flutter_screenutil 屏幕适配方案

Eyes_cc

1、什么是 flutter_screenutil

gitHub https://github.com/OpenFlutter/flutter_screenutil/blob/master/README_CN.md
csdn 博客工具介绍

是一个屏幕适配方案，让你的 UI 在不同尺寸的屏幕上都能显示合理的布局!

注意：此插件仍处于开发阶段，某些 API 可能尚未推出。

2、使用方法:

安装依赖:

安装之前请查看最新版本 新版本如有问题请使用上一版

```
dependencies:
  flutter:
    sdk: flutter
  # 添加依赖
  flutter_screenutil: ^{latest version}
```

在每个使用的地方导入包：

```
import 'package:flutter_screenutil/
flutter_screenutil.dart';
```

属性

属性	类型	默认值	描述
designSize	Size	Size(360, 690)	设计稿中设备的尺寸 (单位随意,建议 dp,但在使用过程中必须保持一致)
builder	Widget Function()	Container()	一般返回一个 MaterialApp 类型的 Function()
orientation	Orientation	portrait	屏幕方向
splitScreenMode	bool	true	支持分屏尺寸

初始化设置适配尺寸及字体大小是否根据系统的“字体大小”辅助选项来进行缩放

在使用之前请设置好设计稿的宽度和高度，传入设计稿的宽

度和高度 (单位随意,但在使用过程中必须保持一致) 一定要进行初始化 (只需设置一次),以保证在每次使用之前设置好了适配尺寸:

初始化

方式一:

```
void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    //填入设计稿中设备的屏幕尺寸,单位dp
    return ScreenUtilInit(
      designSize: Size(360, 690),
      builder: () => MaterialApp(
        debugShowCheckedModeBanner: false,
        title: 'Flutter_ScreenUtil',
        theme: ThemeData(
                  primarySwatch: Colors.blue,
                  //要支持下面这个需要使用第一种初始化方式
                  textTheme: TextTheme(
                    button: TextStyle(fontSize: 45.sp)
                  ),
                ),
        home: HomePage(title: 'FlutterScreenUtil Demo'),
      ),
    );
  }
}
```

方式二:

ScreenUtil.init只需在home或者根路由（即第一个flutter页面）中调用一次即可。

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter_ScreenUtil',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: HomePage(title: 'FlutterScreenUtil Demo'),
    );
  }
}

class HomePage extends StatefulWidget {
  const HomePage({Key key, this.title}) : super(key: key);

  final String title;

  @override
  _HomePageState createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  @override
  Widget build(BuildContext context) {
    //设置尺寸（填写设计中设备的屏幕尺寸）如果设计基于360dp * 690dp的屏幕
```

```
    ScreenUtil.init(
        BoxConstraints(
            maxWidth: MediaQuery.of(context).size.width,
            maxHeight: MediaQuery.of(context).size.height),
        designSize: Size(360, 690),
        orientation: Orientation.portrait);
    return Scaffold();
  }
}
```

3、详细 API:

```
    ScreenUtil().setWidth(540)  (sdk>=2.6 : 540.w)   //根据屏幕宽度适配尺寸
    ScreenUtil().setHeight(200) (sdk>=2.6 : 200.h)   //根据屏幕高度适配尺寸(一般根据宽度适配即可)
    ScreenUtil().radius(200)    (sdk>=2.6 : 200.r)   //根据宽度或高度中的较小者进行适配

    ScreenUtil().setSp(24)      (sdk>=2.6 : 24.sp)   //适配字体
    24.sm          // 取 24和 24.sp 中的最小值

    ScreenUtil.pixelRatio       //设备的像素密度
    ScreenUtil.screenWidth   (sdk>=2.6 : 1.sw)   //设备宽度
    ScreenUtil.screenHeight  (sdk>=2.6 : 1.sh)   //设备高度
    ScreenUtil.bottomBarHeight  //底部安全区距离, 适用于全面屏下面有按键的
    ScreenUtil.statusBarHeight  //状态栏高度 刘海屏会更高
    ScreenUtil.textScaleFactor //系统字体缩放比例

    ScreenUtil().scaleWidth  // 实际宽度与设计稿宽度的比例
    ScreenUtil().scaleHeight // 实际高度与设计稿高度的比例

    ScreenUtil().orientation  //屏幕方向
```

```
    0.2.sw  //屏幕宽度的 0.2倍
    0.5.sh  //屏幕高度的 0.5倍
```

4、适配约束:

传入设计稿的尺寸：（单位和初始化时的单位保持一致）

一般来说，不使用`setHeight !!!`，控件高度也根据宽度进行适配，都使用`setWidth(value)`或`value.w`如：

```
Container(
  width: ScreenUtil().setWidth(50), // 根据屏幕宽度适配
  height:ScreenUtil().setWidth(200), // 根据屏幕宽度适配
)
或  '(dart sdk>=2.6)'
Container(
  width: 50.w,  // 根据屏幕宽度适配
  height:200.w // 根据屏幕宽度适配
)
```

当想要一个正方形的时候

```
Container(
  width: ScreenUtil().setWidth(200), // 根据宽度或高度中的较小者进行调整
  height:ScreenUtil().setWidth(200), // 根据宽度或高度中的较小者进行调整
)
或  '(dart sdk>=2.6)'
Container(
```

```
  width: 200.w,  // 根据屏幕宽度适配
  height:200.w // 根据屏幕宽度适配
)
Container(
  width: ScreenUtil().radius(200), // 根据宽度或高度中的较小者进行调整
  height:ScreenUtil().radius(200), // 根据宽度或高度中的较小者进行调整
)
或  '(dart sdk>=2.6)'
Container(
  width: 200.r,  // 根据宽度或高度中的较小者进行调整
  height:200.r // 根据宽度或高度中的较小者进行调整
)
```

5、适配字体:

传入设计稿的字体大小: (单位和初始化时的单位保持一致)

```
ScreenUtil().setSp(28)
或
28.sp '(dart sdk>=2.6)'
```

注意属性 `textScaleFactor` 属性，因为设置了`textScaleFactor`,所以不会随着系统的文字缩放比例变化'，因为没有设置`textScaleFactor`,所以会随着系统的文字缩放比例变化'。

```
// for example:
Column(
              crossAxisAlignment: CrossAxisAlignment.start,
              children: <Widget>[
                Text(
                  '我的文字大小在设计稿上是16dp，因为设置了
`textScaleFactor`,所以不会随着系统的文字缩放比例变化',
                  style: TextStyle(
                    color: Colors.black,
                    fontSize: 16.sp,
                  ),
                  textScaleFactor: 1.0,
                ),
                Text(
                  '我的文字大小在设计稿上是16dp，会随着系统的文字缩
放比例变化',
                  style: TextStyle(
                    color: Colors.black,
                    fontSize: 16.sp,
                  ),
                ),
              ],
            )
```

设置字体不随系统字体大小进行改变

1. APP全局:

```
MaterialApp(
        debugShowCheckedModeBanner: false,
        title: 'Flutter_ScreenUtil',
        theme: ThemeData(
          primarySwatch: Colors.blue,
        ),
        builder: (context, widget) {
```

```
          return MediaQuery(
            ///设置文字大小不随系统设置改变
            data:
MediaQuery.of(context).copyWith(textScaleFactor: 1.0),
            child: widget,
          );
        },
        home: HomePage(title: 'FlutterScreenUtil Demo'),
      ),
```

2. 单独的 Text:

```
Text("text", textScaleFactor: 1.0)
```