合应用要求、与团队风格冲突不适用；

进阶

1、flutter run 实际走了哪三个命令？分别用于什么操作？

- flutter build apk：通过 gradle 来构建 APK
- adb install：安装 APK
- adb am start：启动应用

2、Flutter 引擎启动过程中做了什么操作？

3、setState 做了哪些工作？是如何更新 UI 的？

setState 其实是调用了 markNeedsBuild ，该方法内部标记此 Element 为 Dirty ，然后在下一帧 WidgetsBinding.drawFrame 才会被绘制， setState 并不是立即生效的。

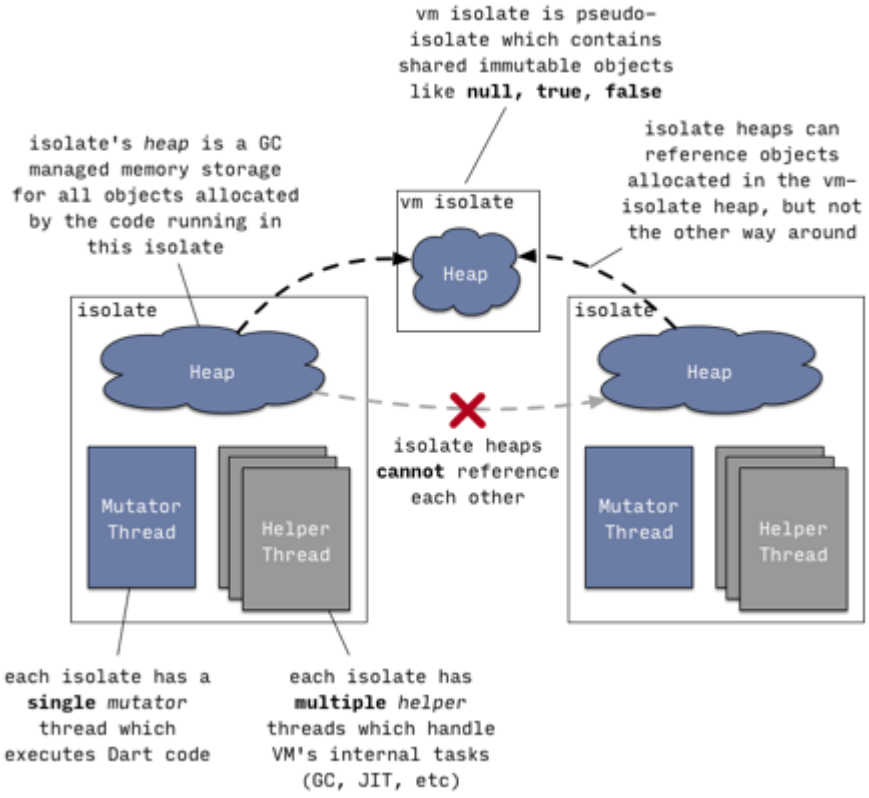
4、Flutter 应用启动 runApp(MyApp)过程是怎么样的？

Flutter 中 runApp 启动入口其实是一个 WidgetsFlutterBinding ，它主要是通过 BindingBase 的子类 GestureBinding 、ServicesBinding 、SchedulerBinding 、PaintingBinding 、SemanticsBinding 、RendererBinding 、WidgetsBinding 等，通过 mixins 的组合而成的。

5.Dart 虚拟机如何管理的？怎么调用？如何跟 Flutter 引擎交互？

Dart 虚拟机拥有自己的 Isolate，完全由虚拟机自己管理的，Flutter 引擎也无法直接访问。Dart 的 UI 相关操作，是由 Root Isolate 通过 Dart 的 C++调用，或者是发送消息通知的方式，将 UI 渲染相关的任务提交到 UIRunner 执行，这样就可以跟 Flutter 引擎相关模块进行交互。

6、Isolate 组成部分有哪些？分别有什么作用？



- isolate 堆是运该 isolate 中代码分配的所有对象的 GC 管理的内存存储；
- vm isolate 是一个伪 isolate，里面包含不可变对象，比如 null，true，false；
- isolate 堆能引用 vm isolate 堆中的对象，但 vm isolate 不能引用 isolate 堆；
- isolate 彼此之间不能相互引用 每个 isolate 都有一个执行 dart 代码的 Mutator thread，一个处理虚拟机内部任务(比如 GC，JIT 等)的 helper thread；

