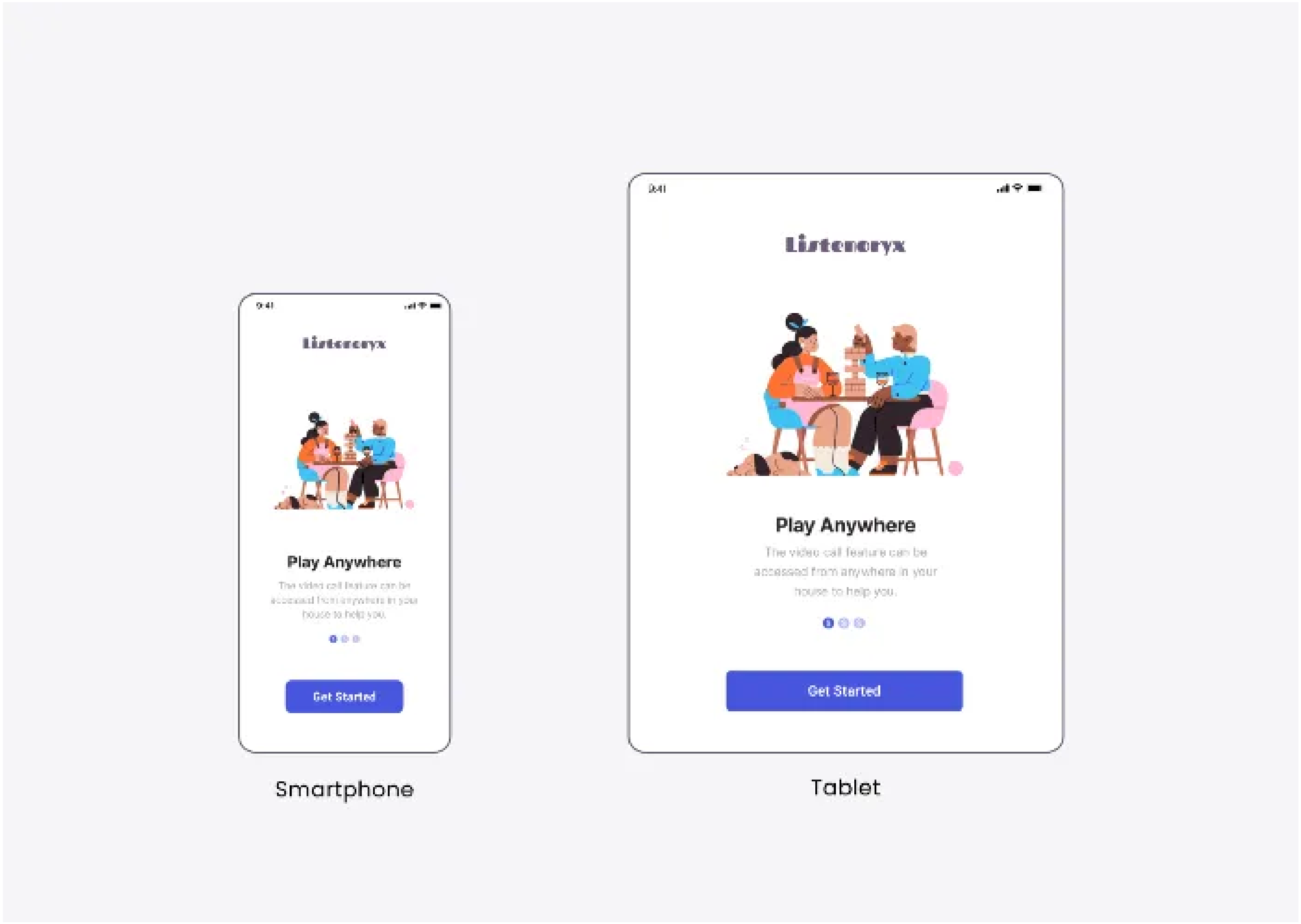
Yasir Quyoom Follow

Feb 28 · 5 min read · Listen

Save

How to make responsive app in flutter ?


