

ABOUT YOU° TECH

A Very New Way of Developing

1.Moving Away from React Native

From React Native to Flutter

2.Fixing our State Management

Our interpretation of the BLoC pattern

3.Golden File Testing with Flutter

Our approach, the benefits, and challenges to Golden File testing

4.Developing Interactions in Flutter

How we created user-driven animations for our new ADP

1.Moving Away from React Native

From React Native to Flutter

2.Fixing our State Management

Our interpretation of the BLoC pattern

3.Golden File Testing with Flutter

Our approach, the benefits, and challenges to Golden File testing

4.Developing Interactions in Flutter

How we created user-driven animations for our new ADP

TypeScript

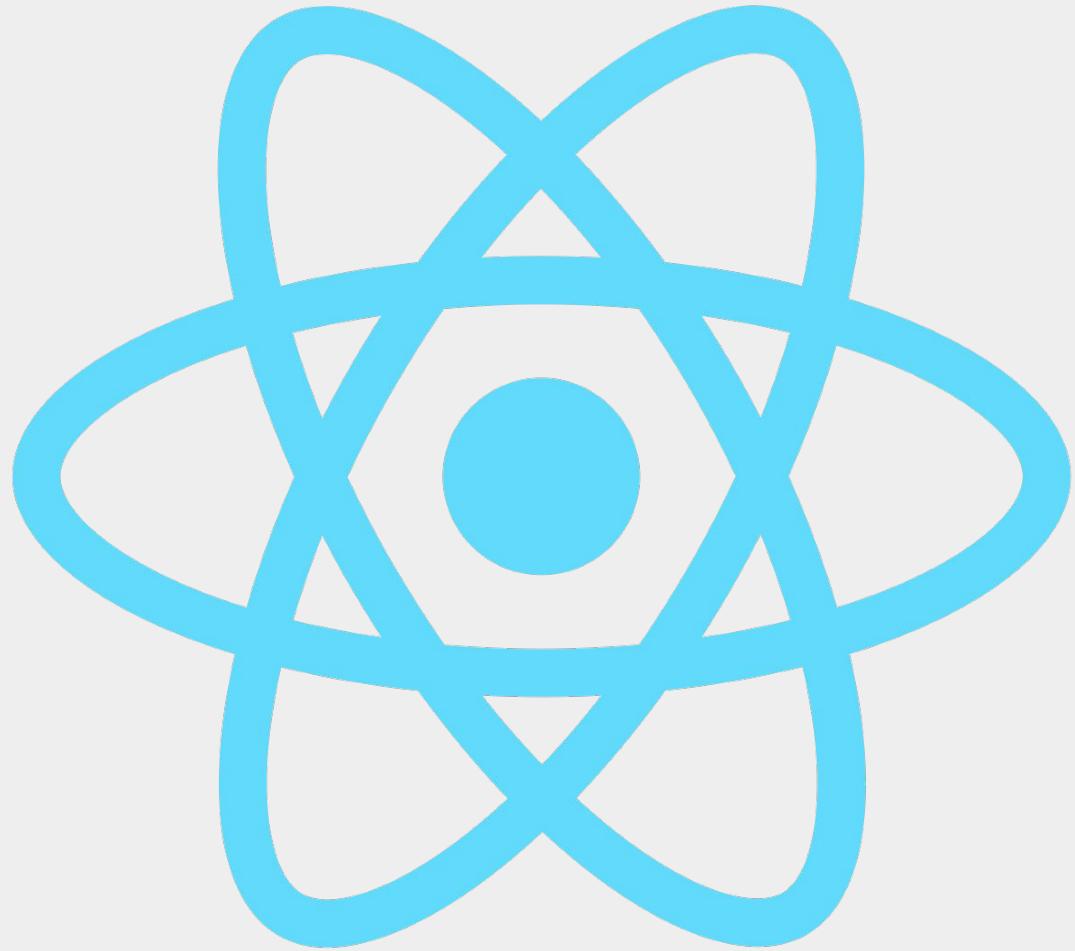


Redux



React Native at AY

- Development in 2017
- Powers current App (4.6 Stars)
- TypeScript
- Redux
- Redux Saga (Side Effect management)
- First “modern” stack with Typescript (Anyscript) and React



React Native at AY - The Good

- Staying in the Web Ecosystem
 - Typescript
 - Eased in types into our code base
 - Helped a lot with the growing codebase
 - React
 - Components
 - Easier Lifecycles
 - Prettier / tslint
 - Takes Care of formatting
- Fast Development Cycles at the beginning

1. Developer Experience

Debug Mode and Hot Reload, State Management, Upgrades

2. Performance

Rerenders and the JS Bridge

3. Rewrite vs. Refactoring

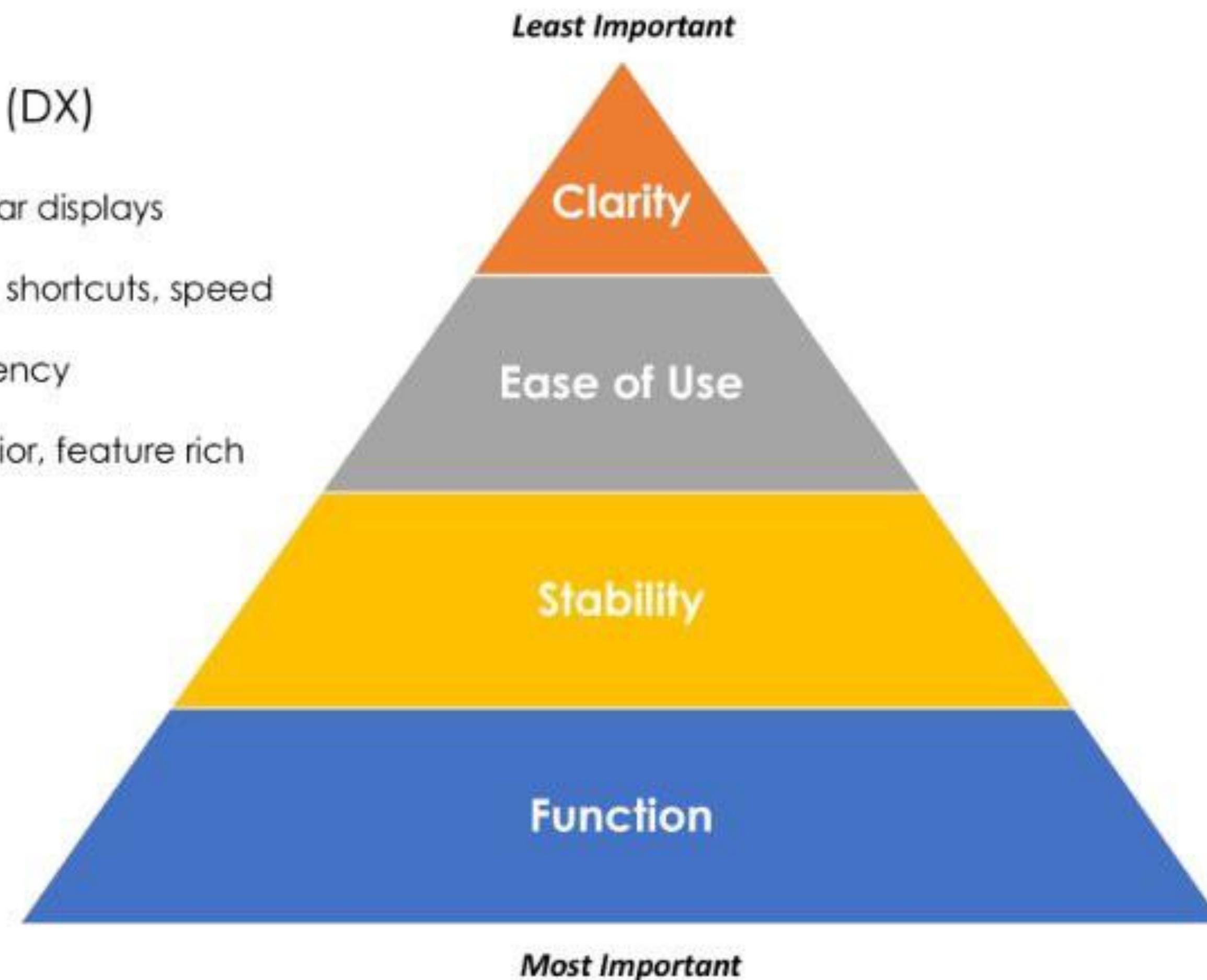
Taking the Leap

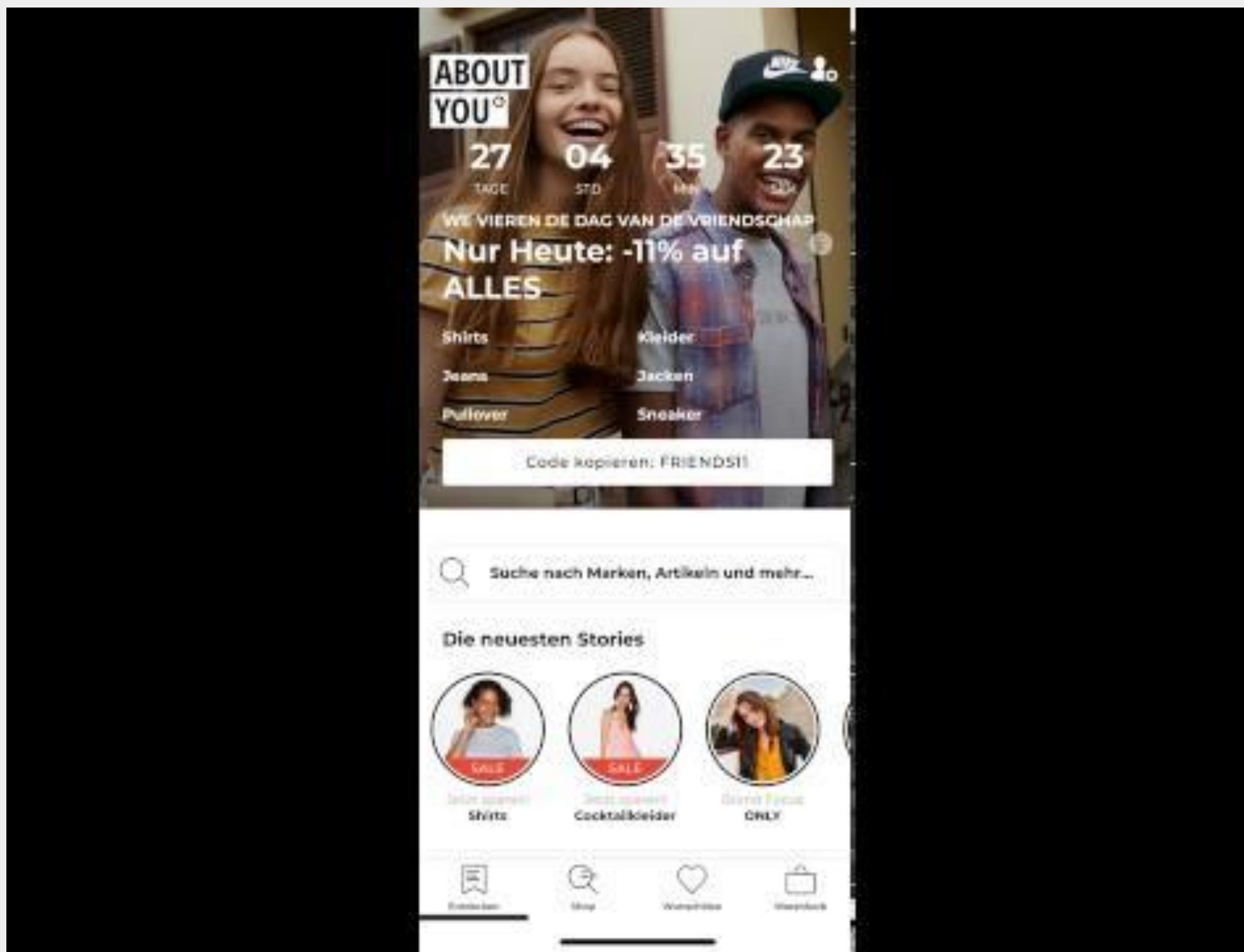
4. What makes Flutter different?

Developer Experience and Performance

Developer Experience (DX)

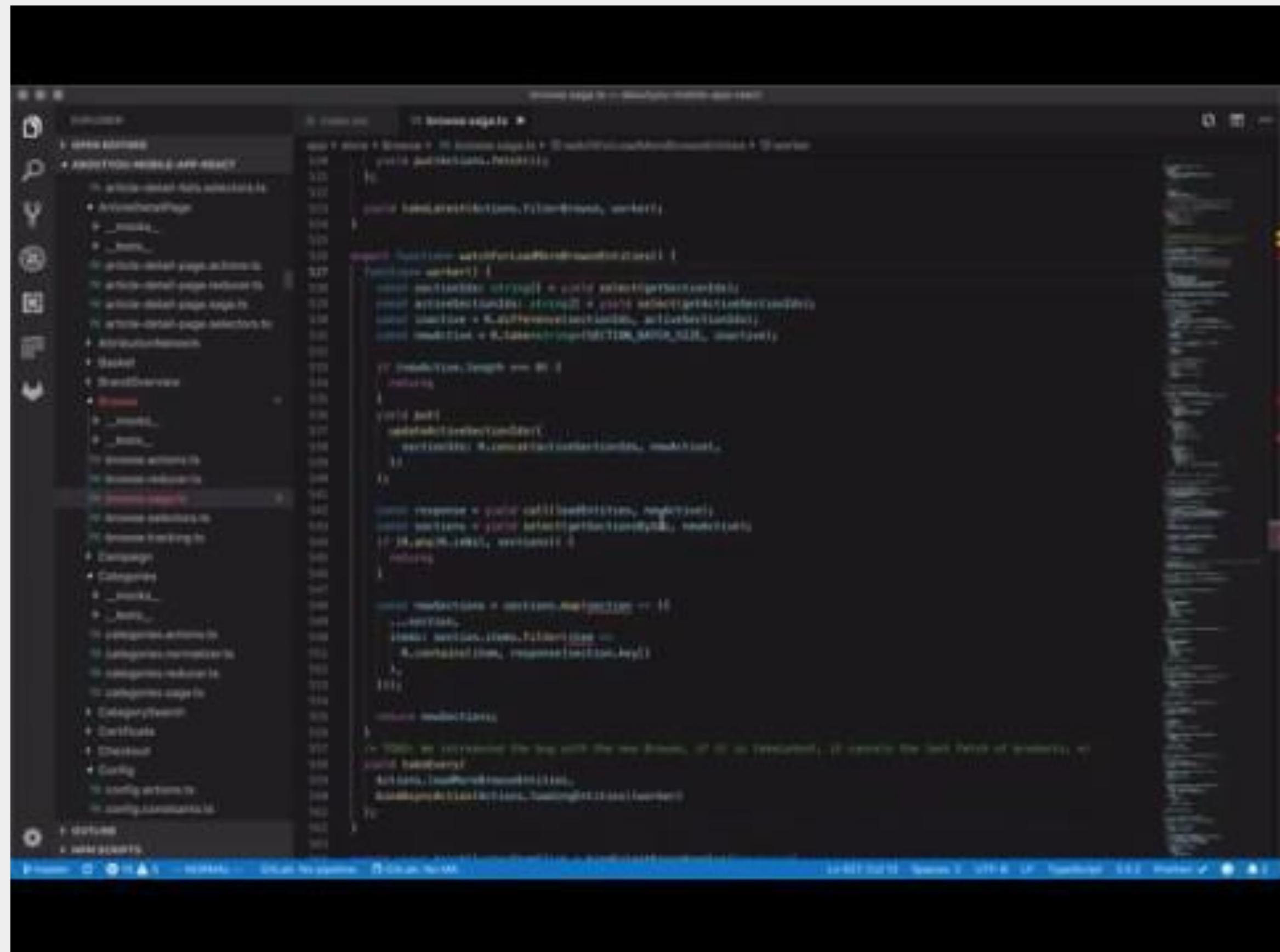
- Clarity – Intuitive visuals, clear displays
- Ease of Use – Quick access, shortcuts, speed
- Stability – Reliability, consistency
- Function – Expected behavior, feature rich





Debug Mode and Hot Reload

- Really good for smaller project
- Started to get painfully slow as the project grew
- Hot Reload started to reset state (react-native-navigation)
- Webpack performance degrades over time
- Breakpoints not working most of the time
- Developers need to wait up to 10 minutes for their build to test



State Management - Code Navigation

- Too much reliance on the global state
 - Every action is handled via sagas
 - Almost no local component state
 - Tons and Tons of boilerplate
 - Too much complexity which makes for a rough onboarding

```
526 export function* watchForLoadMoreBrowseEntities() {
527   function* worker() {
528     const sectionIds: string[] = yield select(getSectionIds);
529     const activeSectionIds: string[] = yield select(getActiveSectionIds);
530     const inactive = R.difference(sectionIds, activeSectionIds);
531     const newActive = R.take<string>(SECTION_BATCH_SIZE, inactive);
532
533     if (newActive.length === 0) {
534       return;
535     }
536     yield put(
537       updateActiveSectionIds({
538         se const response = yield call(loadEntities, newActive);
539       }) const sections = yield select(getSectionsByIds, newActive);
540     );
541     const response: any
542     const response = yield call(loadEntities, newActive);
543     const sections = yield select(getSectionsByIds, newActive);
544     if (R.any(R.isNil, sections)) {
545       return;
546     }
547
548     const newSections = sections.map(section => ({
549       ...section,
550       items: section.items.filter(item =>
551         R.contains(item, response[section.key])
552       ),
553     }));
554 }
```

State Management

- Didn't take into account redux-saga type safety in the beginning (Anyscript)
- Types are lost in Redux-Sagas generators
- Went as far as developing tsaga, which is a typed alternative



Upgrading React Native and packages

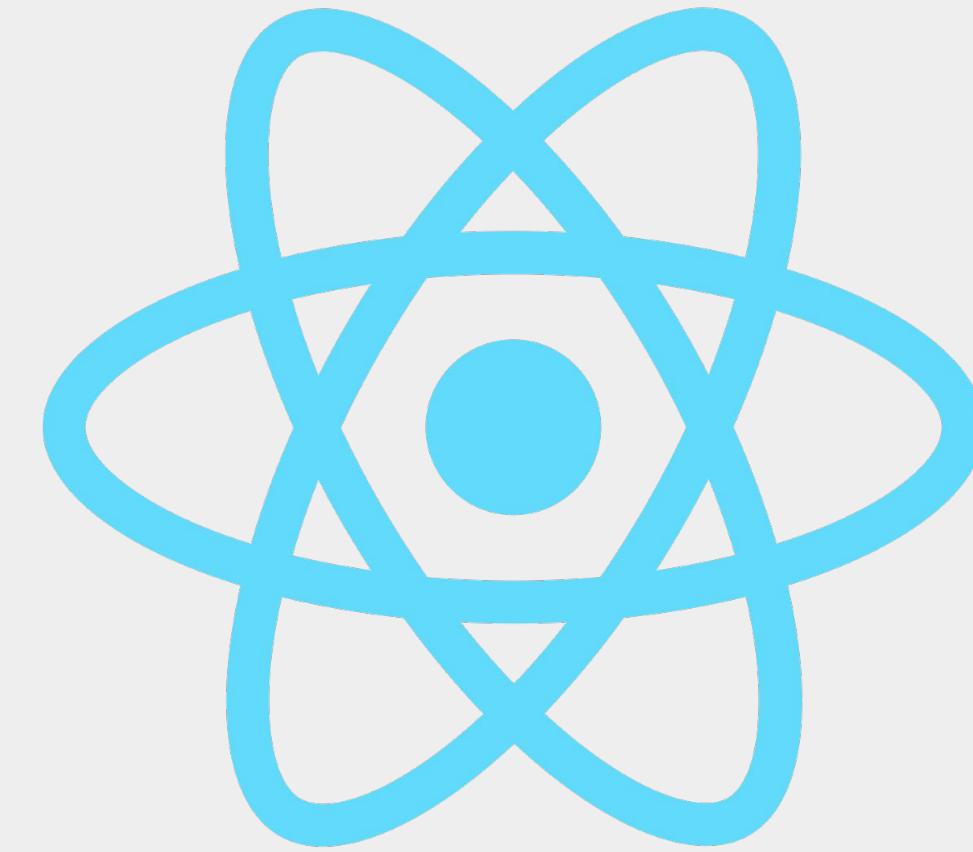
- Third party packages frequently break
- Build system frequently broke especially in the beginning
 - Metro Bundler Upgrade
- Same Issues as bigger web projects with the added complexity of native dependencies

Performance

- App Startup Time
 - Up to 15 seconds on older Android devices
 - Almost 3-5 seconds on newer devices
- We had to be very careful with rerenders



Flutter



All of these Issues are fixable, but...