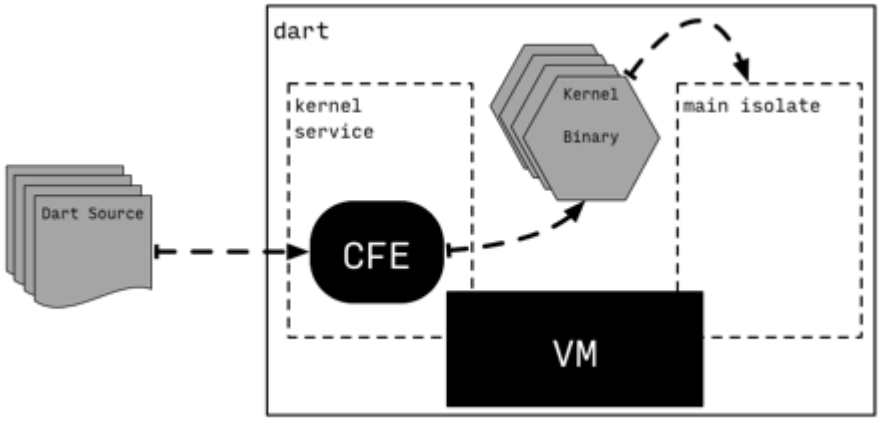
7、线程和 isolate 的关系是什么？

1、同一个线程在同一时间只能进入一个 isolate，当需要进入另一个 isolate 则必须先退出当前的 isolate；

2、一次只能有一个 Mutator 线程关联对应的 isolate，Mutator 线程是执行 Dart 代码并使用虚拟机的公共的 C 语言 API 的线程；

8、介绍下 JIT 运行模式中 kernel service

是一个辅助类 isolate，其核心工作就是 CFE，将 dart 转为 Kernel 二进制，然后 VM 可直接使用 Kernel 二进制运行在主 isolate 里面运行。



9、介绍下 Dart 虚拟机中通过 Snapshots 运行

A：

10、介绍下 Dart 虚拟机中通过 AppAOT Snapshots 运行

A：

11、图片加载流程是怎么样的？

A：

12、简单的说下 GestureDector 底层实现

A：

13、setState 在哪种场景下可能会失效？

1、刷新方法内声明的变量；

2、刷新被 final 修饰的变量；

14、isolate 是怎么进行通信的？实例化过程是怎么样的？

isolate 线程之间的通信主要通过 port 来进行，这个 port 消息传递过程是异步的。

实例化一个 isolate 的过程包括：

- 1.实例化 isolate 结构体。
- 2.在堆中分配线程内存。
- 3.配置 port 等过程。

15、虚拟机如何运行 Dart 代码？

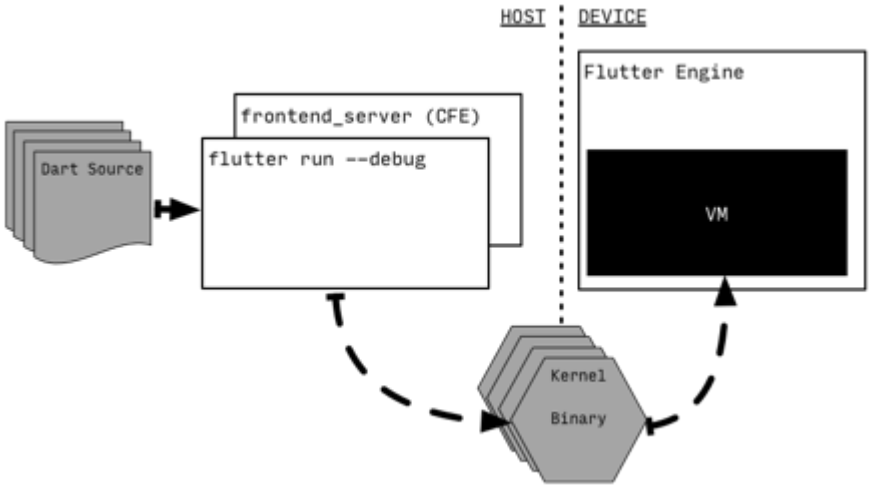
1.源码或者 Kernel 二进制(JIT)

2.snapshot ：

- AOT snapshot
- AppJIT snapshot

16、JIT 运行模式中 debug 运行原理是怎么样的？

将 dart 代码转换为 kernel 二进制和执行 kernel 二进制，这两个过程也可以分离开来，在两个不同的机器执行，比如 host 机器执行编译，移动设备执行 kernel 文件。



17.默认情况下 debug 和 release 会生成哪些架构的 so 库？

图解：

这个编译过程并不是 flutter tools 自身完成，而是交给另一个进程 frontend_server 来执行，它包括 CFE 和一些 flutter 专有的 kernel 转换器。hot reload：热重载机制正是依赖这一点，frontend_server 重用上一次编译中的 CFE 状态，只重新编译实际更改的部分。