LeanCode

WTF is BuildContext?

So how exactly a widget lands on a screen?

Widget

“Describes the configuration for an Element.”

```
@protected
@factory
Element createElement();
```

We build digital products.

LeanCode

StatelessWidget

“A widget that does not require mutable state”

```
@override
StatelessElement createElement() => StatelessElement(this);
```

We build digital products.

LeanCode

StatefulWidget

“A widget that has mutable state”

```
@override
StatefulElement createElement() => StatefulElement(this);
```

We build digital products.

LeanCode

Widgets are immutable

Widgets can’t hold any reference to the tree

We build digital products.

LeanCode

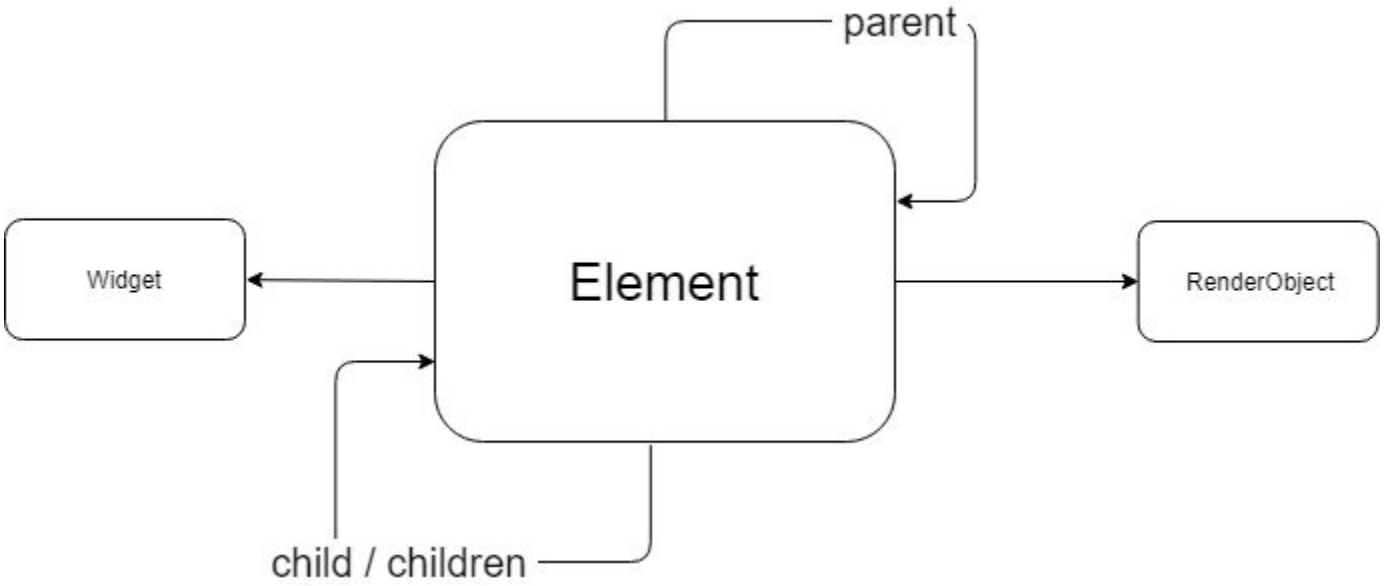
Same Widget can be used in multiple subtrees

We build digital products.

LeanCode

Element

“An instantiation of a Widget at a particular location in the tree.”



Source: didierboelens.com

We build digital products.
LeanCode

Element

Element always has a RenderObject at or below its location.

```
RenderObject? get renderObject {
  RenderObject? result;
  void visit(Element element) {
    assert(result == null); // this verifies that there's only one child
    if (element is RenderObjectElement)
      result = element.renderObject;
    else
      element.visitChildren(visit);
  }
  visit(this);
  return result;
}
```

We build digital products.

LeanCode

Element could either be:
ComponentElement
RenderObjectElement

We build digital products.

LeanCode

StatelessElement

```
@override
Widget build() => widget.build(this);
```

We build digital products.

LeanCode

StatefulElement

```
@override
Widget build() => state.build(this);
```

RenderObject

```
@protected
void performLayout();
```

```
void paint(PaintingContext context, Offset offset) { }
```

We build digital products.

LeanCode

RenderObject

```dart
@override
void performLayout() {
  final BoxConstraints constraints = this.constraints;
  _resolve();
  assert(_resolvedPadding != null);
  if (child == null) {
    size = constraints.constrain(Size(
      _resolvedPadding!.left + _resolvedPadding!.right,
      _resolvedPadding!.top + _resolvedPadding!.bottom,
    ));
    return;
  }
  final BoxConstraints innerConstraints = constraints.deflate(_resolvedPadding!);
  child!.layout(innerConstraints, parentUsesSize: true);
  final BoxParentData childParentData = child!.parentData! as BoxParentData;
  childParentData.offset = Offset(_resolvedPadding!.left, _resolvedPadding!.top);
  size = constraints.constrain(Size(
    _resolvedPadding!.left + child!.size.width + _resolvedPadding!.right,
    _resolvedPadding!.top + child!.size.height + _resolvedPadding!.bottom,
  ));
}
```

We build digital products.