Slow Development of React Native, loss of confidence

-

Android Performance issues were just addressed recently (Hermes VM)

-

Promises of 20% performance improvement, we wanted more

-

Big new features and a revamp of the shop were planned

-

Adoption of Material UI for our designs



Evaluation Phase - Key Questions

- Decision to definitely use it for prototyping
- Better Performance and Developer Experience?
- Can we find an easier state management pattern?
- Do Tracking and Native Packages support that we need?
- Can we migrate our users?
- Can we “fix” tracking?

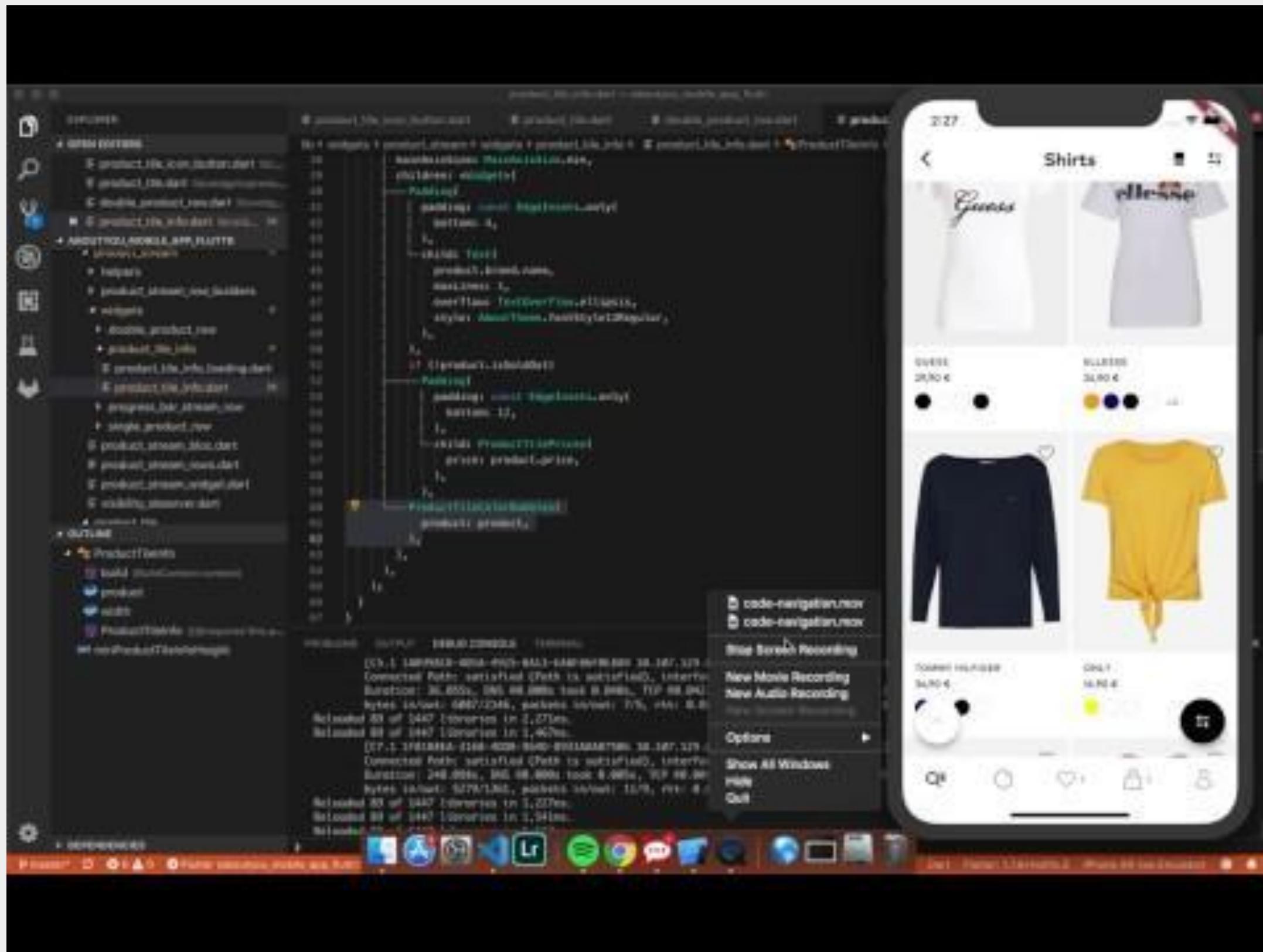


Approaching the Rewrite - Commitments

- Keep things simple
- Drop Features that are complex but not used frequently
- Keep State more local
- Make Code Easy to Navigate

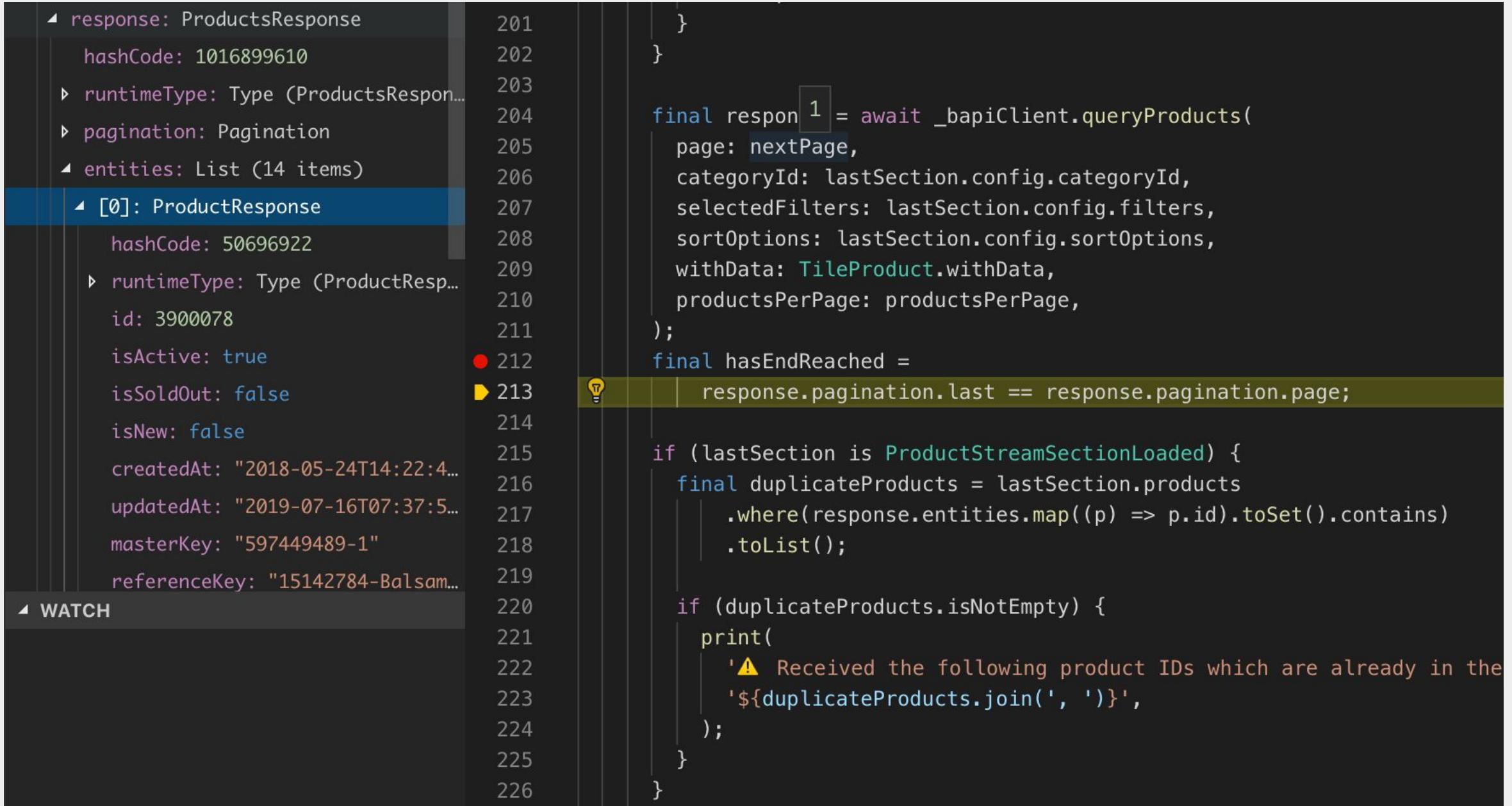


Why Flutter?



Developer Experience

- Very fast Hot Reload that keeps the State
- Dart supports AOT and JIT compilation
- Dart Developer Tools rapidly improving
- Almost no need to doublecheck on the device (except for notches)



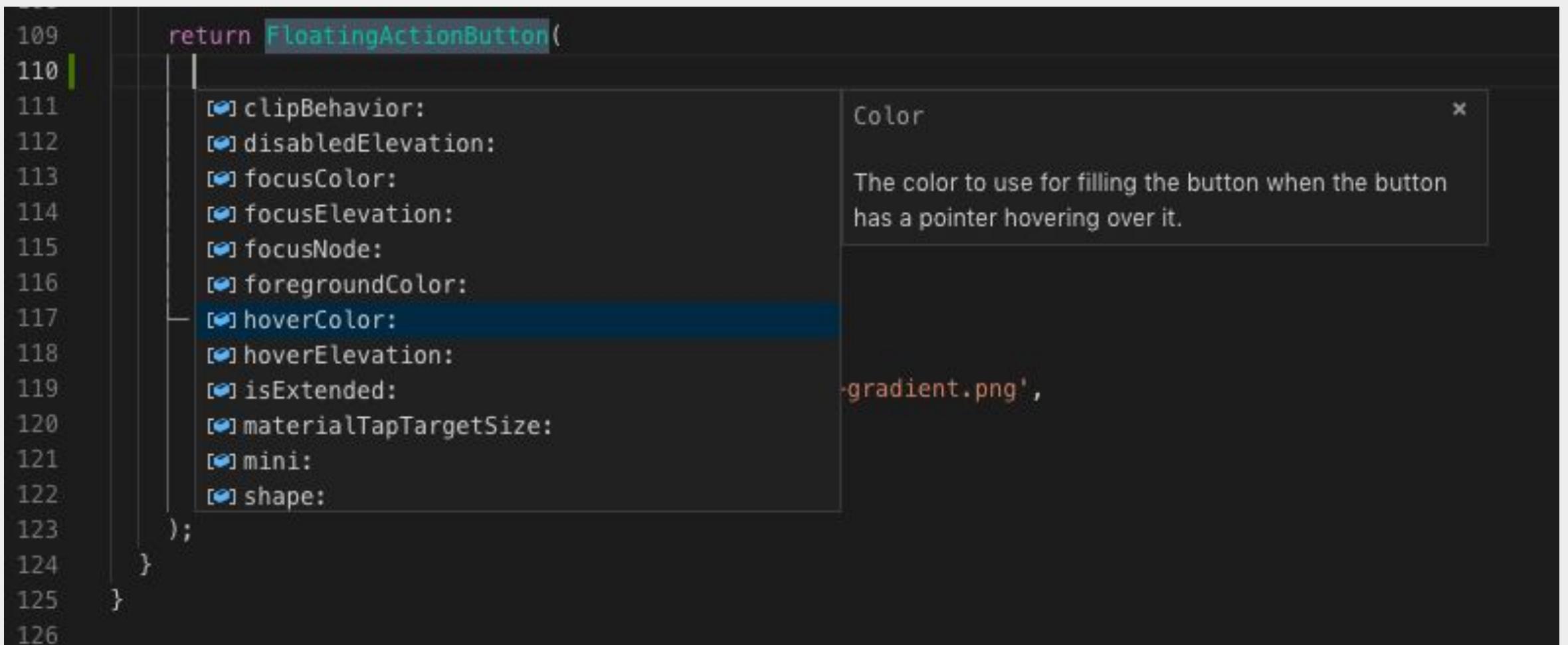
```

  response: ProductsResponse
  hashCode: 1016899610
  runtimeType: Type<ProductsResponse>
  pagination: Pagination
  entities: List<Product> (14 items)
    [0]: ProductResponse
      hashCode: 50696922
      runtimeType: Type<ProductResponse>
      id: 3900078
      isActive: true
      isSoldOut: false
      isNew: false
      createdAt: "2018-05-24T14:22:43Z"
      updatedAt: "2019-07-16T07:37:53Z"
      masterKey: "597449489-1"
      referenceKey: "15142784-Balsam..."

  ▲ WATCH

```

The screenshot shows a Dart code editor with syntax highlighting and a dark theme. A breakpoint is set at line 213, indicated by a red dot and a yellow warning icon. The code is part of a class handling product responses, specifically checking for duplicate products.



```

109 return FloatingActionButton(
110   clipBehavior:
111   disabledElevation:
112   focusColor:
113   focusElevation:
114   focusNode:
115   foregroundColor:
116   hoverColor: Color
117   hoverElevation:
118   isExtended:
119   materialTapTargetSize:
120   mini:
121   shape:
122 );
123
124 }
125
126

```

The screenshot shows a Dart code editor with syntax highlighting and a dark theme. A tooltip is displayed over the 'Color' parameter of the 'FloatingActionButton' constructor, providing a detailed description of its purpose.

Developer Experience

- Dart very easy to adapt with previous TypeScript experience (async, await etc.)
- Breakpoints
- Dart Developer Tools rapidly improving
- Great in Code Documentation of every Flutter Widget and core functions
- Almost no need to double check on the device (except for notches)



```
CustomScrollView(  
    slivers: <Widget>[  
        SliverAppBar(  
            title: Text('SliverAppBar'),  
            backgroundColor: Colors.green,  
            expandedHeight: 200.0,  
            flexibleSpace: FlexibleSpaceBar(  
                background: Image.asset(  
                    'assets/forest.jpg',  
                    fit: BoxFit.cover,  
                ),  
            ),  
        ),  
        SliverFixedExtentList(  
            itemExtent: 150.0,  
            delegate: SliverChildListDelegate(  
                [  
                    Container(color: Colors.red),  
                    Container(color: Colors.purple),  
                    Container(color: Colors.green),  
                    Container(color: Colors.orange),  
                    Container(color: Colors.yellow),  
                    Container(color: Colors.pink),  
                ],  
            ),  
        ),  
    ],  
);
```

Developer Experience - Material UI and Widgets

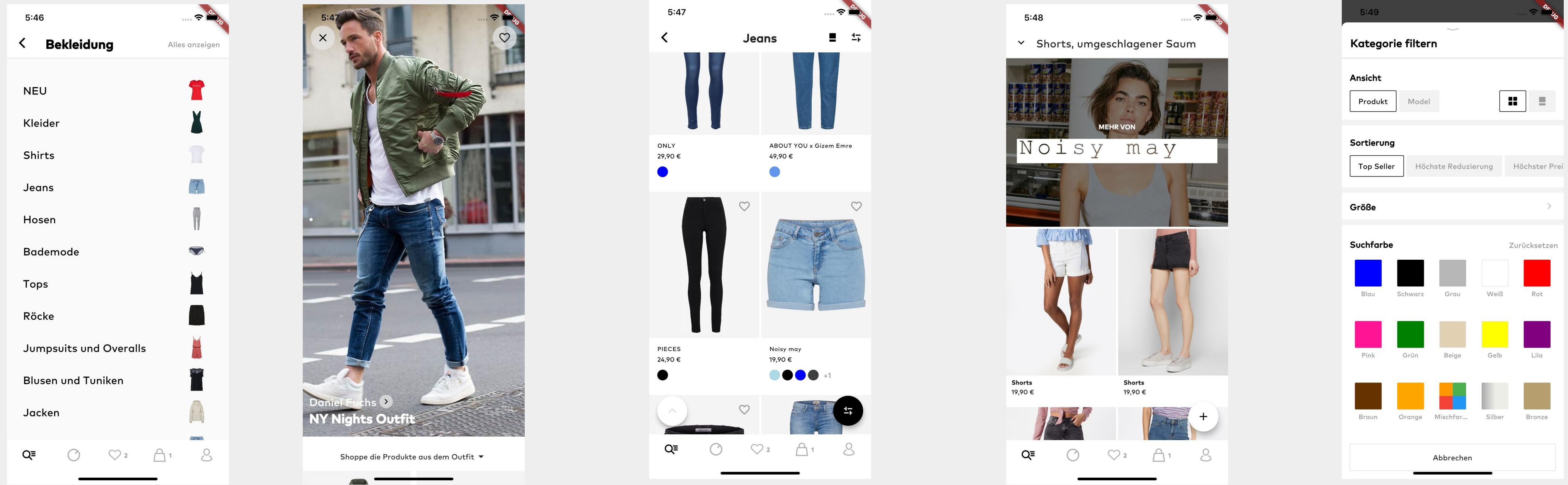
- Everything is a widget
- Material UI Elements “Out of the Box”
 - AppBar
 - FloatingActionButton
 - ...
- Widgets are highly optimized for the use case
 - Padding
 - Column / Row widgets
 - RotatedBox
 - ListView
 - Text
 - ...
- A lot of Requirements are just very easy to cover



