

Building Widgets as a Team

A quick-fire guide to improving
Flutter team productivity to build
better apps.

Full-time employee at
bloom technologies
since 2023

Flutter & Dart
GDE
since 2019

Maintaining
4 packages
on pub.dev

**Senior Software
Consultant**

**Jeroen
Meijer**

Worked on
10+ projects
for enterprise customers

6 years
of Flutter experience

Open source
contributor for 5+ years

Spoken at
8 conferences
and smaller meetups

UI Design

Tooling

- Use tools like Figma, not screenshots
- Accurate measurements
- Easier to edit and iterate
- Allows developers to extract data
- Allows for interactive storyboarding

Design Principles

- Makes it easier for developers to implement designs
- Consistent colors, fonts, sizes, and spacing
- Use pre-existing and native components where possible
- Use components and UI elements consistently

Custom UI Libraries (for designers)

- Create a custom pre-defined set of components
- Text fields, buttons, cards, lists, ...
- Reference these in your designs as much as possible
- When one's missing, create a new one (sparingly)

Custom UI Libraries (for developers)

- Create pixel-perfect recreations of your designs using exact specifications
- Store icons, colors, anything
- Leverage Flutter's ThemeData and BuildContext systems

Creating a UI package

1: Using ThemeData

- Use Flutter's own theme properties (`primaryColor`, `fontFamily`)
- For more control over colors, use `colorScheme`
- Set elevations, shapes, border radii, and more
- Create a custom theme class for easier editing, multiple color scheme and user-defined theming support

Creating a UI package

2: Using composed widgets

- Compose custom widgets using Flutter's built-in components
- Use Flutter's theming system as much as possible
- Create properties in your custom theme class

Creating a UI package

3: Completely custom

- Use raw painters and render objects to create custom components from scratch
- Use this approach sparingly
- Maintain compatibility with Flutter's theme and your custom theme class





























































Creating a UI package

Always use Widget arguments

- Makes all of your widgets more flexible
- Little to no downsides
- Avoids the need to refactor later,
reduced chance of “parameter hell”

Text Styling

- Use a consistent text theme
- Even better: use Flutter's text theme
- All Flutter and custom widgets get your brand's styling — for free

 Regular	 57	 64	 0	Display Large
 Regular	 45	 52	 0	Display Medium
 Regular	 36	 44	 0	Display Small
 Regular	 32	 40	 0	Headline Large
 Regular	 28	 36	 0	Headline Medium
 Regular	 24	 32	 0	Headline Small
 Medium	 22	 28	 0	Title Large
 Medium	 16	 24	 0.15	Title Medium
 Medium	 14	 20	 0.1	Title Small
 Medium	 14	 20	 0.1	Label Large
 Medium	 12	 16	 0.5	Label Medium
 Medium	 11	 16	 0.5	Label Small
 Regular	 16	 24	 0.15	Body Large
 Regular	 14	 20	 0.25	Body Medium
 Regular	 12	 16	 0.4	Body Small

UI Gallery

- Great playground for developers and designers
- Build components in many scenarios in isolation
- Preview and verify widget integrity

CI/CD

Private builds

- Let your employees preview your app
- Perform internal testing and quality checks
- Upload and test anything — feature previews, UI gallery, debug apps

Public builds

- Automatic tests and builds to upload to Google Play and the App Store
- Saves time, decreases effort, reduces mental load for everyone, avoids errors

State Management

Why

- Requirements and complexity grow
- Data that flows through your app must be managed properly
- “State management” describes the process and approaches for data in your application

How to pick

- Familiarity
- Simplicity
- Scalability
- Performance
- Testability
- Community support
- Documentation

Some options

Riverpod

Provider

Bloc

MobX

Redux

setState

Fish-Redux

Rx

solidart

flutter_reactive_value

Thank you!

Have any questions or just wanna chat?
Come talk to me after the session or
reach out online anytime.