LeanCode

InheritedWidget

```
T? dependOnInheritedWidgetOfExactType<T extends InheritedWidget>(
        {Object? aspect}
)
```

We build digital products.

LeanCode

InheritedWidget - Theme example

```
static ThemeData of(BuildContext context) {
  final _InheritedTheme? inheritedTheme = context.dependOnInheritedWidgetOfE
  final MaterialLocalizations? localizations = Localizations.of<MaterialLoca
  final ScriptCategory category = localizations?.scriptCategory ?? ScriptCat
  final ThemeData theme = inheritedTheme?.theme.data ?? _kFallbackTheme;
  return ThemeData.localize(theme, theme.typography.geometryThemeFor(categor
}
```

We build digital products

LeanCode

InheritedWidget - actually it’s not like that

```dart
@override
T? dependOnInheritedWidgetOfExactType<T extends InheritedWidget>({Object? aspect}) {
  assert(_debugCheckStateIsActiveForAncestorLookup());
  final InheritedElement? ancestor = _inheritedWidgets == null ? null : _inheritedWidgets![T];
  if (ancestor != null) {
    return dependOnInheritedElement(ancestor, aspect: aspect) as T;
  }
  _hadUnsatisfiedDependencies = true;
  return null;
}
```

Provider

```dart
class MyWidget extends StatelessWidget {
  const MyWidget({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Provider<int>.value(
      value: 5,
      child: Column(
        children: [
          Text('number is ${Provider.of<int>(context)}'),
          Text('number is ${context.watch<int>()}'),
        ],
      ), // Column
    ); // Provider.value
  }
}
```

We build digital products.

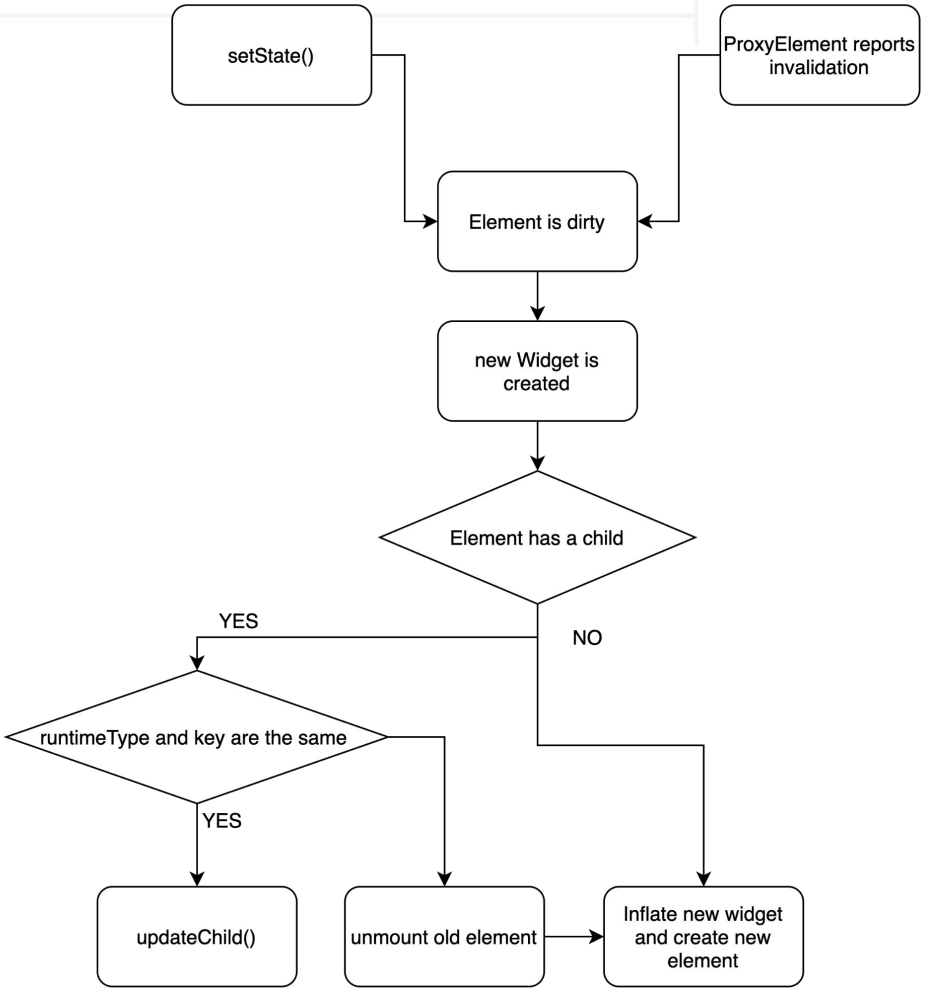
LeanCode

So how it works?