Performance

- Dart is compiled down to native code
- Almost instant startup time
- No Slow Bridge
- No difference between newer iOS and Android devices
- Good performance even on old Android device





After 3 Months, so far:

Developers are happy

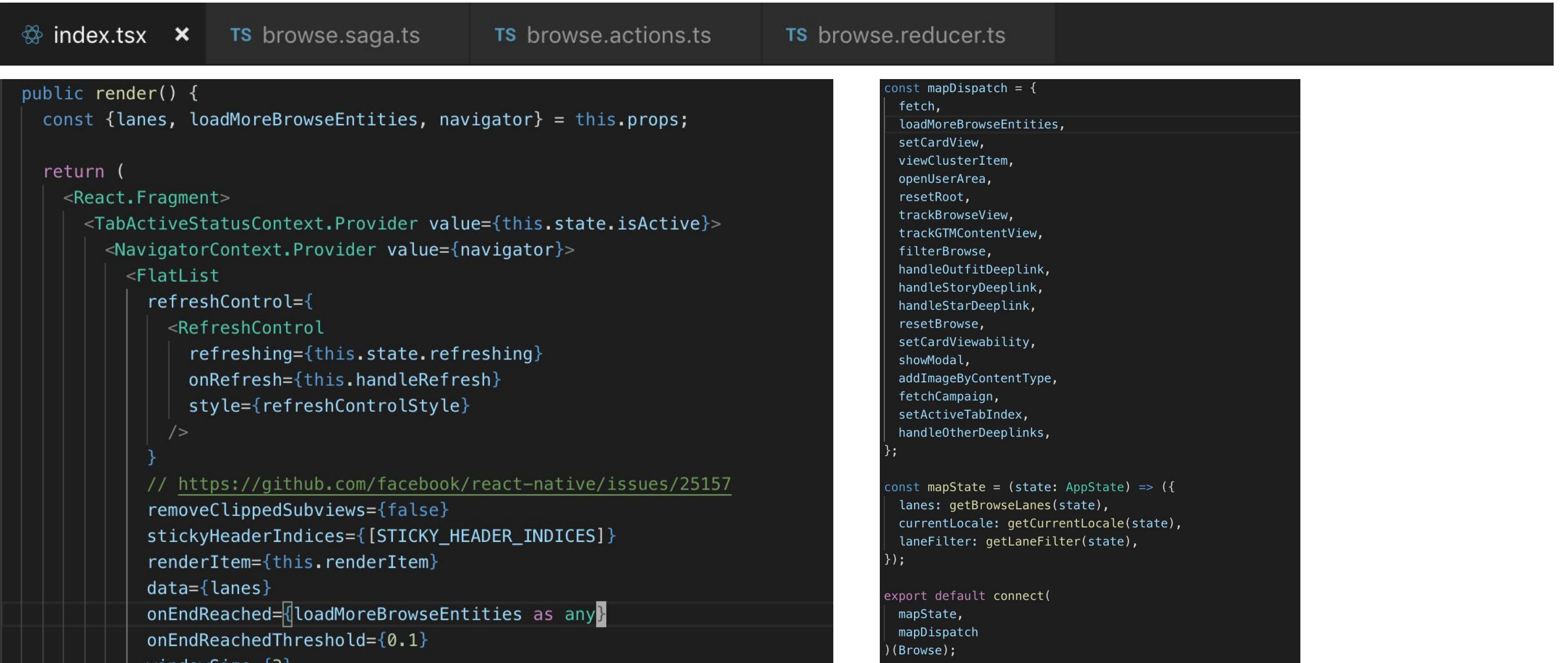
Improved Iteration Speed

Golden File Tests still a little flaky

Writing asserts is annoying

Timm says: "It's a lot less stupid than what we've used before"

The Bad - Code Navigation



index.tsx TS browse.saga.ts TS browse.actions.ts TS browse.reducer.ts

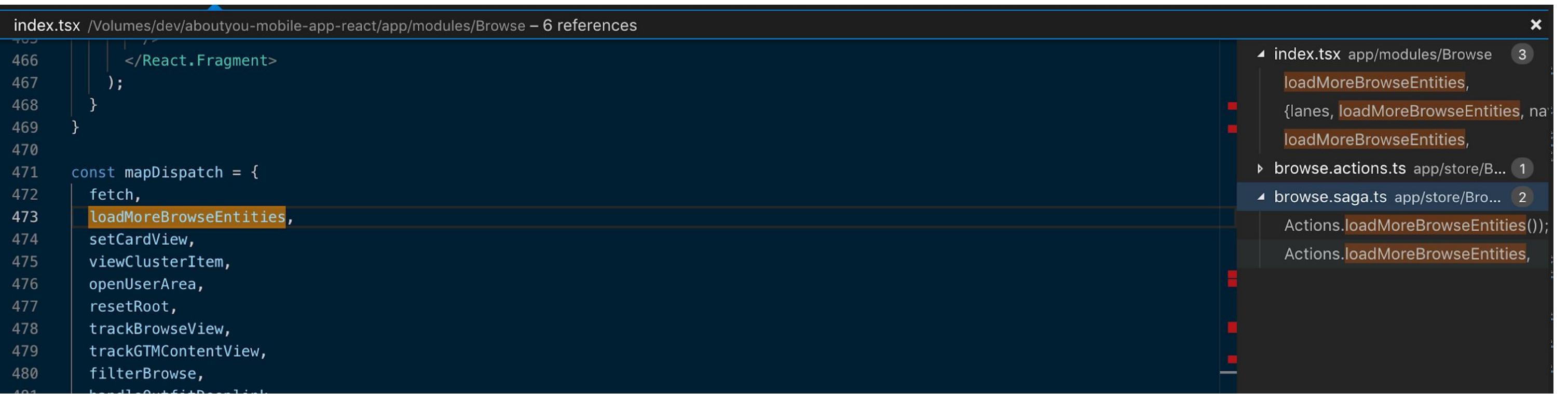
```
public render() {
  const {lanes, loadMoreBrowseEntities, navigator} = this.props;

  return (
    <React.Fragment>
      <TabActiveStatusContext.Provider value={this.state.isActive}>
        <NavigatorContext.Provider value={navigator}>
          <FlatList
            refreshControl={
              <RefreshControl
                refreshing={this.state.refreshing}
                onRefresh={this.handleRefresh}
                style={refreshControlStyle}
              />
            }
            // https://github.com/facebook/react-native/issues/25157
            removeClippedSubviews={false}
            stickyHeaderIndices={[STICKY_HEADER_INDICES]}
            renderItem={this.renderItem}
            data={lanes}
            onEndReached={loadMoreBrowseEntities as any}
            onEndReachedThreshold={0.1}
            initialNumToLoad={2}
          />
        
```

```
const mapDispatch = {
  fetch,
  loadMoreBrowseEntities,
  setCardView,
  viewClusterItem,
  openUserArea,
  resetRoot,
  trackBrowseView,
  trackGTMContentView,
  filterBrowse,
  handleOutfitDeeplink,
  handleStoryDeeplink,
  handleStarDeeplink,
  resetBrowse,
  setCardViewability,
  showModal,
  addImageByContentType,
  fetchCampaign,
  setActiveTabIndex,
  handleOtherDeeplinks,
};

const mapState = (state: AppState) => ({
  lanes: getBrowseLanes(state),
  currentLocale: getCurrentLocale(state),
  laneFilter: getLaneFilter(state),
});

export default connect(
  mapState,
  mapDispatch
)(Browse);
```



index.tsx /Volumes/dev/aboutyou-mobile-app-react/app/modules/Browse – 6 references

```
465   );
466   </React.Fragment>
467 );
468 }
469 }

const mapDispatch = {
  fetch,
470   loadMoreBrowseEntities,
471   setCardView,
472   viewClusterItem,
473   openUserArea,
474   resetRoot,
475   trackBrowseView,
476   trackGTMContentView,
477   filterBrowse,
478   handleOutfitDeeplink,
```

- index.tsx app/modules/Browse 3
loadMoreBrowseEntities,
{lanes, loadMoreBrowseEntities, na...
loadMoreBrowseEntities,
- browse.actions.ts app/store/B... 1
loadMoreBrowseEntities,
- browse.saga.ts app/store/B... 2
Actions.loadMoreBrowseEntities();
Actions.loadMoreBrowseEntities,

The Bad - Code Navigation

```
yield takeEvery(  
  Actions.loadMoreBrowseEntities,  
  bindAsyncAction(Actions.loadingEntities)(worker)  
)
```

- No easy Code Navigation with CMD + Shift
- Navigation through a lot of boilerplate
- Actions can be used throughout the code and even be used in the side effects

```
export function* watchForLoadMoreBrowseEntities() {  
  function* worker() {  
    const sectionIds: string[] = yield select(getSectionIds);  
    const activeSectionIds: string[] = yield select(getActiveSectionIds);  
    const inactive = R.difference(sectionIds, activeSectionIds);  
    const newActive = R.take<string>(SECTION_BATCH_SIZE, inactive);  
  
    if (newActive.length === 0) {  
      return;  
    }  
    yield put(  
      updateActiveSectionIds({  
        sectionIds: R.concat(activeSectionIds, newActive),  
      })  
    );  
  }  
}
```

The Bad - Selectors

```
// TODO: needs major refactoring
export const getBrowseLanes = createSelector(
  getSectionsWithItems,
  getFilteredNewsCards,
  getLaneFilter,
  (sections, cards, currentLane): BrowseLaneTypes[] => {
    if (!cards || sections.length === 0) {
      return [];
    }

    if (currentLane === LANE_FILTER_ALL) {
      let laneIndex = 0;
      let cardIndex = 0;

      return BROWSE_LANE_ORDER.map((type, index) => {
        switch (type) {
          case BROWSE_LANES.NORMAL_BROWSE_LANE:
            const section: Cluster | undefined = sections[laneIndex++];
            return section && clusterToLane(section);

          case BROWSE_LANES.NEWS_CARD:
            const fullWidth = cardIndex === 0;
            return getCardLane(cards[cardIndex++], fullWidth);

          case BROWSE_LANES.SEARCH_BAR:
            return getSearchBarLane();

          case BROWSE_LANES.TABS:
            return getTabLane();

          case BROWSE_LANES.STORY:
            return getStoryLane();

          case BROWSE_LANES.NEWS_CARD_SLIDER:
            return getCardSliderLane(cards.slice(cardIndex));

          default:
            return {type, data: 'normalLane', key: index.toString()}; // TODO remove
        }
      })
        .filter(notEmpty)
        .concat(sections.slice(laneIndex).map(clusterToLane));
    }
  }
)
```

State is normalized so selectors are used to gather data
Grow in complexity fast
Carelessly implemented selectors are often the cause for wasted renders



=====
RN Bridge
(Java/C++)
=====

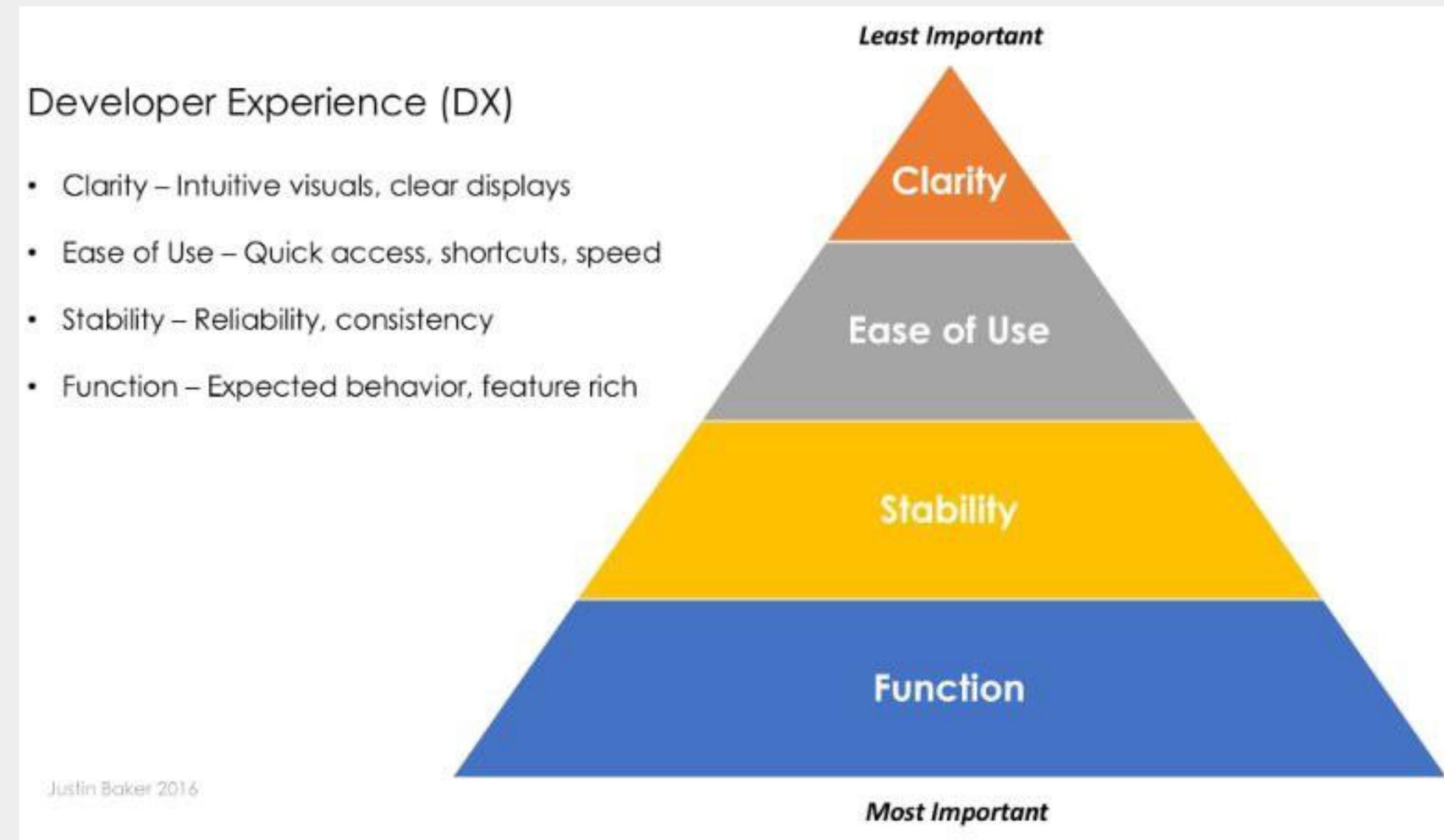
JS
Virtual Machine
(JavaScriptCore)

Pain Point: React Native JS Bridge

- 2 Threads: UI and Javascript
- In itself both of them blazingly fast
- Issues arise, when too much data gets passed between them
- Rerender can cause a lot more issues than in the web
- Naive implementation of animations causes countless rerenders
- -> **You need to be careful about the state**

State Management with Redux (+ Redux-Saga)

- First project that adapted Redux / Redux-Saga
- At some parts of the code it's almost impossible to follow the application flow
- Were we went wrong:
 - Keeping state strictly normalized (entities)
 - Almost no local state in components (lot's of boilerplate and useless sagas)
 - Redux-Saga's generators are not typed (we introduced our own library that relies on async / await)