Listenoryx
Play Anywhere
The video call feature can be accessed from anywhere in your house to help you.
Get Started
Smartphone
Tablet

In this article i will show you how to build responsive app without overflowing widgets.

I am sharing my experiences knowledge with you , to make a Flutter app responsive for all devices without overflow errors, you can use a combination of Flutter widgets and layout techniques. Here is an example function that you can use as a starting point:

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

Widget buildResponsiveApp() {
  return LayoutBuilder(
    builder: (BuildContext context, BoxConstraints constraints) {
      return SingleChildScrollView(
        child: ConstrainedBox(
          constraints: BoxConstraints(
            minHeight: constraints.maxHeight,
          ),
          child: IntrinsicHeight(
            child: Column(
              crossAxisAlignment: CrossAxisAlignment.stretch,
              children: [
                // Your app content here
              ],
            ),
          ),
        ),
      );
    },
  );
}
```

This function uses the `LayoutBuilder` widget to get the size constraints of the parent widget, and applies those constraints to a `ConstrainedBox` widget to ensure that the content doesn't overflow the available space. The `SingleChildScrollView` widget allows the user to scroll if the content is too large to fit on the screen. Finally, the `IntrinsicHeight` widget ensures that all of the children of the `Column` widget are sized to fit the available space.

You can add your app content inside the `Column` widget, and use responsive Flutter widgets like `Expanded`, `Flexible`, and `AspectRatio` to ensure that your content looks good on all devices. For example, you might use an `Expanded` widget to make a list fill the available vertical space:

```
Column(
  crossAxisAlignment: CrossAxisAlignment.stretch,
  children: [
    Expanded(
      child: ListView.builder(
        itemCount: 100,
        itemBuilder: (BuildContext context, int index) {
          return ListTile(
            title: Text('Item $index'),
          );
        },
      ),
    ),
  ],
),
```

By using these responsive widgets and layout techniques, you can build a Flutter app that looks great on all devices without overflowing the screen.

Another method

To build a Flutter app that looks great on all devices without overflowing the containers, widgets, screens, you can follow these steps:

1. Use responsive widgets: Flutter provides many responsive widgets that adjust their size and position based on the available screen size. Some examples include `Expanded` , `Flexible` , `AspectRatio` , and `FractionallySizedBox` . Using these widgets can help you build a layout that adapts to different screen sizes without overflowing the containers.

2. Use constraints: Use constraints to define the maximum and minimum sizes of widgets. For example, you can use the `BoxConstraints` class to define a maximum height for a widget. This will prevent the widget from overflowing its container.

3. Use media queries: Use `MediaQuery.of(context)` to get information about the screen size and adjust your layout accordingly. For example, you might use a larger font size on larger screens to improve readability.

4. Use scrollable widgets: If you have a lot of content that doesn't fit on the screen, use a scrollable widget like `ListView`, `GridView`, or `SingleChildScrollView`. These widgets allow the user to scroll through the content and prevent overflow errors.

5. Test on different devices: Make sure to test your app on different devices to ensure that it looks great and doesn't overflow containers, widgets, or screens. You can use emulators or physical devices to test your app on different screen sizes.

Here is an example code snippet that demonstrates the use of `Expanded`, `Flexible`, and `MediaQuery` to build a responsive layout:

```
Column(
  children: [
    Expanded(
      child: Container(
        color: Colors.blue,
        child: Center(
          child: Text(
            'Header',
            style: TextStyle(
              fontSize: MediaQuery.of(context).size.width * 0.1,
              color: Colors.white,
            ),
          ),
        ),
      ),
    ),
    Flexible(
      child: SingleChildScrollView(
        child: Column(
          children: [
            Container(
              height: MediaQuery.of(context).size.width * 0.3,
              color: Colors.green,
            ),
            Container(
              height: MediaQuery.of(context).size.width * 0.3,
              color: Colors.yellow,
            ),
            Container(
              height: MediaQuery.of(context).size.width * 0.3,
              color: Colors.orange,
                  ),
                ],
              ),
            ),
          ),
          Expanded(
            child: Container(
              color: Colors.blue,
              child: Center(
                child: Text(
                  'Footer',
                  style: TextStyle(
                    fontSize: MediaQuery.of(context).size.width * 0.1,
                    color: Colors.white,
                  ),
                ),
              ),
            ),
          ),
        ],
      ),
```

Another method