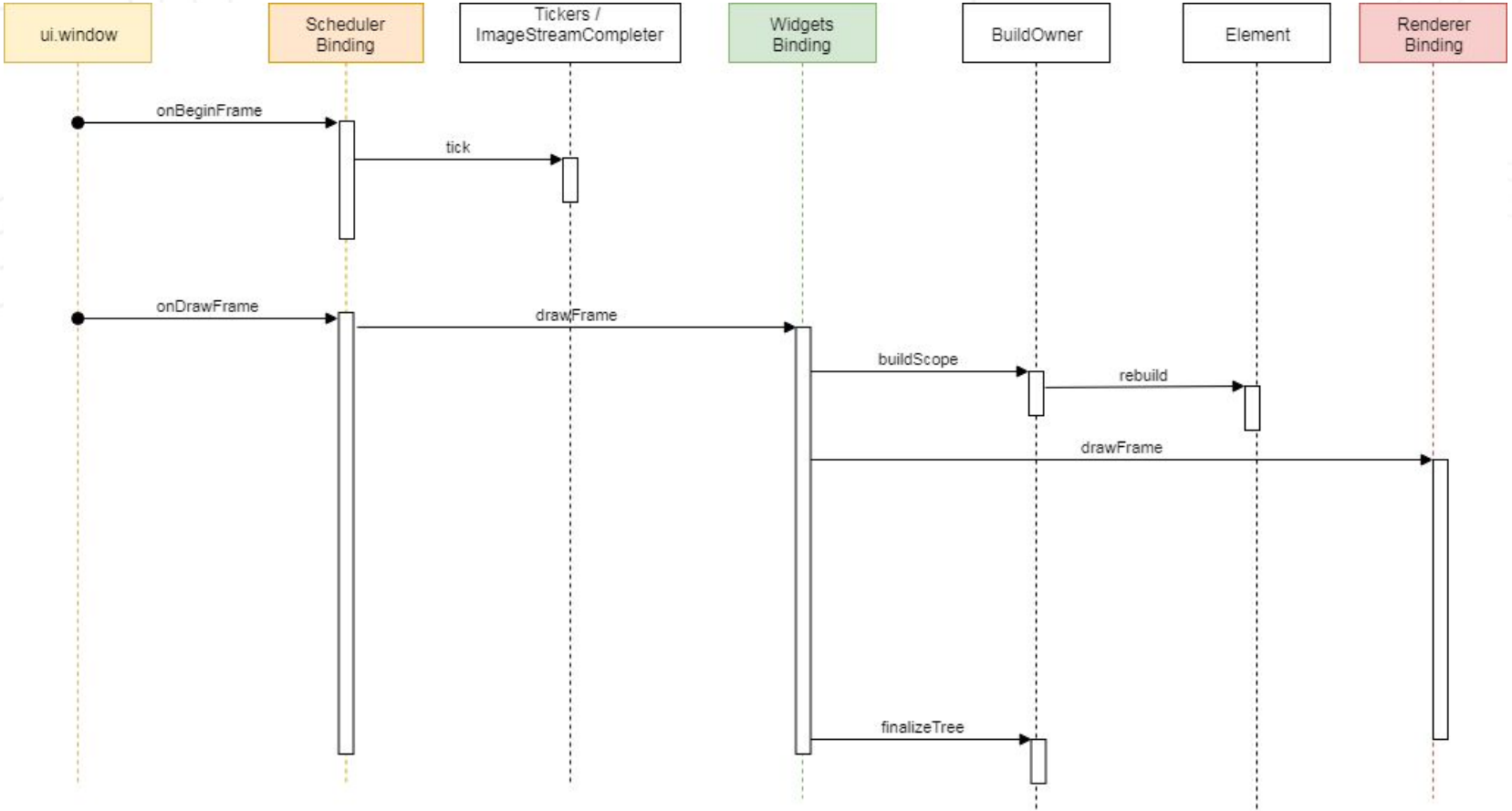
```dart
@protected
void setState(VoidCallback fn) {
  final Object? result = fn() as dynamic;
  _element!.markNeedsBuild();
}
```

We build digital products.

LeanCode



We build digital products.
LeanCode



Source: didierboelens.com

We build digital products.
LeanCode

BuildContext is just an Element

Thanks