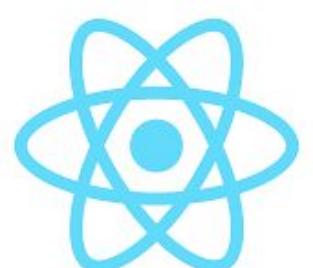
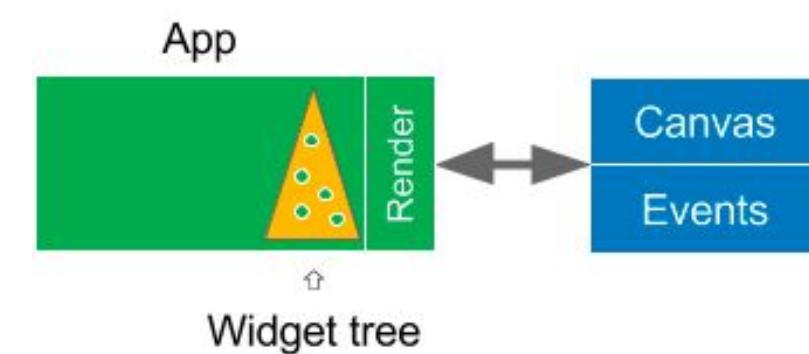
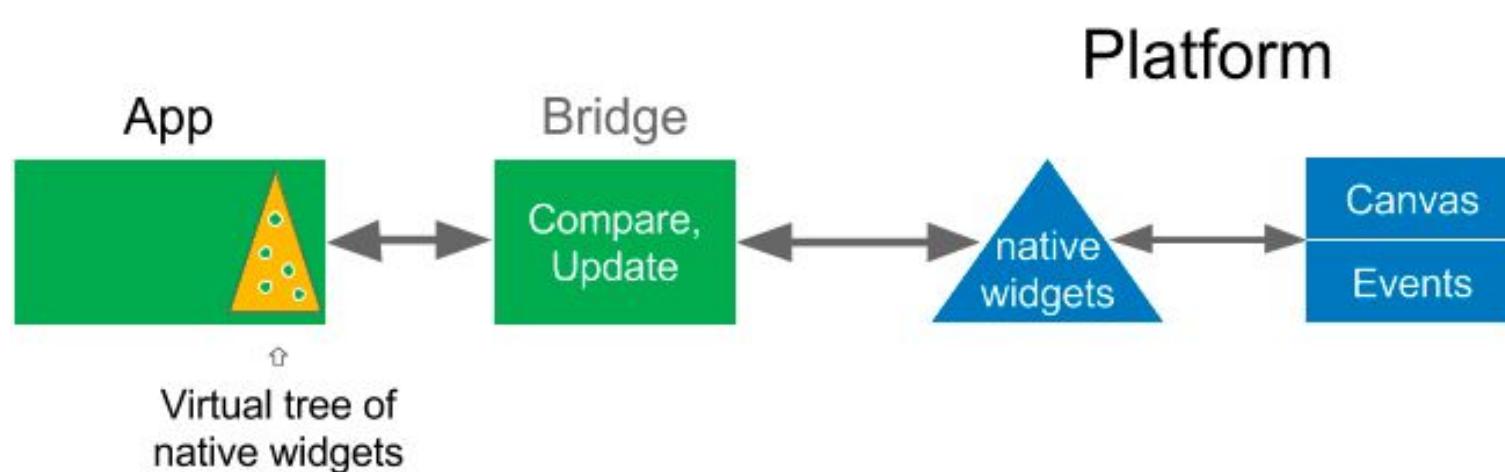
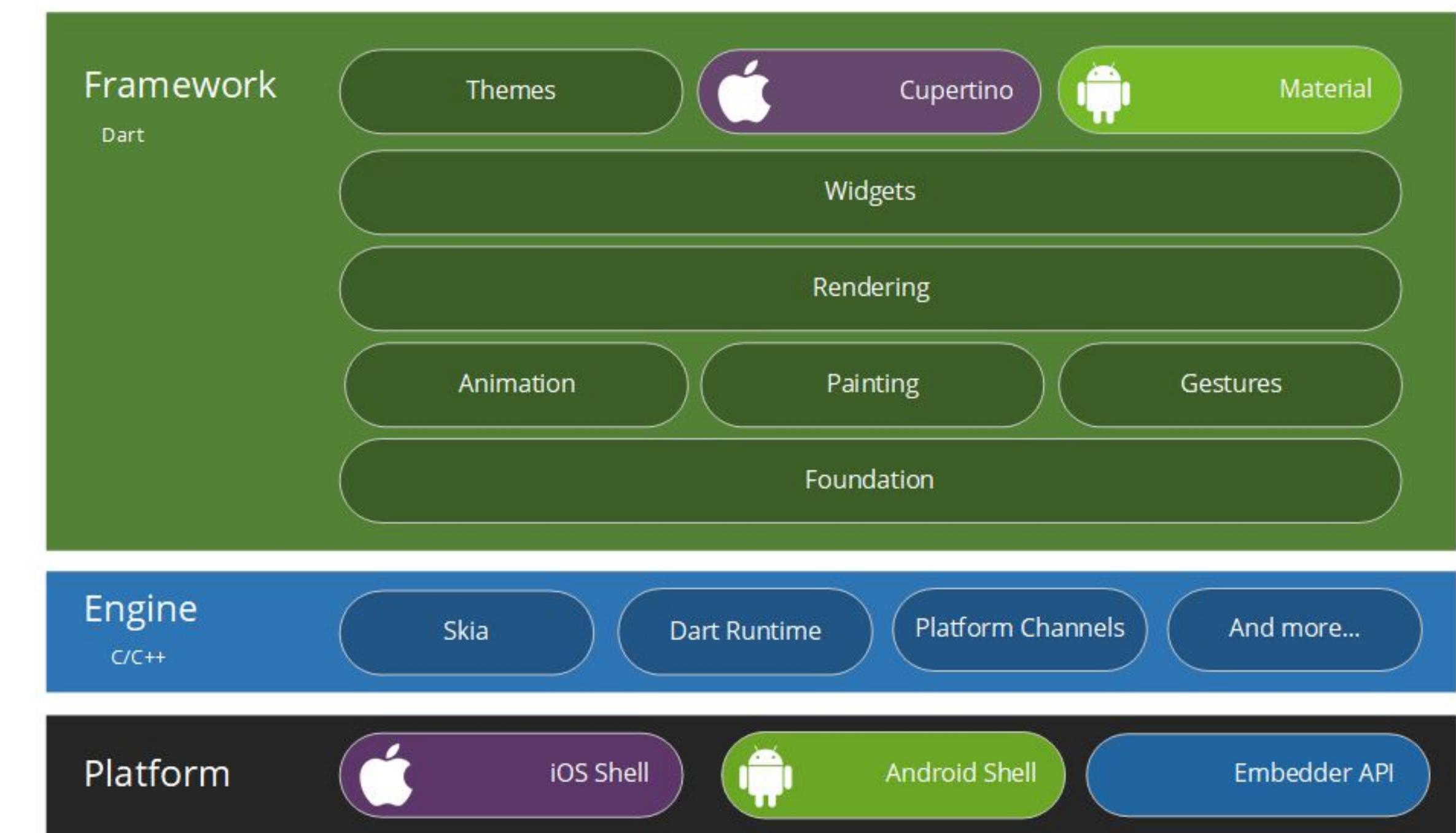
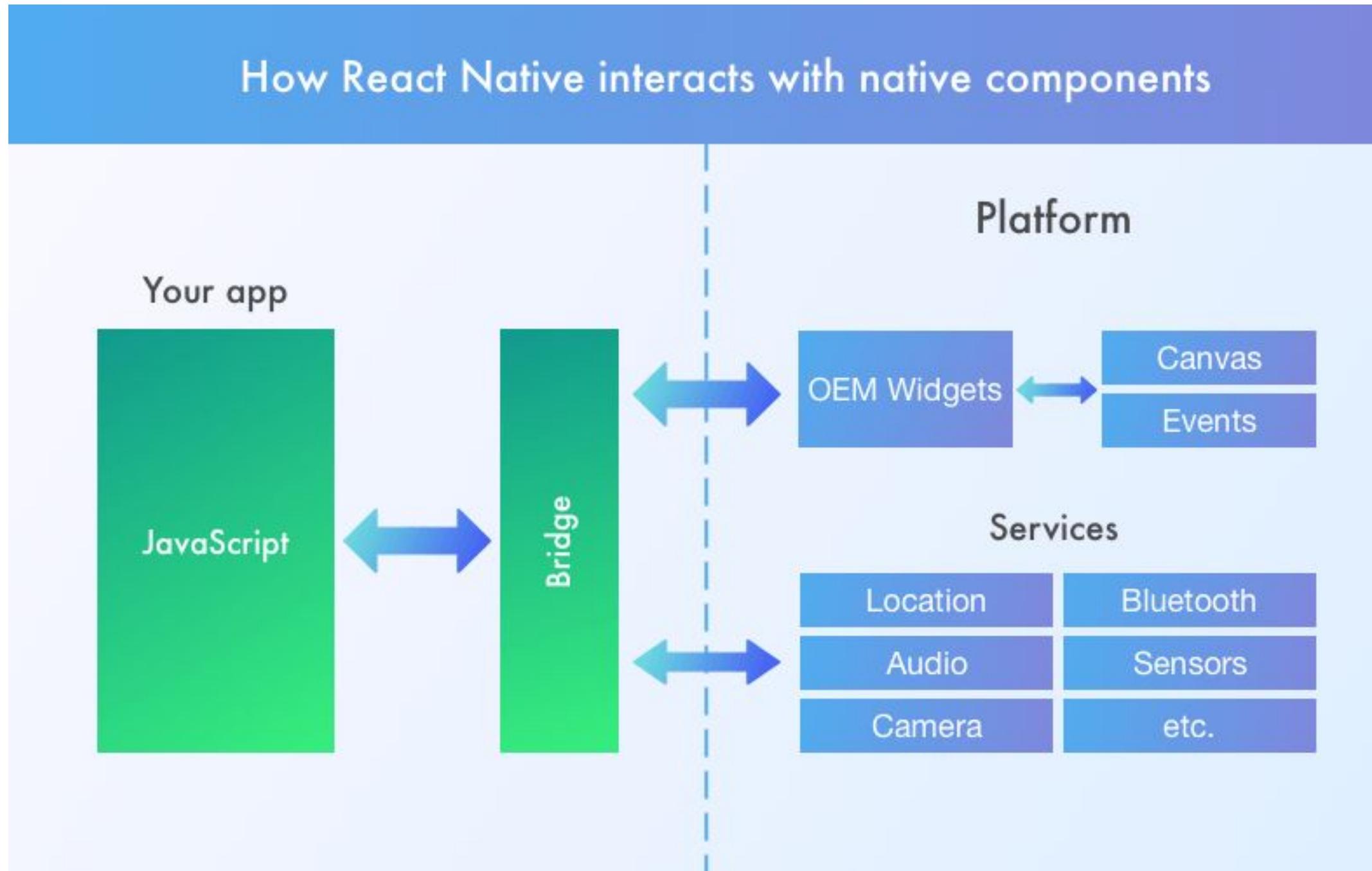


Pain Point: Developer Experience

- Webpack tends to get slow in bigger projects
- Debug Mode and Hot Reload are slow
- DEV mode uses a different environment (Chrome V8 vs. JavascriptCore VM)
- Bugs that only occur on Android / iOS and are hard to debug
- Bob the Buildserver fails often
- React Native Updates introduce a lot of changes are notoriously hard to follow up on
- Component APIs not always consistent between platforms

What is so different about Flutter?

“Flutter Owns Every Pixel”



1.Moving Away from React Native

From React Native to Flutter

2.Fixing our State Management

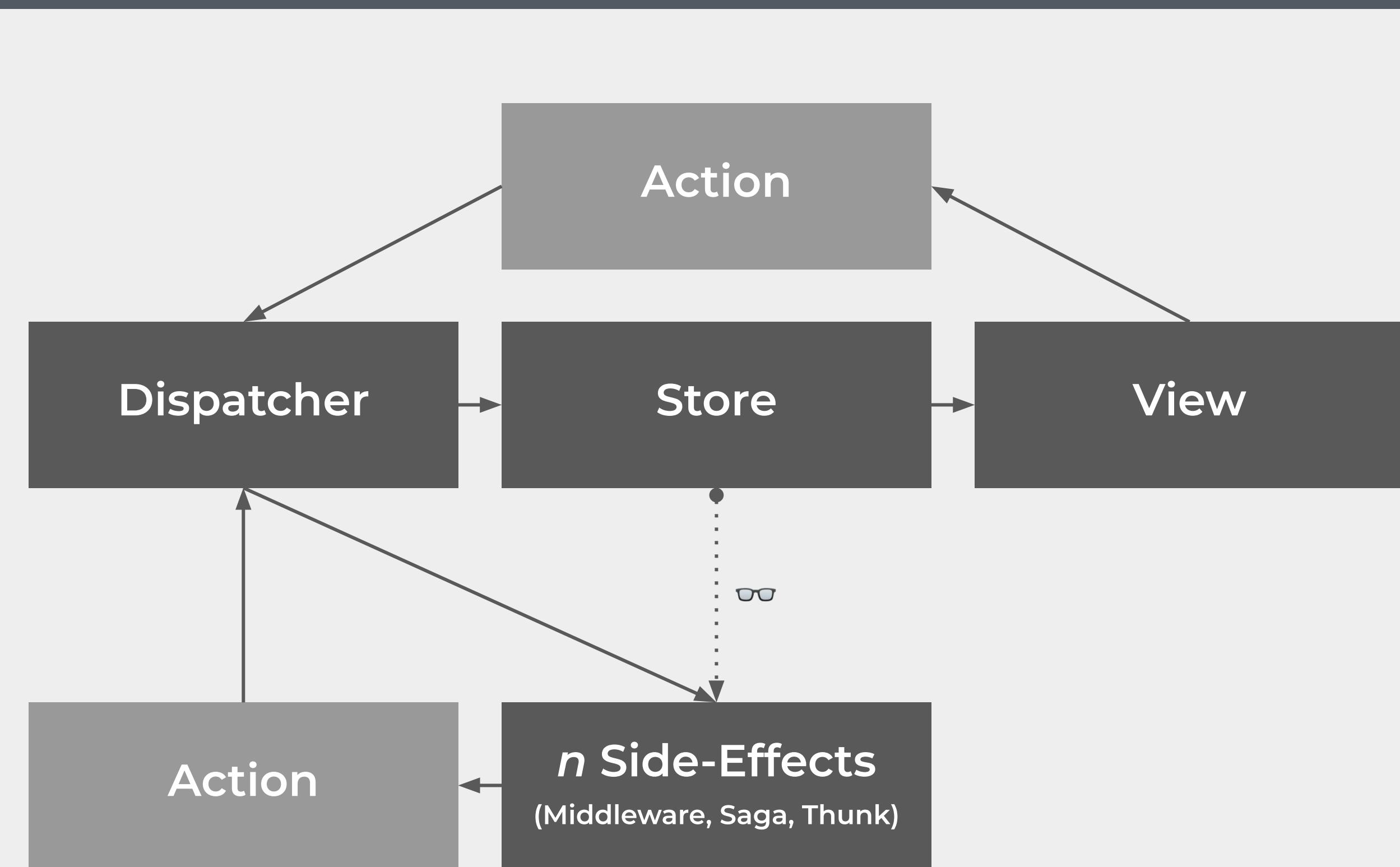
Our interpretation of the BLoC pattern

3.Golden File Testing with Flutter

Our approach, the benefits, and challenges to Golden File testing

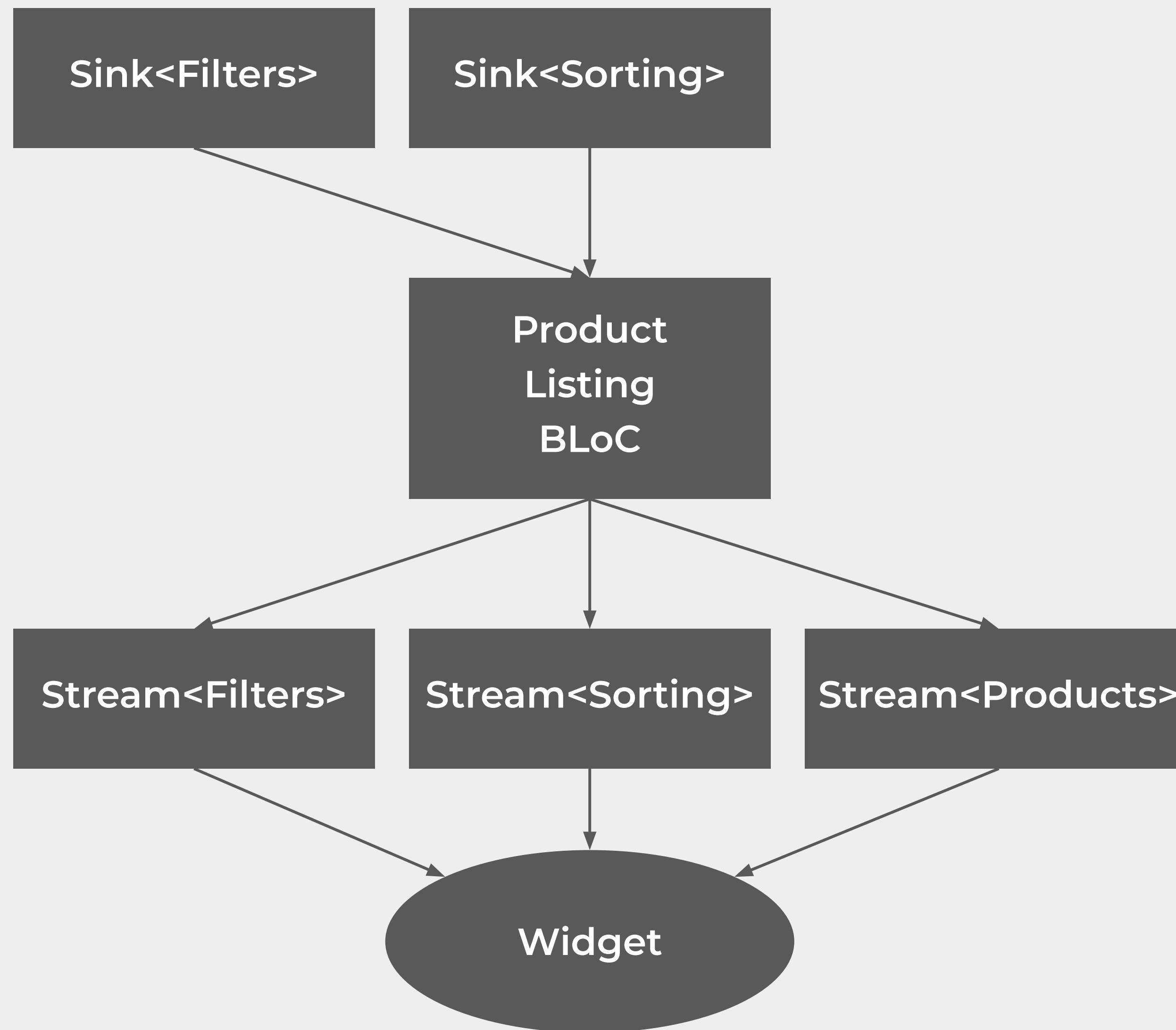
4.Developing Interactions in Flutter

How we created user-driven animations for our new ADP



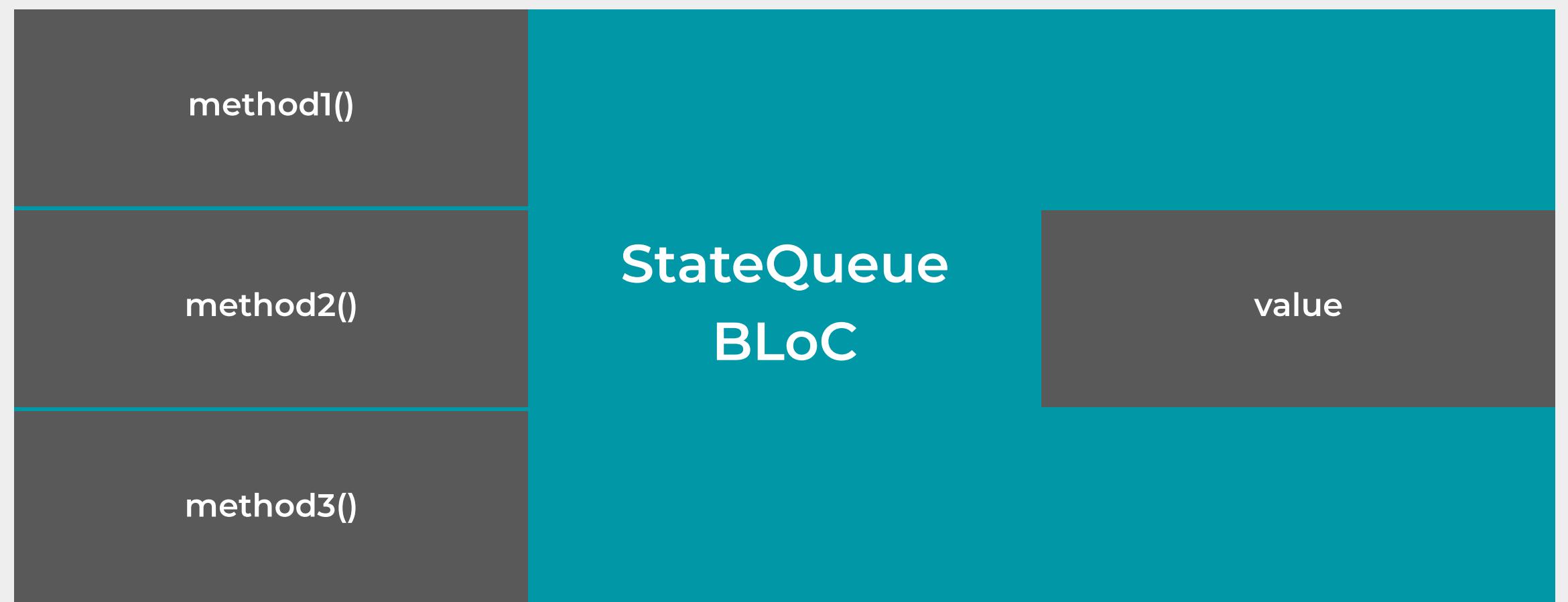
Redux Recap

- 1 Global Dispatcher, 1 Global State
- Simple in theory, not so much in our reality



