**Flutter 100% Responsive Design Cheat Sheet**

**Core Responsive Principles**

**1. Screen Size Detection & Breakpoints**

dart

*// Define breakpoints in a constants file*

class Breakpoints {

static const double mobile = 600;

static const double tablet = 900;

static const double desktop = 1200;

}

*// Create responsive helper*

class Responsive {

static bool isMobile(BuildContext context) =>

MediaQuery.of(context).size.width < Breakpoints.mobile;

static bool isTablet(BuildContext context) =>

MediaQuery.of(context).size.width >= Breakpoints.mobile &&

MediaQuery.of(context).size.width < Breakpoints.tablet;

static T value<T>(BuildContext context, {

required T mobile,

T? tablet,

T? desktop,

}) {

if (isMobile(context)) return mobile;

if (isTablet(context)) return tablet ?? mobile;

return desktop ?? tablet ?? mobile;

}

}

**2. Flexible Layouts Instead of Fixed Sizes**

**Never use fixed pixel values for layouts. Always use:**

* Flexible and Expanded widgets in Row/Column
* FractionallySizedBox for percentage-based sizing
* AspectRatio for maintaining proportions
* LayoutBuilder for constraint-based layouts

dart

*// DON'T*

Container(width: 350, height: 200)

*// DO*

FractionallySizedBox(

widthFactor: 0.9,

child: AspectRatio(

aspectRatio: 16/9,

child: Container(),

),

)

**3. Responsive Text Scaling**

dart

*// Create scalable text sizes*

class AppTextStyles {

static double getScaledSize(BuildContext context, double baseSize) {

final width = MediaQuery.of(context).size.width;

*// Scale between 0.8x and 1.2x based on screen width*

final scaleFactor = (width / 375).clamp(0.8, 1.2);

return baseSize \* scaleFactor;

}

static TextStyle headline(BuildContext context) => TextStyle(

fontSize: getScaledSize(context, 24),

fontWeight: FontWeight.bold,

);

}

*// Alternative: Use textScaleFactor*

MediaQuery(

data: MediaQuery.of(context).copyWith(

textScaleFactor: MediaQuery.of(context).size.width / 375,

),

child: YourWidget(),

)

**4. Safe Area & System UI Handling**

dart

*// Always wrap your main content*

SafeArea(

child: Scaffold(

body: Container(

padding: EdgeInsets.only(

bottom: MediaQuery.of(context).viewInsets.bottom, *// Keyboard*

),

child: YourContent(),

),

),

)

**5. Responsive Padding & Spacing**

dart

class AppSpacing {

static EdgeInsets screenPadding(BuildContext context) {

final width = MediaQuery.of(context).size.width;

return EdgeInsets.symmetric(

horizontal: width < 600 ? 16 : 24,

vertical: 16,

);

}

static double dynamicSpacing(BuildContext context, double base) {

final width = MediaQuery.of(context).size.width;

return base \* (width / 375).clamp(0.8, 1.5);

}

}

**Performance Optimization**

**6. Image Optimization**

dart

*// Use cached\_network\_image with proper sizing*

CachedNetworkImage(

imageUrl: url,

memCacheWidth: (MediaQuery.of(context).size.width \*

MediaQuery.of(context).devicePixelRatio).round(),

placeholder: (context, url) => Shimmer.fromColors(...),

errorWidget: (context, url, error) => Icon(Icons.error),

)

*// For local images, use proper resolution variants*

*// Put images in: 1.0x/, 2.0x/, 3.0x/ folders*

**7. List Performance**

dart

*// For long lists, always use builder constructors*

ListView.builder(

itemCount: items.length,

itemExtent: 80, *// Fixed height improves performance*

cacheExtent: 200, *// Pre-render buffer*

itemBuilder: (context, index) {

return ListTile(...);

},

)

*// Use const constructors where possible*

const Padding(

padding: EdgeInsets.all(8.0),

child: Text('Static content'),

)

