flutter key的基本使用

DaZenD

flutter中万物皆组件，组件皆有key，flutter开发是组件堆砌的，组件很多，能复用就复用，如果没有一个标记，flutter做diff算法复用element的时候很容易数据错乱，所以，key根本作用在这

key能解决大部分的问题，flutter能分清key对应的元素。但是别指望能解决所有的问题。下面逐个解析，综合应用key解决所有问题

比如：

center(container(text("", key))) 这种结构，使用和不使用center的情况下内部的text就复用不了了

Widget和ELement

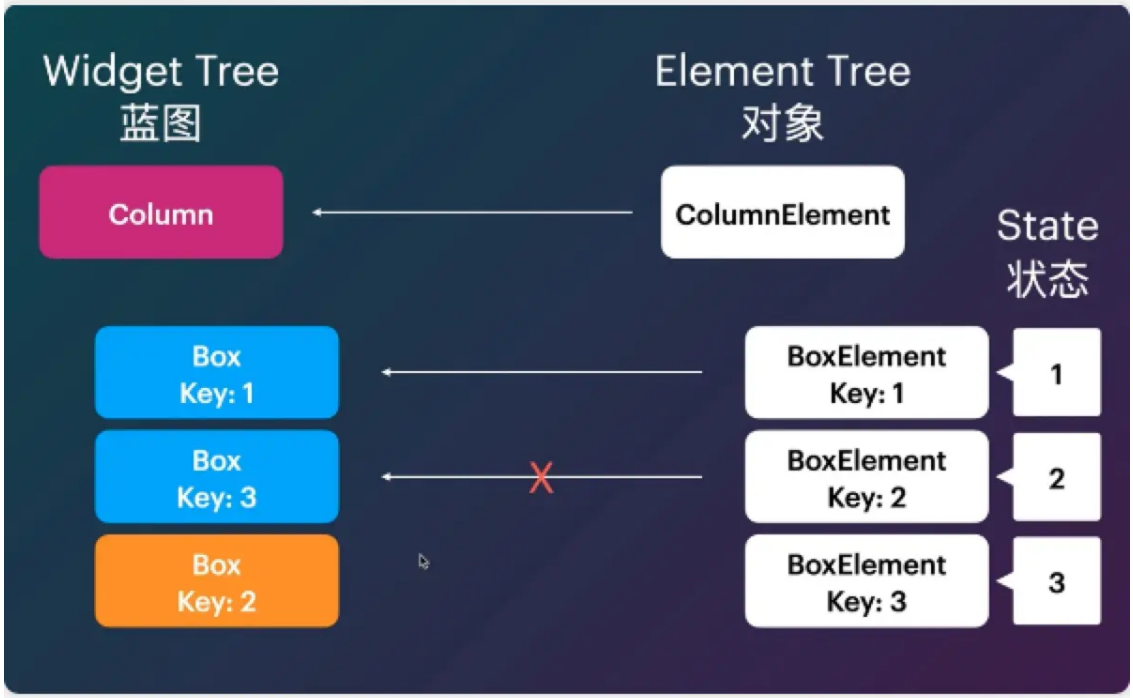
widget是虚拟的，并不是实际渲染的那个东西

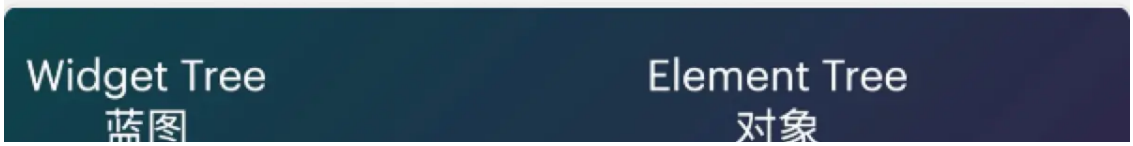
flutter中有widget tree，element tree，renderobject tree。widget tree可以理解为蓝图，一个布局页面的工具，element tree管理状态，上能访问widget tree，下能关联renderobject tree。