BLoC (Business Logic Component)

- Sinks as Inputs
- Streams as Outputs
- Popular in the Flutter ecosystem due to being shown by Google



Our current setup

- ValueNotifier
- Provider
- StateQueue
- WithBloc
- HTTP Caching (WIP)

```
/// A [ChangeNotifier] that holds a single value.  
///  
/// When [value] is replaced with something that is not equal to the old  
/// value as evaluated by the equality operator ==, this class notifies its  
/// listeners.  
Michael Goderbauer, 3 months ago | 5 authors (Adam Barth and others)  
class ValueNotifier<T> extends ChangeNotifier implements ValueListenable<T> {  
    /// Creates a [ChangeNotifier] that wraps this value.  
    ValueNotifier(this._value);  
  
    /// The current value stored in this notifier.  
    ///  
    /// When the value is replaced with something that is not equal to the old  
    /// value as evaluated by the equality operator ==, this class notifies its  
    /// listeners.  
    @override  
    T get value => _value;  
    T _value;  
    set value(T newValue) {  
        if (_value == newValue)  
            return;  
        _value = newValue;  
        notifyListeners();  
    }  
  
    @override  
    String toString() => '${describeIdentity(this)}($value)';  
}
```

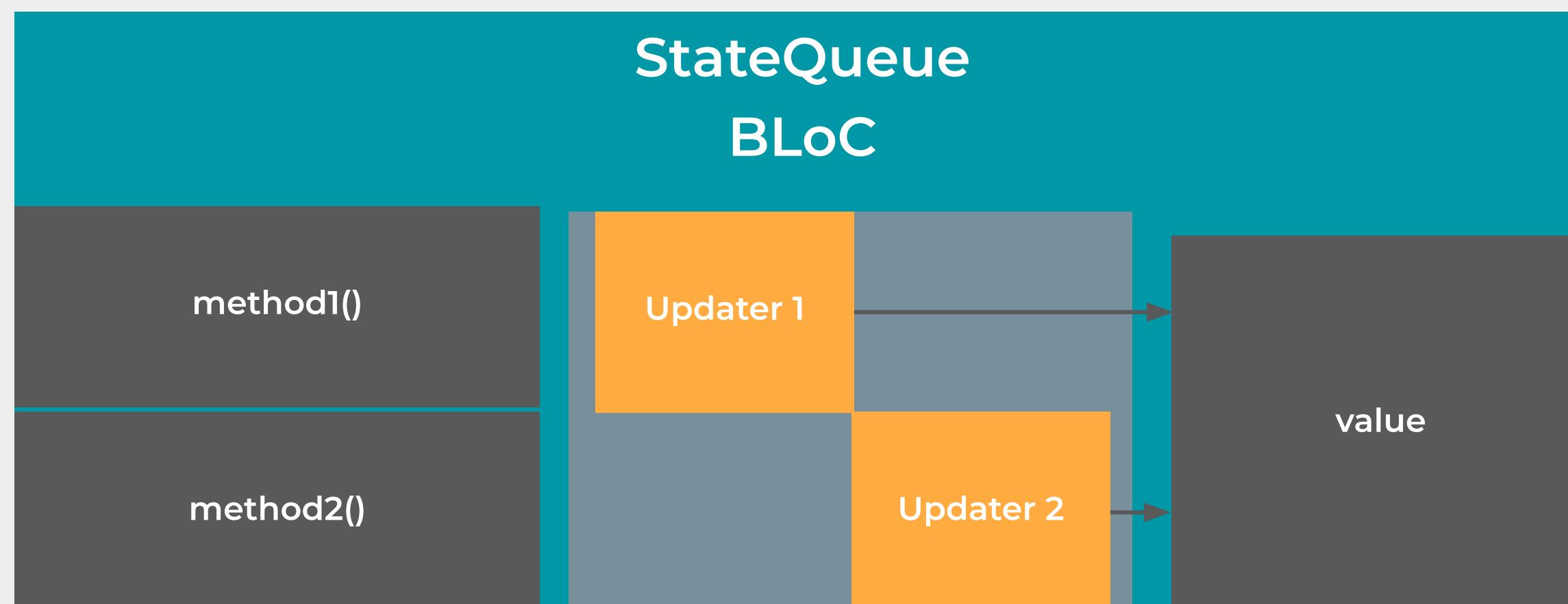
ValueNotifier

- ValueListenable (interface)
 - Holds a value of type T
 - Exposes listener registration
- ChangeNotifier (class)
 - Manages listeners
 - Allows subclass to notify listeners
- ValueNotifier implements ValueListenable using ChangeNotifier

```
class AppRoot extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return ChangeNotifierProvider<BasketBloc>(  
      builder: (_) => BasketBloc(),  
      child: MainNavigation(),  
    ); // ChangeNotifierProvider  
  }  
  
  // [...]  
  
  class BasketIcon extends StatelessWidget {  
    @override  
    Widget build(BuildContext context) {  
      final basketState = Provider.of<BasketBloc>(context).value;  
  
      return Row(  
        mainAxisAlignment: MainAxisAlignment.center,  
        children: <Widget>[  
          Icon(  
            Icons.shopping_cart,  
            color: Colors.black,  
          ), // Icon  
          SizedBox(width: 5),  
          Text(  
            '${basketState.productCount}',  
          ) // Text  
        ], // <Widget>[]  
      ); // Row  
    }  
  }
```

Provider

- Dependency Injection
- Implemented as a Widget
- Makes higher-up values available to children “via the context”
- Used to “pass” objects implicitly through many intermediate layers
 - E.g. from global basket BLoC to leaf icon badge



StateQueue

- Implements sequential execution of updaters on top of ValueNotifier
- Simplifies reasoning as it ensures that at most 1 state modification is running at any time
- Uninterruptible updaters can emit multiple states
 - E.g *loading*, then *loaded* state
- Suitable for most of our cases
 - Concurrent states are handled separately

StateQueue Demo

```
class BasketBloc extends StateQueue<Basket> {  
  BasketBloc() : super(BasketLoading());  
  
  /// Adds the product to the basket or increases its quantity by 1 if it exists  
  void addOrUpdateProductInBasket(int productId) {  
    run((state) async* {  
      yield BasketLoading();  
  
      final existingItem = state.exisitingItem(productId);  
  
      if (existingItem == null) {  
        yield await basketWithNewProduct(productId);  
      } else {  
        yield await basketWithUpdatedQuantities(  
          existingItem.key,  
          existingItem.quantity + 1,  
        );  
      }  
    });  
  }  
}
```

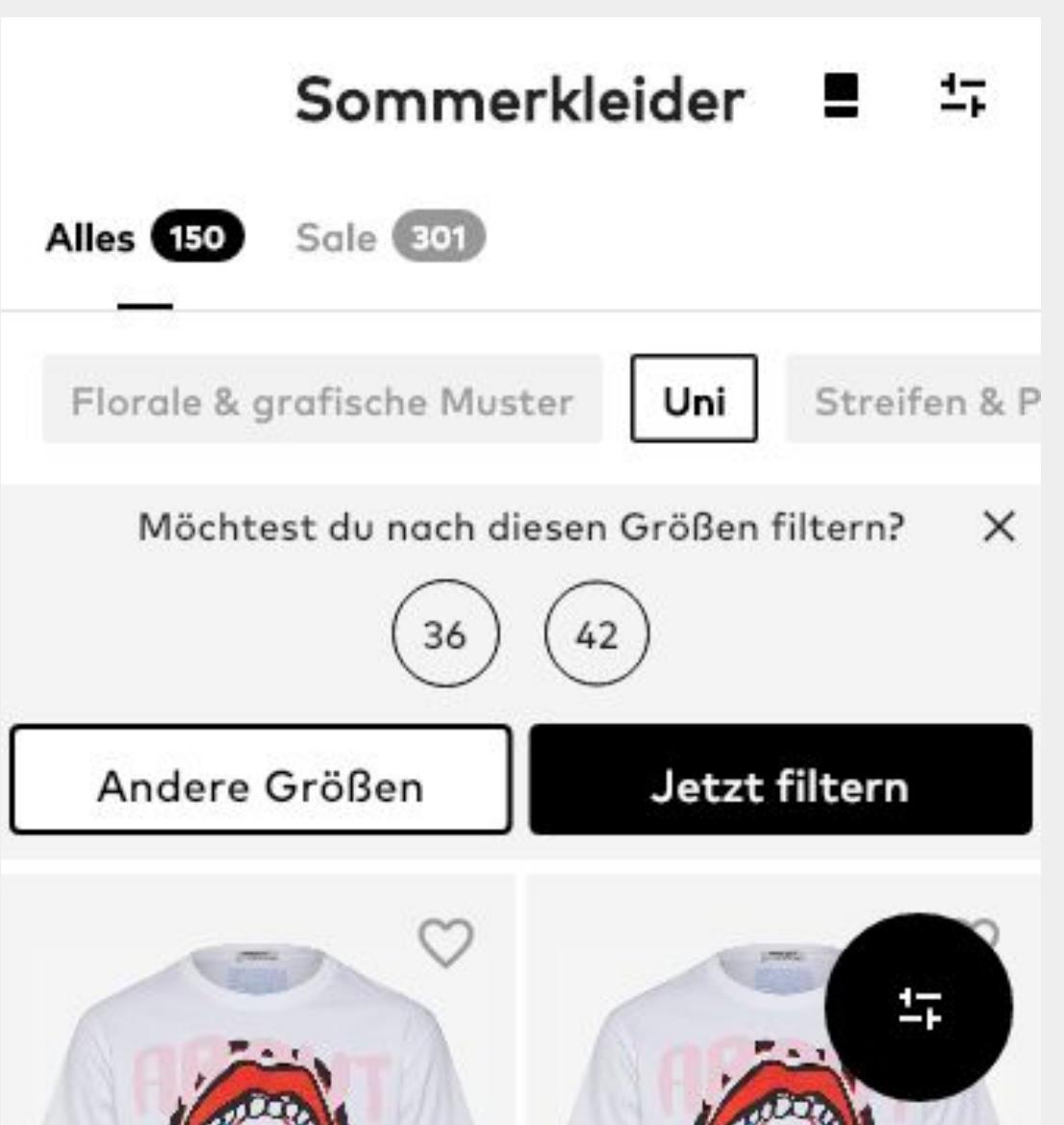
```
class CounterBloc extends StateQueue<int> {
    CounterBloc() : super(0);

    void increment() {
        run((state) async* {
            yield state + 1;
        });
    }
}

class Counter extends StatelessWidget {
    @override
    Widget build(BuildContext context) {
        return WithBloc<CounterBloc, int>(
            createBloc: (_) => CounterBloc(),
            builder: (context, bloc, state, _) {
                return MaterialButton(
                    child: Text('Count = $state'),
                    onPressed: bloc.increment,
                ); // MaterialButton
            },
        ); // WithBloc
    }
}
```

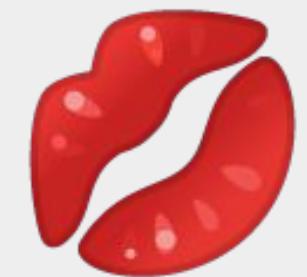
WithBloc

- Makes it easy to create local blocs during *build*
- Automatically recreates the BLoC when inputs change
 - Simpler than manual tracking in lifecycle methods
- Disposes the BLoC once the widget unmounts



HTTP Caching (WIP)

- Due to the decentralized nature of our state, the same requests happen in multiple places
- Sometimes even at nearly the same time
- Avoids having to align state placement with UI behavior
 - E.g. efficient scrolling lists, which discard hidden widgets
- We found it simpler to handle deduplication on this layer than to connect all those BLoCs to share data
- Today this is also commonly abstracted in GraphQL and similar technologies



Keeping It Simple

- BLoCs have a single, atomic state to the outside
- Concurrency not an issue as it's not possible
- Avoiding one-off problems where possible
 - E.g. some changes trigger a “restart” (sub second) of the app instead of trying to notify and update all states
- Blocs are either global (shared using Provider), or local and then passed as parameters
- Implementations are explicit
 - go to definition works