**8. Orientation Handling**

dart

class ResponsiveGrid extends StatelessWidget {

@override

Widget build(BuildContext context) {

return OrientationBuilder(

builder: (context, orientation) {

return GridView.builder(

gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(

crossAxisCount: orientation == Orientation.portrait ? 2 : 4,

childAspectRatio: 1.0,

crossAxisSpacing: 10,

mainAxisSpacing: 10,

),

itemBuilder: (context, index) => YourWidget(),

);

},

);

}

}

**Advanced Responsive Patterns**

**9. Responsive Navigation**

dart

class AdaptiveNavigation extends StatelessWidget {

@override

Widget build(BuildContext context) {

final isTablet = MediaQuery.of(context).size.width >= 600;

if (isTablet) {

return Row(

children: [

NavigationRail(...), *// Side navigation for tablets*

Expanded(child: ContentArea()),

],

);

}

return Scaffold(

body: ContentArea(),

bottomNavigationBar: BottomNavigationBar(...), *// Bottom nav for phones*

);

}

}

**10. Platform-Specific Adaptations**

dart

import 'dart:io' show Platform;

import 'package:flutter/foundation.dart' show kIsWeb;

class PlatformAware extends StatelessWidget {

@override

Widget build(BuildContext context) {

*// Adjust UI density based on platform*

final visualDensity = kIsWeb

? VisualDensity.compact

: Platform.isIOS || Platform.isAndroid

? VisualDensity.standard

: VisualDensity.comfortable;

return Theme(

data: Theme.of(context).copyWith(

visualDensity: visualDensity,

),

child: YourWidget(),

);

}

}

**11. Text Overflow Prevention**

dart

*// Always handle text overflow*

Flexible(

child: Text(

longText,

maxLines: 2,

overflow: TextOverflow.ellipsis,

),

)

*// For responsive font sizes with auto-sizing*

AutoSizeText(

'Your text',

minFontSize: 12,

maxFontSize: 20,

maxLines: 2,

overflow: TextOverflow.ellipsis,

)

**12. Touch Target Sizing**

dart

*// Ensure minimum touch targets (48x48 dp)*

InkWell(

onTap: () {},

child: Container(

constraints: BoxConstraints(

minHeight: 48,

minWidth: 48,

),

child: Icon(Icons.close, size: 24),

),

)

**Testing Responsive Design**

**13. Device Preview Setup**

dart

*// Use device\_preview package for testing*

void main() {

runApp(

DevicePreview(

enabled: !kReleaseMode,

builder: (context) => MyApp(),

),

);

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

useInheritedMediaQuery: true,

locale: DevicePreview.locale(context),

builder: DevicePreview.appBuilder,

home: HomePage(),

);

}

}

**Critical Performance Rules**

1. **Never rebuild unnecessarily**: Use const constructors, split widgets, and ValueListenableBuilder
2. **Lazy load content**: Use ListView.builder, pagination, and lazy loading for images
3. **Optimize animations**: Keep them at 60fps, use AnimatedBuilder, avoid rebuilding entire trees
4. **Monitor performance**: Use Flutter DevTools, track frame rendering times
5. **Test on low-end devices**: Always test on devices with 2GB RAM or less

**Package Recommendations**

* flutter\_screenutil: For size adaptation
* responsive\_builder: For responsive layouts
* auto\_size\_text: For responsive text
* cached\_network\_image: For image caching
* device\_preview: For testing responsiveness

This cheat sheet ensures your Flutter app will adapt seamlessly across all mobile devices, maintaining smooth performance and excellent user experience regardless of screen size, resolution, or device capabilities.

Retry

[Claude can make mistakes.   
Please double-check responses.](https://support.anthropic.com/en/articles/8525154-claude-is-providing-incorrect-or-misleading-responses-what-s-going-on)

Research

Opus 4.1