



coderwhy Flutter

屏幕适配

flutter

Flutter屏幕适配

原创 coderwhy coderwhy

目前移动端的设备已经非常多，并且不同的设备手机屏幕也不相同。

目前做移动端开发都要针对不同的设备进行一定的适配，无论是移动原生开发、小程序、H5页面。

Flutter中如何针对不同的手机屏幕来进行适配呢？我们一起来聊聊这个话题。

一. Flutter单位

1.1. Flutter中的单位

在进行Flutter开发时，我们通常不需要传入尺寸的单位，那么Flutter使用的是什么单位呢？

- Flutter使用的是类似于iOS中的点pt，也就是point。
- 所以我们经常说iPhone6的尺寸是375x667，但是它的分辨率其实是750x1334。
- 因为iPhone6的dpr (devicePixelRatio) 是2.0，iPhone6plus的dpr是3.0

设备 iPhone	宽 Width	高 Height	对角线 Diagonal	逻辑分辨率 (point)	缩放因子 Scale Factor	物理分辨率 (pixel)	像素密度 PPI
3GS	2.4 inches (62.1 mm)	4.5 inches (115.5 mm)	3.5-inch	320x480	@1x	320x480	163
4/4s	2.31 inches (58.6 mm)	4.5 inches (115.2 mm)	3.5-inch	320x480	@2x	640x960	326
5/5s/se	2.31 inches (58.6 mm)	4.87 inches (123.8 mm)	4-inch	320x568	@2x	640x1136	326
6/7/8	2.64 inches (67.1 mm)	5.44 inches (138.3 mm)	4.7-inch	375x667	@2x	750x1334	326
6P/7P/8P	3.07 inches (77.9 mm)	6.23 inches (158.2 mm)	5.5-inch	414x736	@3x	(1242x2208) 1080x1920	401
X/Xs	2.79inches (70.9mm)	5.65inches (143.6mm)	5.8-inch	375x812	@3x	1125x2436	458
XR	2.98inches (75.7mm)	5.94inches (150.9mm)	6.1-inch	414x896	@2x	828x1792	326
Xs Max	3.05inches (77.4mm)	6.20inches (157.5mm)	6.5-inch	414x896	@3x	1242x2688	 coderwhy 458

iPhone设备参数

在Flutter开发中，我们使用的是对应的逻辑分辨率

1.2. Flutter设备信息

获取屏幕上的一些信息，可以通过`MediaQuery`:

```
// 1. 媒体查询信息
final mediaQueryData = MediaQuery.of(context);

// 2. 获取宽度和高度
final screenWidth = mediaQueryData.size.width;
final screenHeight = mediaQueryData.size.height;
final physicalWidth = window.physicalSize.width;
final physicalHeight = window.physicalSize.height;
final dpr = window.devicePixelRatio;
print("屏幕width:$screenWidth height:$screenHeight");
print("分辨率: $physicalWidth - $physicalHeight");
print("dpr: $dpr");

// 3. 状态栏的高度
// 有刘海的屏幕:44 没有刘海的屏幕为20
final statusBarHeight = mediaQueryData.padding.top;
// 有刘海的屏幕:34 没有刘海的屏幕0
final bottomHeight = mediaQueryData.padding.bottom;
print("状态栏height: $statusBarHeight 底部高度:$bottomHeight")
;
```