1.Moving Away from React Native

From React Native to Flutter

2.Fixing our State Management

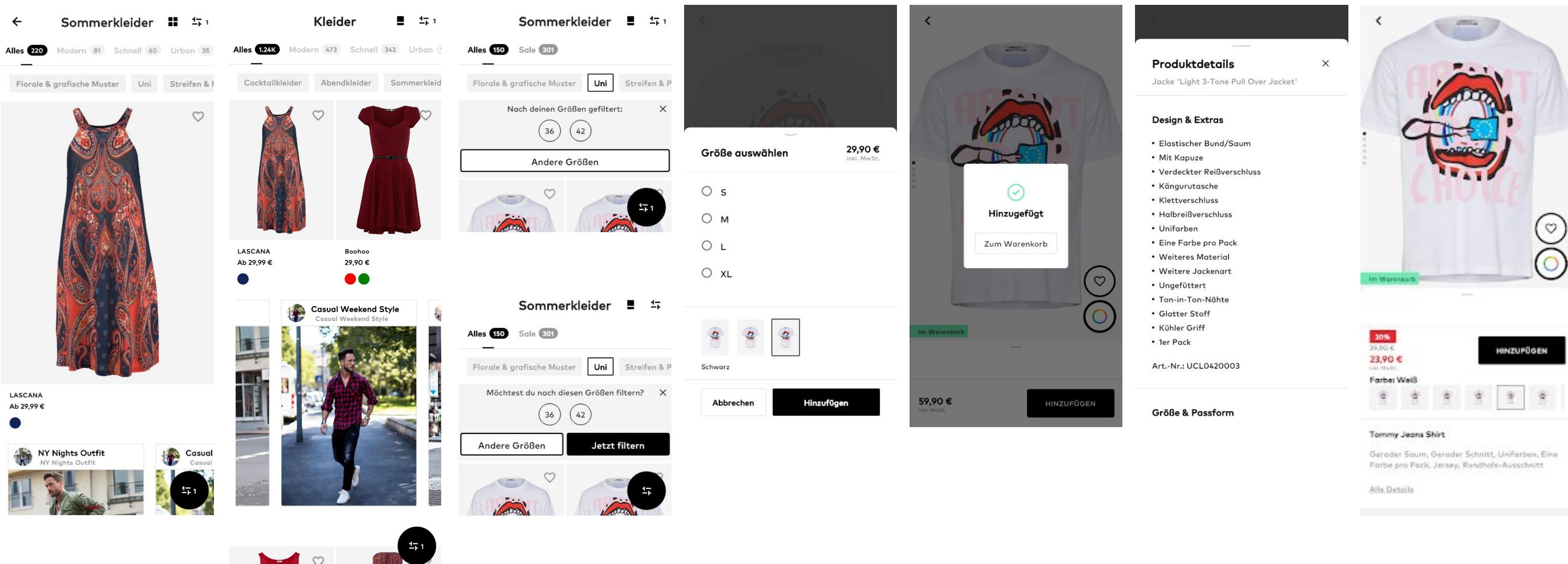
Our interpretation of the BLoC pattern

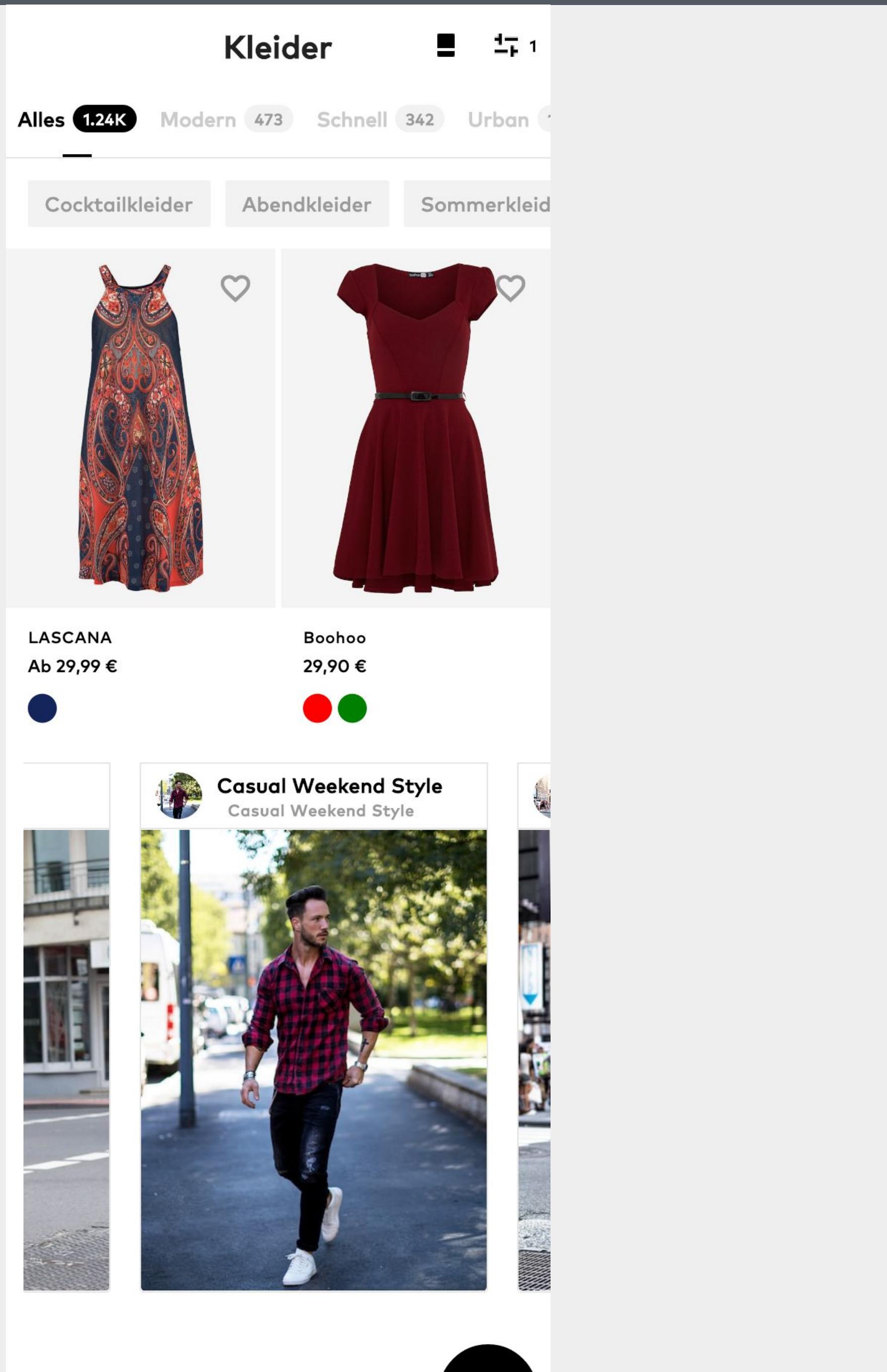
3.Golden File Testing with Flutter

Our approach, the benefits, and challenges to Golden File testing

4.Developing Interactions in Flutter

How we created user-driven animations for our new ADP





Our current approach

- Test higher-level widgets
 - Usually full pages
 - Sometimes even including navigation to another page
- Use few mocks
 - Test all involved app code
- Build up state / data from captured requests / responses
 - Saved on disk during “record mode”
 - Provided by an alternate HTTP client implementation used in BLoCs

```
import 'package:flutter/material.dart';
import 'package:flutter_test/flutter_test.dart';

import 'pages/article-detail-page-widget/article_detail_page_loader.dart';
import 'pages/page_test_utils.dart';

Future<void> main() async {
  testWidgets('Renders ADP', (tester) async {
    await tester.runAsync(
      () async {
        final testSetup = await wrapInMockedProviders(
          ArticleDetailPageLoader(
            productId: 123456,
            pageSource: DeeplinkPageSource(),
          ), // ArticleDetailPageLoader
          lastSeenProducts: [4072105, 3938478],
          mocksPath: './pages/adp/api-mocks/product-1',
        );

        await tester.pumpWidget(testSetup.widget);

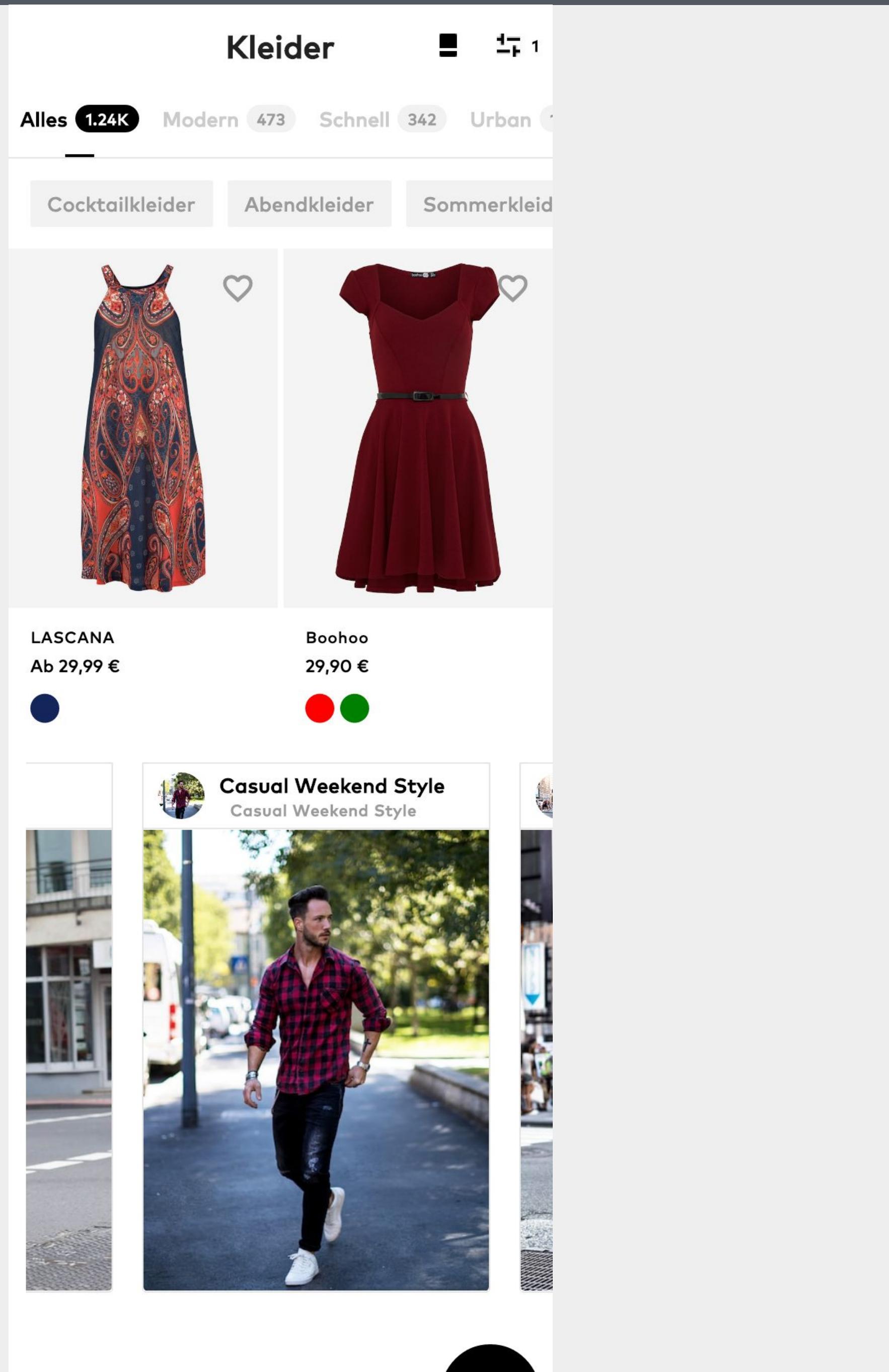
        await testSetup.pumpAndSettleIncludingHttpCalls(tester);

        await expectLater(
          find.byType(MaterialApp),
          matchesGoldenFile(
            'goldens/article_detail_page_widget_test_product1.png',
          ),
        );

        expect(testSetup.deleteUnusedRequestsFromDisk(), 0);
      },
    );
  });
}
```

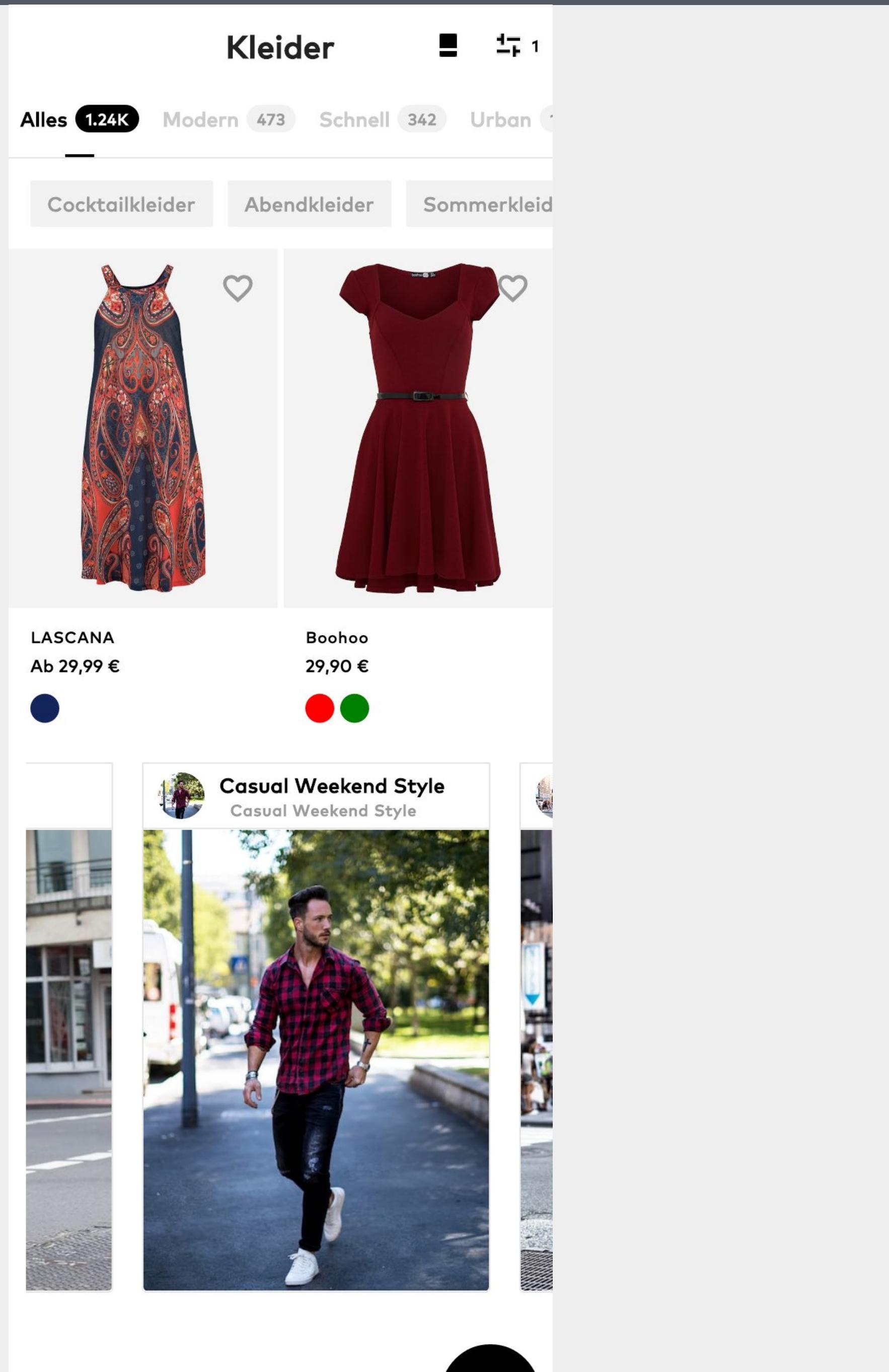
How it works

- testWidgets
 - Runs a headless version of the engine
 - No simulator or devices needed
- pump
 - Resolves pending Futures
 - Ability to advance the clock to control animations
- Custom HTTP Client using JSON API Provider
- Mocks (using Mockito)
- Ability to trigger user input like taps and drags
- GitLab CI + MacPro build server



Benefits of our current approach

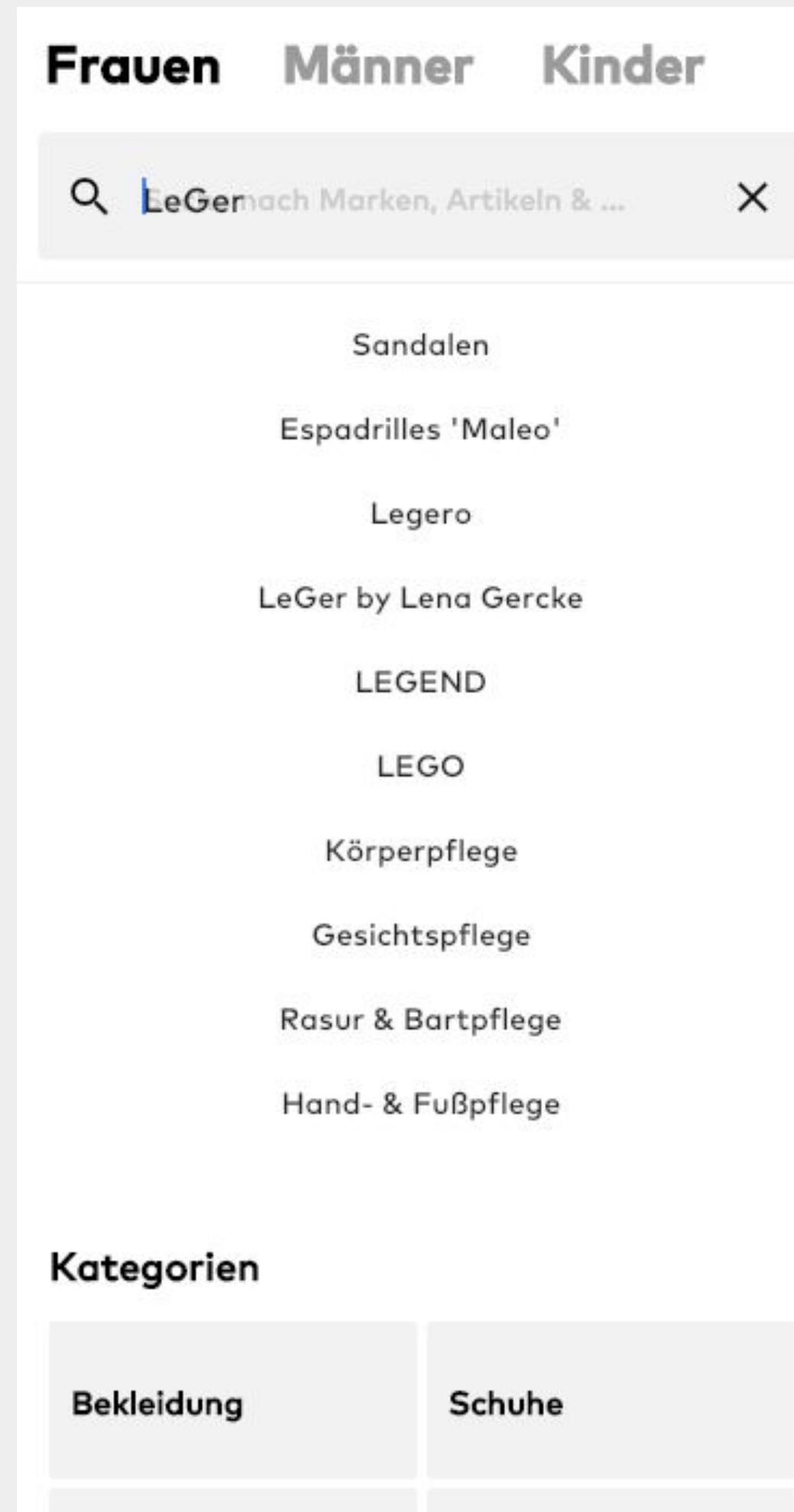
- We catch regressions (immediately and in CR)
- Full integration tests
 - Feels like a good primary test level
- Visual results
 - Much nicer than HTML snapshots
- Shared widgets are shown in many different context
- Self-contained due to captured responses
 - Not dependent on 3rd party system at test time
- Fast feedback on UI changes (~3 minutes for whole suite)
 - No simulator or devices required



Downsides of our current approach

- Testing variations of scenarios results in duplication of test code and data
 - Only a few very specific things might change between variants, while most of the page stays the same
- Screenshots alone often don't tell what to focus on
 - People approve when it looks good, but might miss what's important
- Some widgets are tested many times without significant changes
 - E.g. header bars
- Using real API responses
 - Schema updates of popular APIs lead to changes in many test cases
 - Response data changes over time (e.g. campaign status)
 - Some API calls are not to be repeated





General sources of flakiness

- Sometimes the rendering is unpredictable due to our approach to loading the data via HTTP responses
 - Amount of pumps depends on the amount of HTTP responses
 - For legacy reasons we currently advance the clock in each pump
 - Meaning animations move ahead for each response
 - Fix forthcoming
- Examples of UI elements changing with time
 - Scroll bars
 - Blinking cursors
 - Placeholder texts which don't disappear when setting input texts in tests
- Can't yet easily inspect file differences from failures on CI 😞



Learnings

- Write more tests
- Do not pack every scenario into an integration test
 - Use individual test for leaf widgets in a certain state
- Provide model mock data to get to specific cases quickly
 - Documents the desired state better than tiny changes in responses
- Use assertions in addition to the final screenshot
 - Point to what's important
- Testing complex user interactions is possible
- Constantly keep improving our tooling
 - It's worth the investment