```
    return screenWidth(context, dividedBy: dividedBy) / 100;
  }

  double blockSizeVertical(BuildContext context, {double dividedBy = 1}) {
    return screenHeight(context, dividedBy: dividedBy) / 100;
  }

  double fontSize(BuildContext context, double size) {
    return size * blockSizeHorizontal(context);
  }

  double iconSize(BuildContext context, double size) {
    return size * blockSizeHorizontal(context);
  }

  EdgeInsets margin(BuildContext context, double left, double top, double right, doubl
    return EdgeInsets.fromLTRB(
        left * blockSizeHorizontal(context),
        top * blockSizeVertical(context),
        right * blockSizeHorizontal(context),
        bottom * blockSizeVertical(context));
  }
```

This function provides a set of helpers for responsive design in Flutter. Here’s how you can use them:

- `screenWidth` and `screenHeight` return the width and height of the screen, respectively. You can use them to size your widgets dynamically.
- `blockSizeHorizontal` and `blockSizeVertical` return the horizontal and vertical size of a "block" on the screen, respectively. You can use them to calculate sizes based on a percentage of the screen size.
- `fontSize` and `iconSize` take a `size` parameter and return a font size or icon size that's proportional to the screen size. You can use them to ensure that your text and icons scale correctly on different devices.
- `margin` takes four parameters for the left, top, right, and bottom margins and returns an `EdgeInsets` object that you can use to set the margins of your widgets.

Another method

To build a responsive function in Flutter, you can use the `MediaQuery` class, which provides information about the device's screen size and orientation.

Here is an example of a responsive function that uses `MediaQuery` to calculate the appropriate padding, size, and spacing between widgets:

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

double screenWidth(BuildContext context) {
  return MediaQuery.of(context).size.width;
}

double screenHeight(BuildContext context) {
  return MediaQuery.of(context).size.height;
}

double blockSizeHorizontal(BuildContext context) {
  return screenWidth(context) / 100;
}

double blockSizeVertical(BuildContext context) {
  return screenHeight(context) / 100;
}

double fontSize(BuildContext context, double size) {
  return size * blockSizeHorizontal(context);
}

double padding(BuildContext context, double padding) {
  return padding * blockSizeHorizontal(context);
}

double margin(BuildContext context, double margin) {
  return margin * blockSizeHorizontal(context);
}

class ResponsiveWidget extends StatelessWidget {
  final Widget child;
  final double padding;
  final double margin;
  final double width;
  final double height;

  const ResponsiveWidget({
    Key? key,
    required this.child,
    this.padding = 0,
    this.margin = 0,
    this.width = 100,
    this.height = 100,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      width: width * blockSizeHorizontal(context),
      height: height * blockSizeVertical(context),
      margin: EdgeInsets.all(margin(context, margin)),
      padding: EdgeInsets.all(padding(context, padding)),
      child: child,
    );
  }
}
```

In this example, we define several helper functions that use `MediaQuery` to calculate values based on the screen size. The `ResponsiveWidget` class takes in several parameters that allow us to customize the size and spacing of the widget, and it uses the helper functions to calculate the appropriate values based on the screen size.

To use the `ResponsiveWidget`, you can simply wrap your existing widgets with it:

```
ResponsiveWidget(
  padding: 2,
  margin: 2,
  width: 50,
  height: 50,
  child: Container(
    color: Colors.red,
  ),
),
```

Using these helpers, you can make your Flutter app responsive and ensure that it looks great on all devices.

Flutter Development Programing