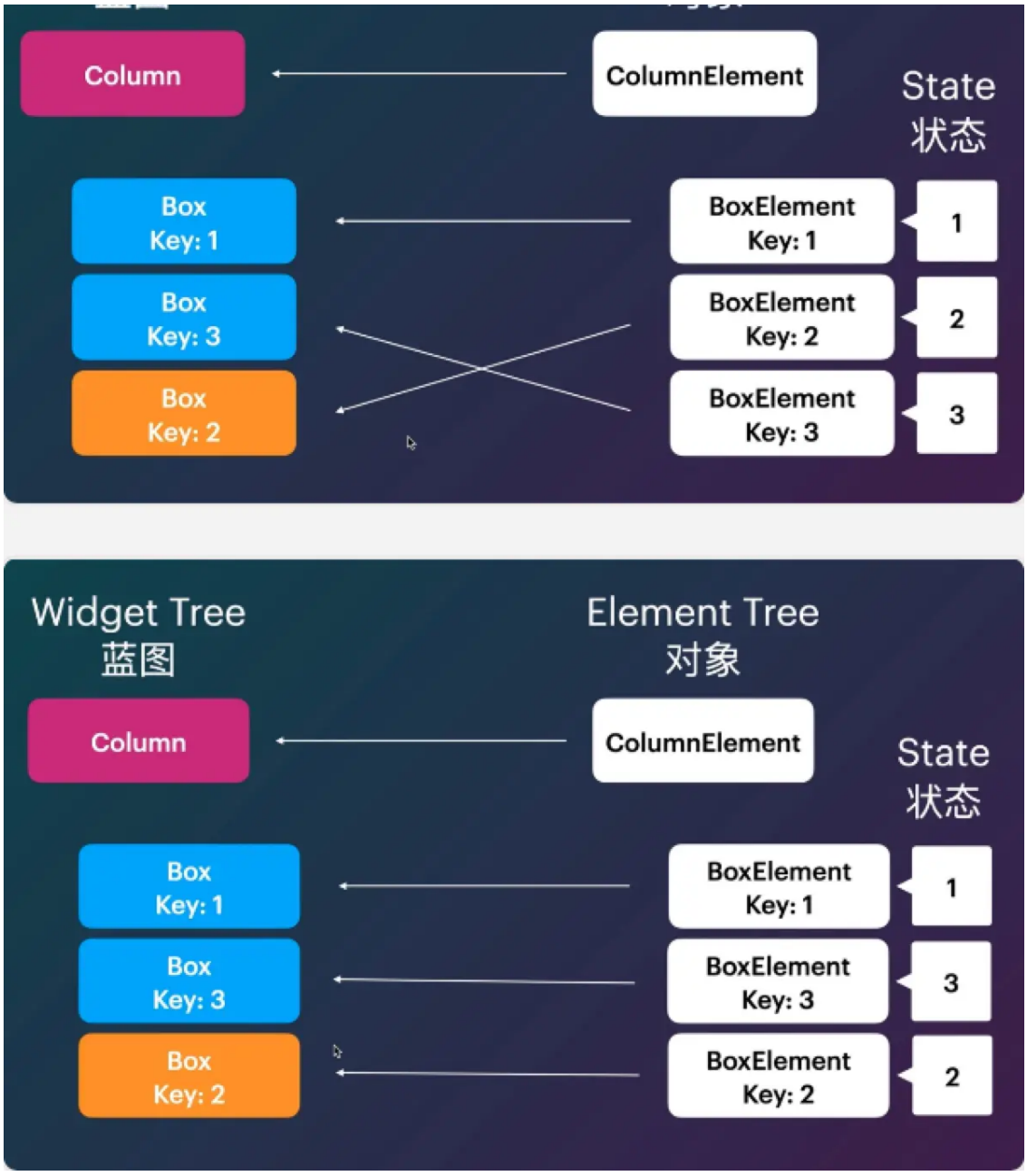
diff算法：flutter需要刷新页面布局的时候，会从要刷新的Element tree第一级逐级往下对比：widget tree 和 element tree对比，判断类型是否改变 & 判断key是否一致。如果有一个条件不满足，就在当前widget tree同级别组件中找同类

型同 key 的组件关联


Widget Tree
蓝图
Element Tree
对象
Column
ColumnElement
State
状态
Box
Key: 1
Box
Key: 2
Box
Key: 3
BoxElement
Key: 1
1
BoxElement
Key: 2
2
BoxElement
Key: 3
3


Widget Tree
蓝图
Element Tree
对象
Column
ColumnElement
State
状态
Box
Key: 1
Box
Key: 3
Box
Key: 2
BoxElement
Key: 1
1
BoxElement
Key: 2
2
BoxElement
Key: 3
3


Widget Tree
蓝图
Element Tree
对象



widget-element-key.png

针对 element tree 在 widget tree 关联不到的，会销毁实例。

widget tree 中在 element tree 中没有的，element tree 会新建并关联上

懂了这个复用流程后，上面说的那个有没有 center 的情况就能解释了。

局部键

ValueKey

flutter对比key的规则是值是否相等，这个可以自己重写operator

```
class MyValueKey {
  String id;
  String key;

  MyValueKey(this.id, this.key);

  @override
  bool operator ==(Object other) =>
      identical(this, other) ||
      other is MyValueKey &&
          runtimeType == other.runtimeType &&
          id == other.id &&
          key == other.key;

  @override
  int get hashCode => id.hashCode ^ key.hashCode;
}
```

generate快速生成

ObjectKey

key的对比规则是对比key的内存地址，是否是同一个obj，而不是单单看值。跟java中的equals和==的区别差不多。

```
/// Check whether two references are to the same object.
///
/// Example:
```

```
/// ```dart
/// var o = new Object();
/// var isIdentical = identical(o, new Object()); // false,
different objects.
/// isIdentical = identical(o, o); // true, same object
/// isIdentical = identical(const Object(), const
Object()); // true, const canonicalizes
/// isIdentical = identical([1], [1]); // false
/// isIdentical = identical(const [1], const [1]); // true
/// isIdentical = identical(const [1], const [2]); // false
/// isIdentical = identical(2, 1 + 1); // true, integers
canonicalizes
/// ```
external bool identical(Object? a, Object? b);
```

UniqueKey

组件每次刷新时候，UniqueKey都会是新的，这种key用的不多，可以理解一下下面的代码场景：

```
    return Center(
      child: AnimatedSwitcher(
        duration: const Duration(seconds: 1),
        child: Text("content", key: UniqueKey(),),
      ),
    );
```

这种应用，当content有变化的时候会有动画过渡。每次content有变动触发刷新，uniqueKey都不一样，text就会新建

全局键

上面有说当使用局部键的时候，如果widget tree层级有变

化，那么状态就会丢失，但是这种情况怎么解决呢?

可以使用 GlobalKey，跟普通的 key 使用一样

注意：

1、globalKey 在 app 中是唯一的，一个 GlobalKey 实例只能给一个组件使用。所以，需要几个实例化几个

根据 globalKey 找组件

前端：document.getElementById

iOS：getViewWithTag

android：findViewById

应用都差不多

```
_globalKey.currentState as yourWidgetState

_globalKey.currentWidget as yourWidget

_globalKey.currentContext as RenderBox
```