1.Moving Away from React Native

From React Native to Flutter

2.Fixing our State Management

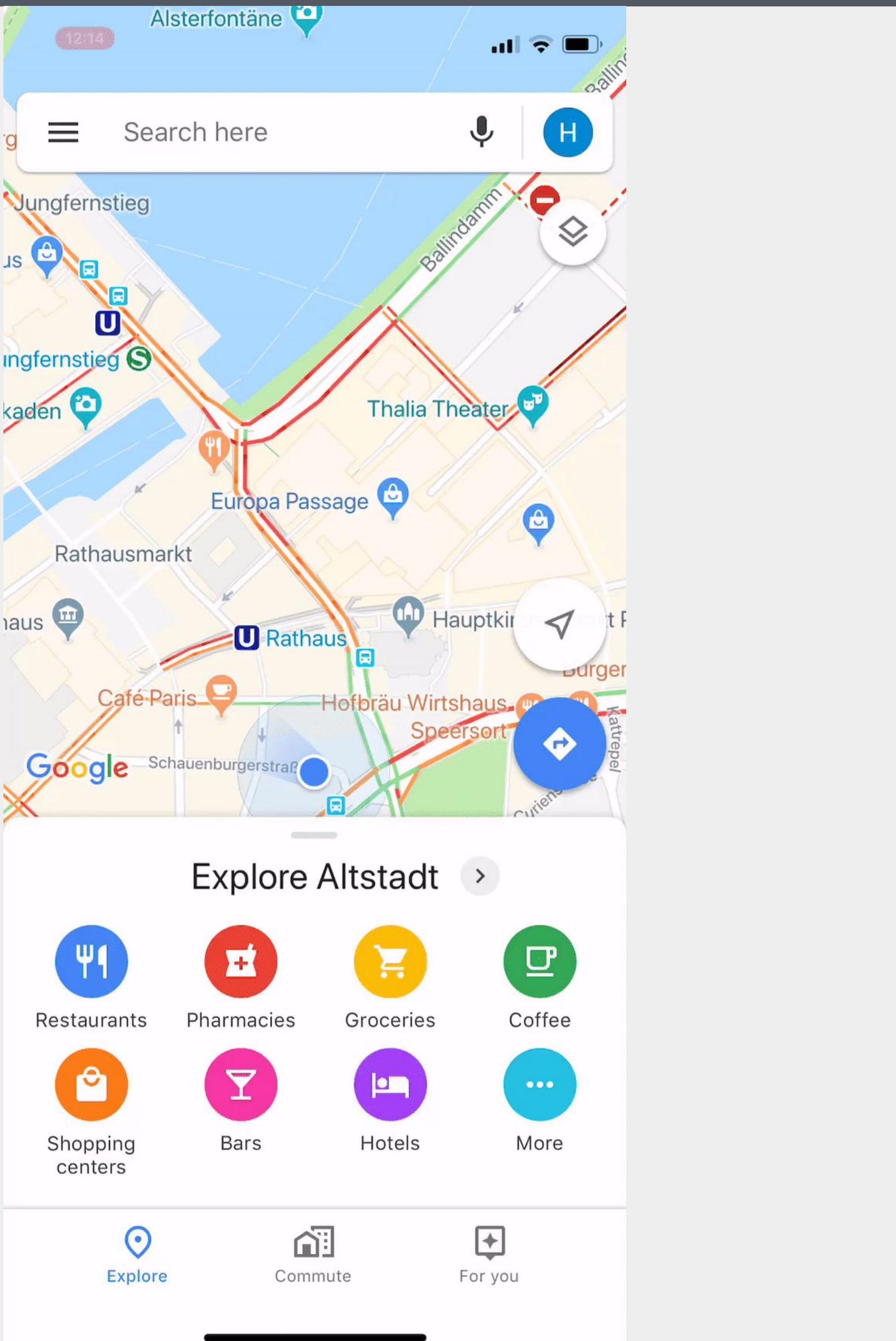
Our interpretation of the BLoC pattern

3.Golden File Testing with Flutter

Our approach, the benefits, and challenges to Golden File testing

4.Developing Interactions in Flutter

How we created user-driven animations for our new ADP



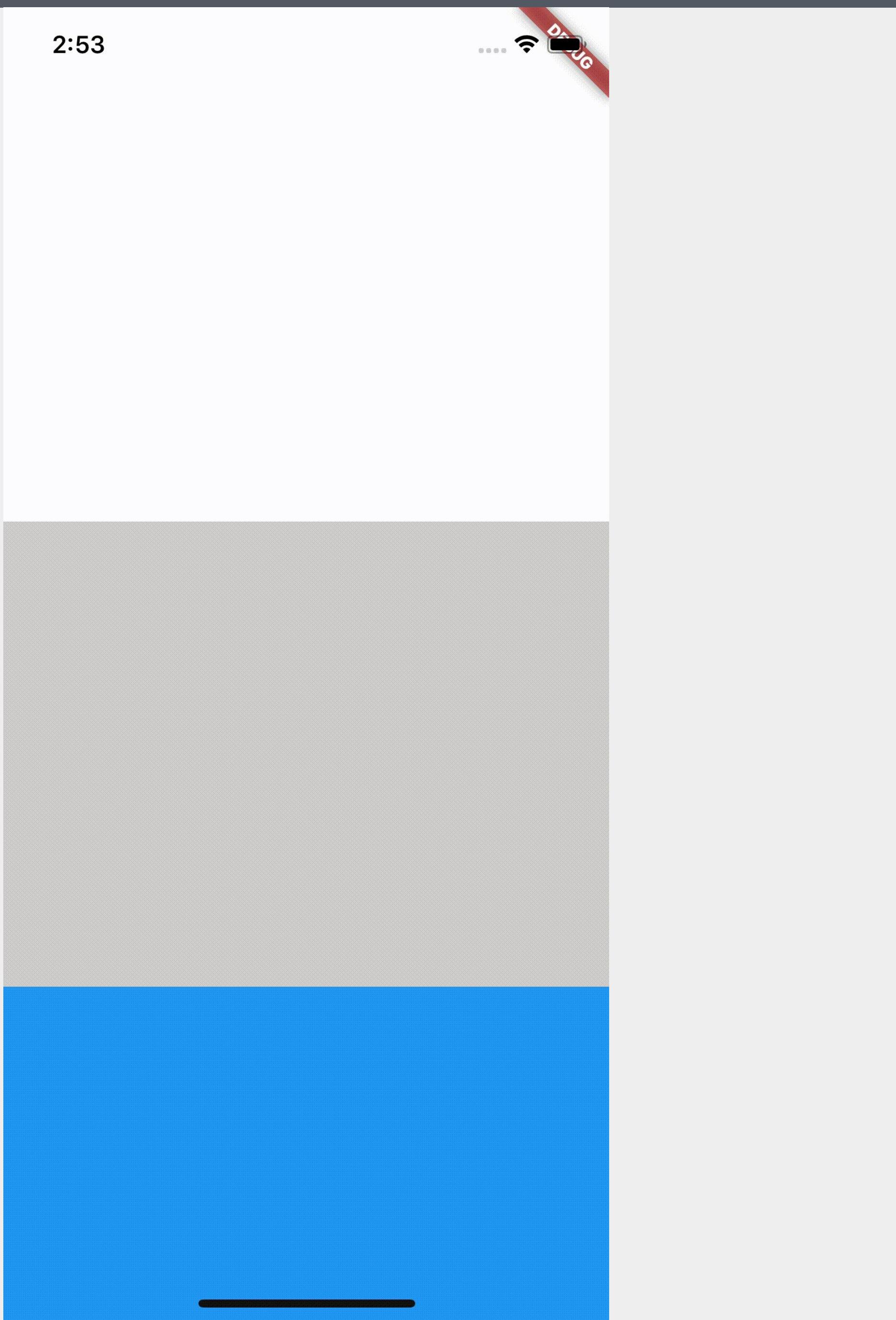
Our new Product Page

- Draggable Sheet which becomes Scrollable when fully extended (Google Maps style)
- Fade in a dark overlay when being dragged
 - should feel responsive to the user
 - When clicked, move the sheet back to initial position



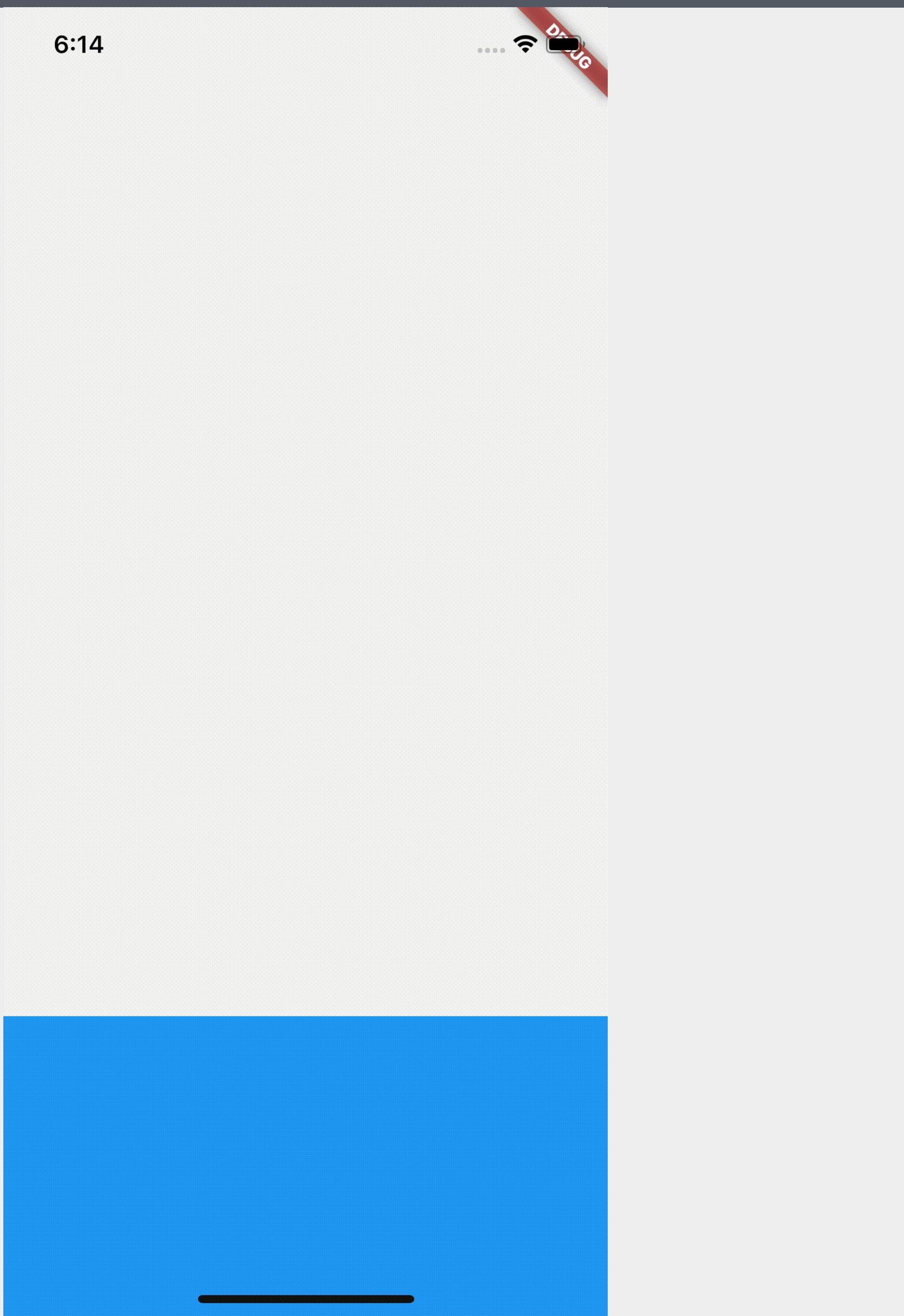
Draggable Scrollable Sheet

- Flutter has one 🎉



Independent animations approach

- Starting an independent animation when the sheet is being dragged
- We don't know how long the user interaction will continue
=> How long should the animation run?
- What happens if the user changes the direction?
=> Synchronisation needed between two animations
- Animations will not be in sync with the interaction speed

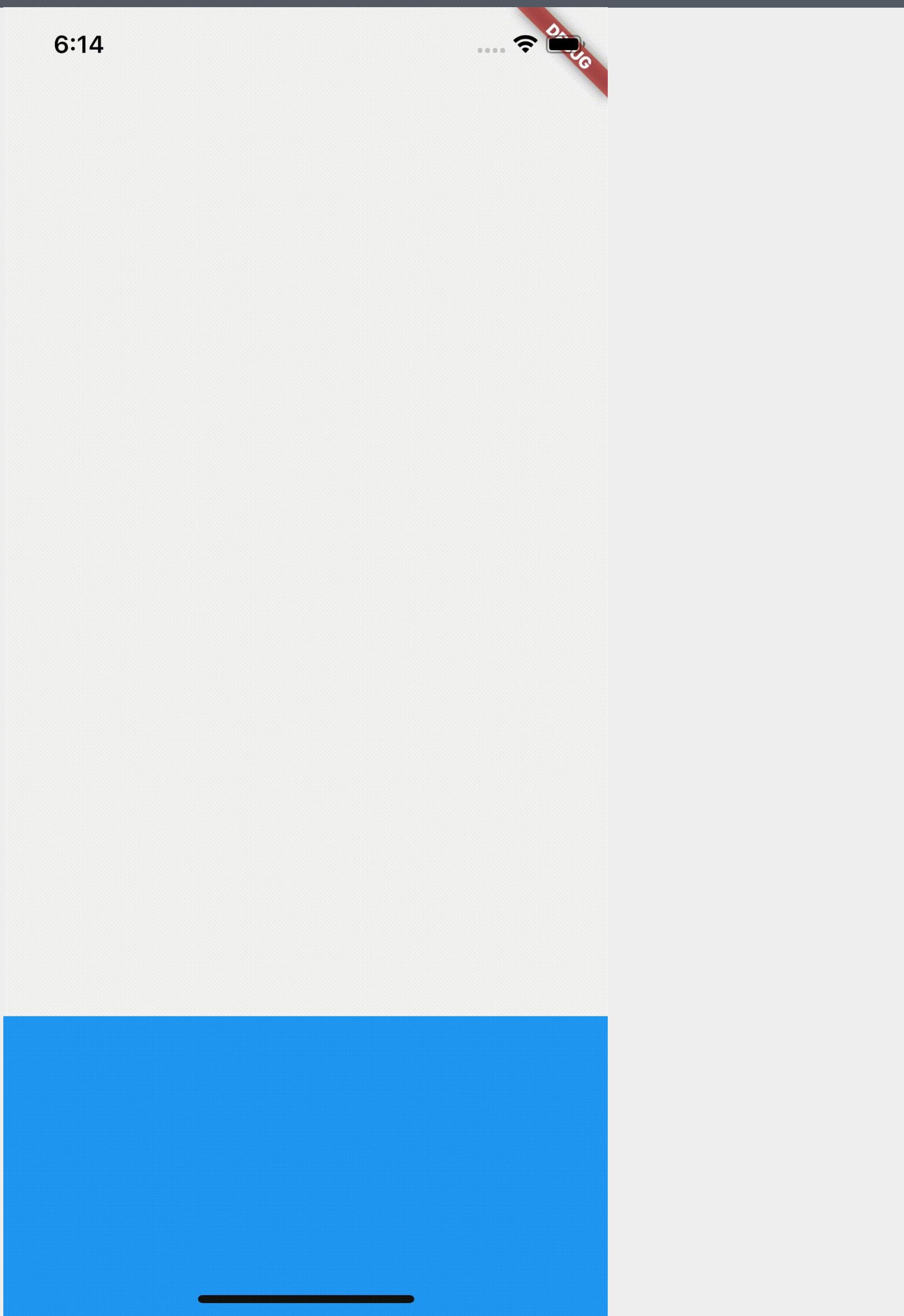


Solution: No animations at all

- No Synchronisation between interaction and animations needed
- “Animations” react to the speed of the user interaction
- Code for the “animations” is minimal and easy to understand

```
return ValueListenableBuilder<double>(  
    valueListenable: measurements.currentSheetExtent,  
    builder: (context, currentExtent, child) {  
        final range = measurements.maxSheetExtent -  
        measurements.initialSheetExtent;  
  
        final position = currentExtent - measurements.initialSheetExtent;  
  
        final percentage = position / range;  
  
        return Container(  
            color: Colors.black.withOpacity(  
                0.7 * percentage,  
            ),  
        );  
    },  
);
```

Let's see some code



Driving Animations through Animation

- Moving the sheet to its initial position

```
final controller = AnimationController(  
    duration: Duration(milliseconds: 200),  
    vsync: this,  
) ;  
  
final animation = Tween<double>(  
  
    begin: measurements.currentSheetExtent.value,  
    end: measurements.initialSheetExtent,  
) .animate(controller);  
  
animation.addListener( () {  
    measurements.currentSheetExtent.value = animation.value;  
});  
  
controller.forward();
```

Let's see some code

Thanks!

Q & A

- From React Native to Flutter
- State Management
- Golden File Testing
- Developing Interactions

Slides & samples:

tinyurl.com/flutter-hh-1

1. Topic A

eirmod tempor invidunt ut labore et dolore magna aliquyam voluptua.

2. Topic B

eirmod tempor invidunt ut labore et dolore magna oluptua.

3. Topic C

eirmod tempor invidunt ut labore et dolore magna aliquya voluptua.

4. Topic D

eirmod tempor invidunt ut labore et dolore magd diam voluptua.

5. Topic E

tempor invidunt erat, sed diam voluptua.

**Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy eirmod
tempor invidunt ut labore et dolore.**

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor

**Lore*m* ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod
tempor invidunt ut labore et dolore.**

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum.
 Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

**Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy eirmod
tempor invidunt ut labore et dolore.**

- Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor

**Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy eirmod
tempor invidunt ut labore et dolore.**

- Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy

1.

 Lorem ipsum dolor sit
 amet, consetetur
 sadipscing elitr, sed
 diam nonumy eirmod
 tempor invidunt ut
 labore et.

2.

 Lorem ipsum dolor sit
 amet, consetetur
 sadipscing elitr, sed
 diam nonumy eirmod
 tempor invidunt ut
 labore et.

3.

 Lorem ipsum dolor sit
 amet, consetetur
 sadipscing elitr, sed
 diam nonumy eirmod
 tempor invidunt ut
 labore et.

4.

 Lorem ipsum dolor sit
 amet, consetetur
 sadipscing elitr, sed
 diam nonumy eirmod
 tempor invidunt ut
 labore et.

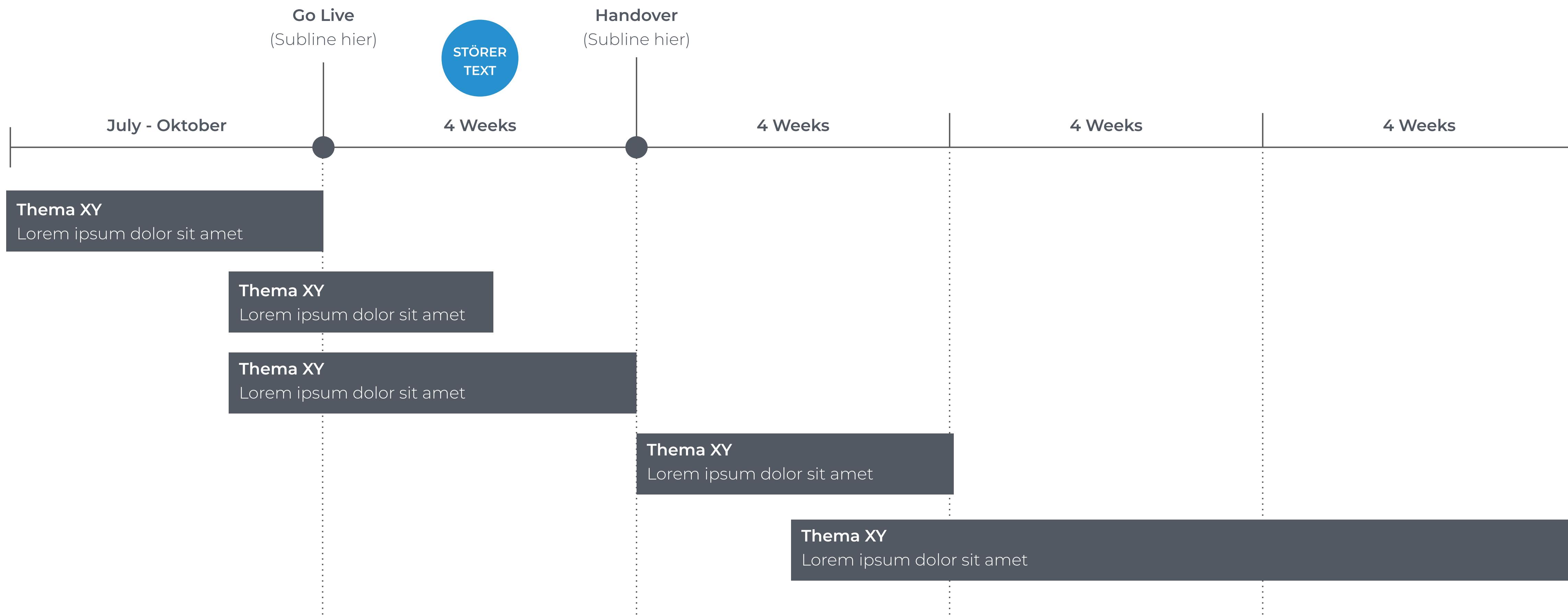
Lorem ipsum dolor sit amet, consetetur sadipscing sit ameitr.



Lorem ipsum dolor sit amet, consetetur sadipscing sit ameitr.

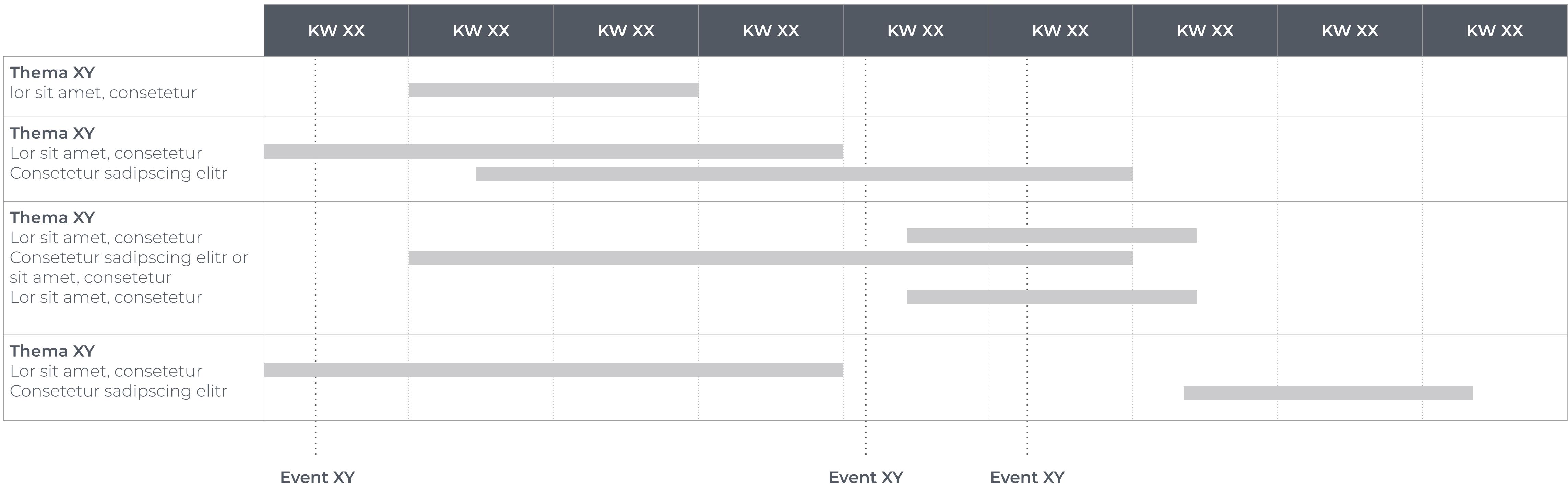


Lore ipsum dolor sit amet, consetetur sadipscing sit amet, consetetur sadipscing elitr

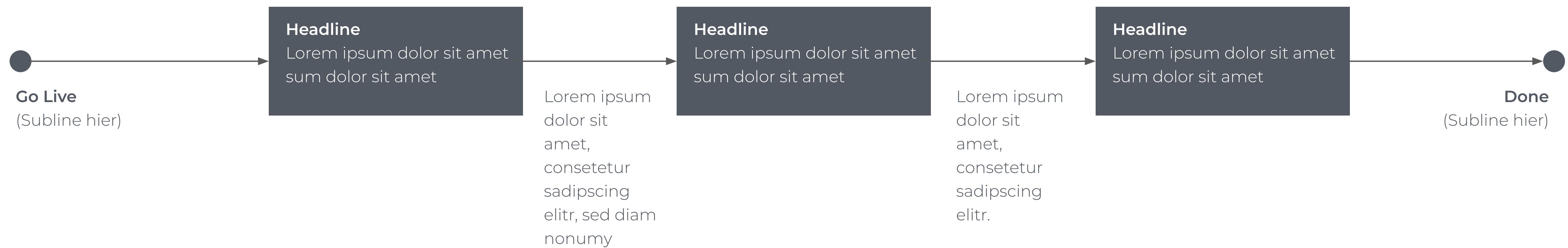


Lorem ipsum dolor sit amet, consetetur sadipscing sit amet, consetetur sadipscing elitr

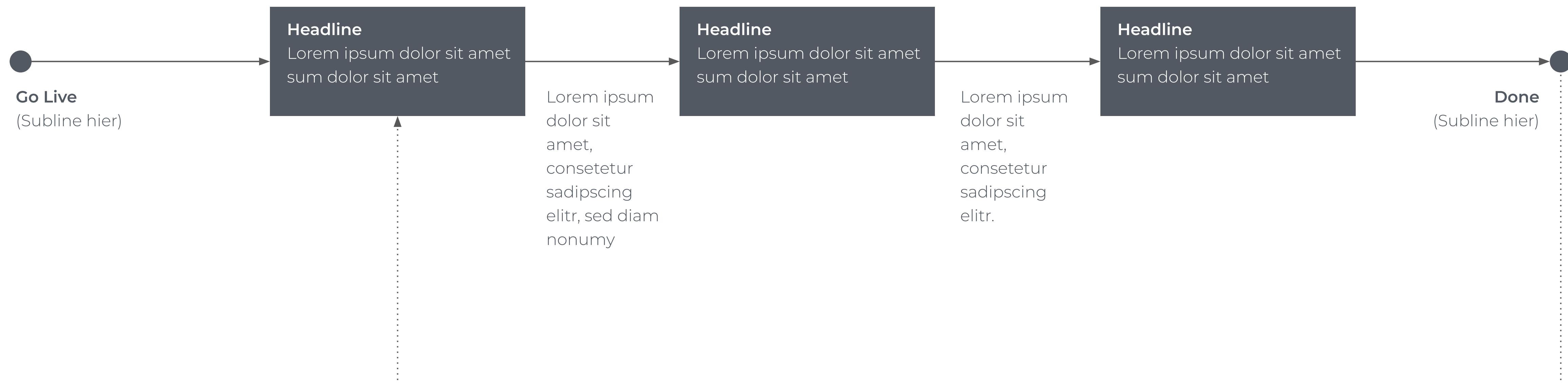
STÖRER
TEXT



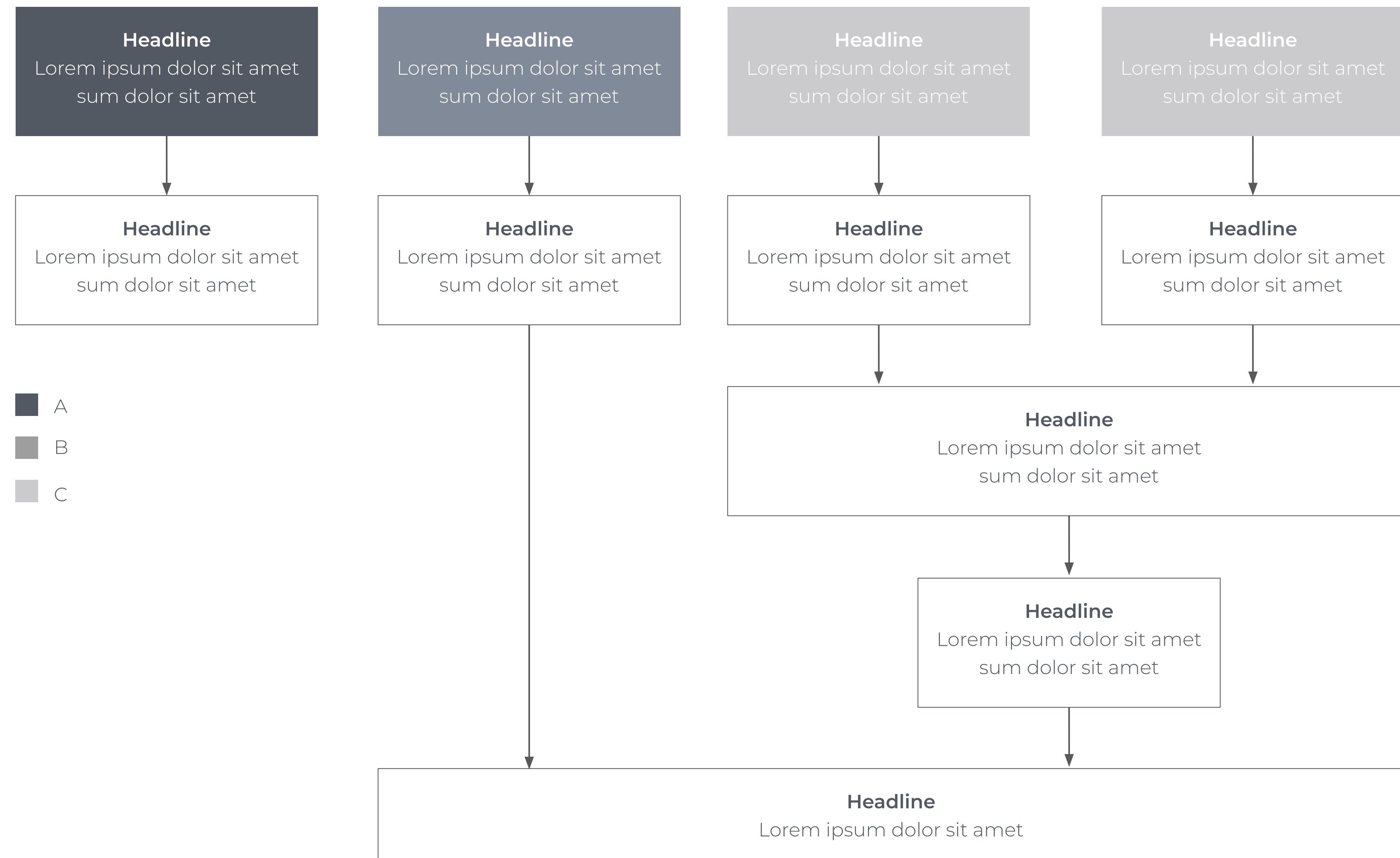
Lorem ipsum dolor sit amet, consetetur sadipscing sit amet, consetetur sadipscing elitr



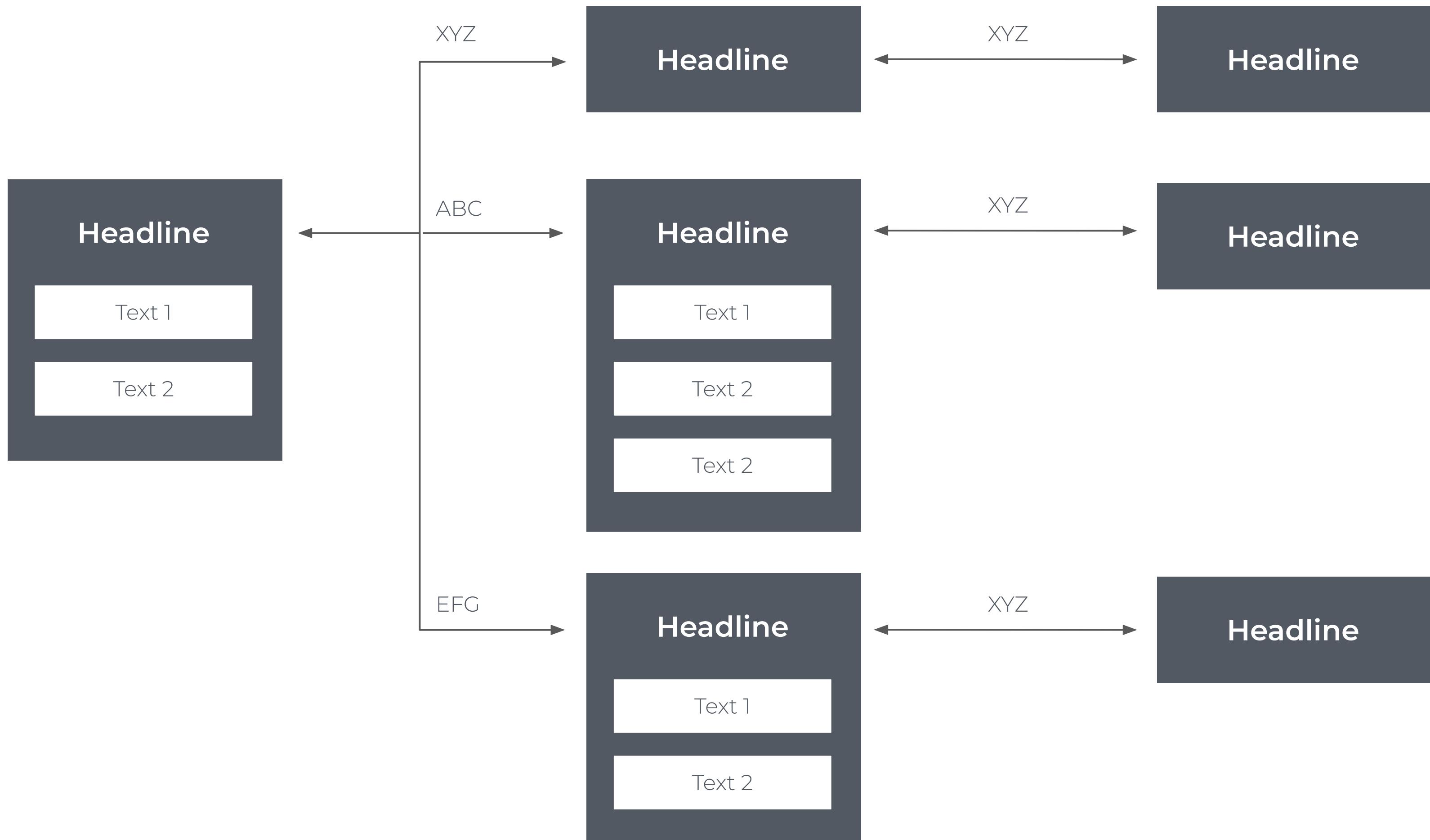
Placeholder text: Lorem ipsum dolor sit amet, consetetur sadipscing sit amet, consetetur sadipscing elitr



Placeholder text: Lorem ipsum dolor sit amet, consetetur sadipscing elitr.



Lorem ipsum dolor sit amet, consetetur sadipscing sit amet, consetetur sadipscing elitr.



Option 1:

**LOREM IPSUM DOLOR SIT AME OREM
IPSUM DOLOR SIT AME.**

eirmod tempor invidunt ut labore et
dolore magna aliquyam erat, sed
diam voluptua.

Option 2:

**LOREM IPSUM DOLOR SIT AME OREM
IPSUM DOLOR SIT AME. LOREM IPSUM
DOLOR SIT AME OREM IPSUM DOLOR SIT
AME.**

eirmod tempor invidunt ut labore et
dolore magna aliquyam erat, sed
diam voluptua.

Option 3:

**LOREM IPSUM DOLOR SIT AME OREM
IPSUM DOLOR SIT AME.**

eirmod tempor invidunt ut labore et
dolore magna aliquyam erat, sed
diam voluptua.

Option 1:

- + eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
- + e et dolore magna aliquyam erat, sed diam voluptua.
- + eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

- eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
- eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Option 2:

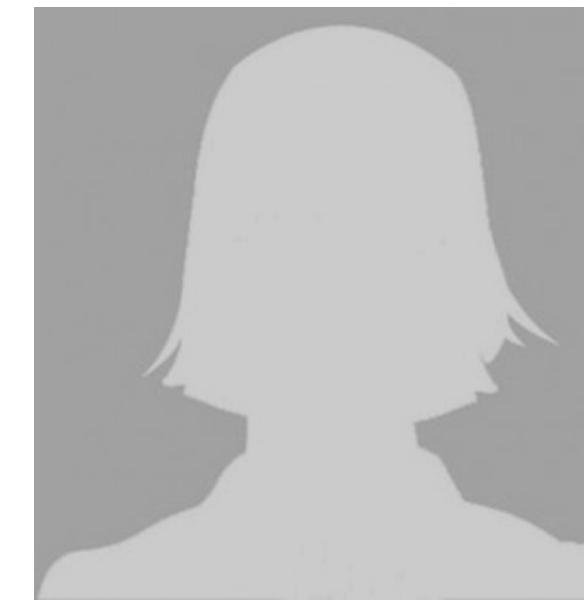
- + eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
- + e et dolore magna aliquyam erat, sed diam voluptua.

- eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.
- eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Lorem ipsum dolor sit amet, consetetur sadipscing sit amet, consetetur sadipscing elitr.



Role: **No sea takimata sanctus est**
Name: Vorname Nachname
Position: Rem ipsum dolor sit



Role: **No sea takimata sanctus est**
Name: Vorname Nachname
Position: Rem ipsum dolor sit



Role: **No sea takimata sanctus est**
Name: Vorname Nachname
Position: Rem ipsum dolor sit



Role: **No sea takimata sanctus est**
Name: Vorname Nachname
Position: Rem ipsum dolor sit



Role: **No sea takimata sanctus est**
Name: Vorname Nachname
Position: Rem ipsum dolor sit



Role: **No sea takimata sanctus est**
Name: Vorname Nachname
Position: Rem ipsum dolor sit



Role: **No sea takimata sanctus est**
Name: Vorname Nachname
Position: Rem ipsum dolor sit

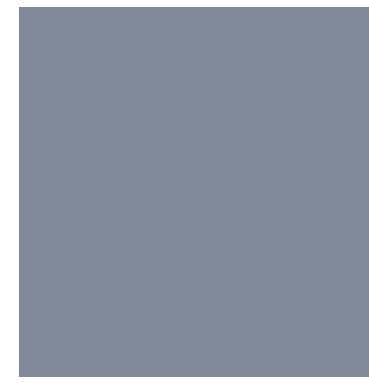


Role: **No sea takimata sanctus est**
Name: Vorname Nachname
Position: Rem ipsum dolor sit

Please use these colors:



#5259621
(Text)



#808a99ff



#cbcfcfff



Only use for:
AY TECH
AY CLOUD
AY Backbone

#2790cf
(Störer)

Please use these Fonts:

Montserrat Light
Montserrat Semibold

Standard icons - can be used for all topics:



Icons - Only use for AY Tech & Backbone (can be mixed with all grey icons)