获取一些设备相关的信息，可以使用官方提供的一个库：

```
dependencies:
  device_info: ^0.4.2+
1
```

二. 适配方案

2.1. 适配概述

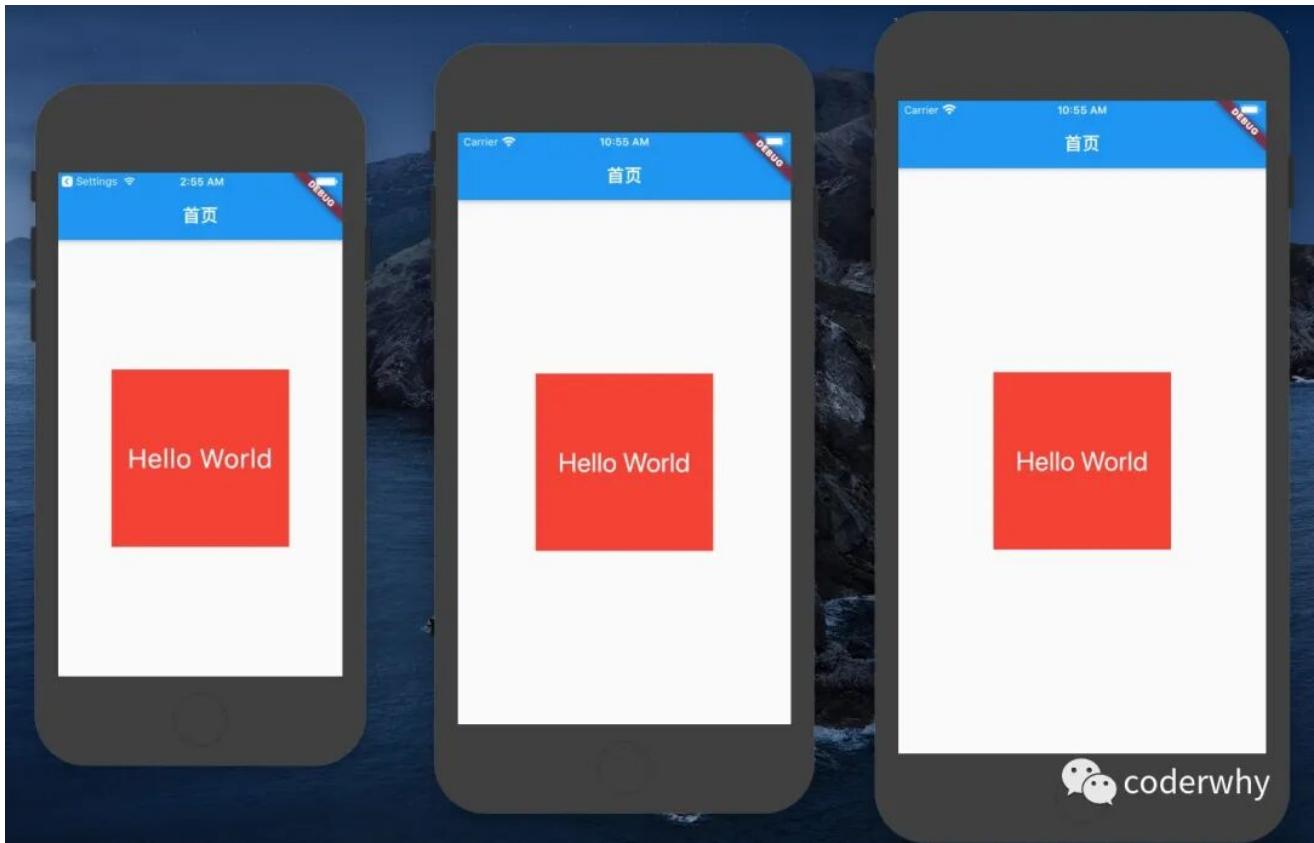
假如我们有下面这样一段代码：

- 在屏幕中间显示一个200*200的Container
- Container中有一段文字是30

```
class HYHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("首页"),
      ),
      body: Center(
        child: Container(
          width: 200,
          height: 200,
          color: Colors.red,
          alignment: Alignment.center,
          child: Text("Hello World", style: TextStyle(fontSize: 30, color: Colors.white),),
        ),
      ),
    );
  }
}
```

上面的代码在不同屏幕上会有不同的表现：

- 很明显，如果按照上面的规则，在iPhone5上面，尺寸过大，在iPhone6plus上面尺寸过小
- 在开发中，我们应该可以根据不同的屏幕来完成尺寸的缩放



不同屏幕表现



在前端开发中，针对不同的屏幕常见的适配方案有下面几种：

- rem：
 - rem是给根标签（HTML标签）设置一个字体大小；
 - 但是不同的屏幕要动画设置不同的字体大小（可以通过媒体查询，也可以通过js动态计算）；
 - 其它所有的单位都使用rem单位（相对于根标签）；
- vw、wh：
 - vw和vh是将屏幕（视口）分成100等份，一个1vw相当于是1%的大小；
 - 其它所有的单位都使用vw或wh单位；
- rpx：
 - rpx是小程序中的适配方案，它将750px作为设计稿， $1rpx = \text{屏幕宽度}/750$ ；
 - 其它所有的单位都使用rpx单位；

这里我采用小程序的rpx来完成Flutter的适配

2.2. rpx适配

小程序中rpx的原理是什么呢？

- 不管是什么屏幕，统一分成750份
- 在iPhone5上： $1rpx = 320/750 = 0.4266 \approx 0.42px$
- 在iPhone6上： $1rpx = 375/750 = 0.5px$
- 在iPhone6plus上： $1rpx = 414/750 = 0.552px$

设备	rpx换算px (屏幕宽度/750)	px换算rpx (750/屏幕宽度)
iPhone5	$1rpx = 0.42px$	$1px = 2.34rpx$
iPhone6	$1rpx = 0.5px$	$1px = 2rpx$
iPhone6 Plus	$1rpx = 0.552px$	$1px = 1.81rpx$

rpx的对应关系

那么我们就可以通过上面的计算方式，算出一个rpx，再将自己的size和rpx单位相乘即可：



- 比如100px的宽度: $100 * 2 * \text{rpx}$
- 在iPhone5上计算出的结果是84px
- 在iPhone6上计算出的结果是100px
- 在iPhone6plus上计算出的结果是110.4px

我们自己来封装一个工具类:

- 工具类需要进行初始化, 传入context
 - 可以通过传入context, 利用媒体查询获取屏幕的宽度和高度
 - 也可以传入一个可选的参数, 以什么尺寸作为设计稿

```
class HYSizeFit {
  static MediaQueryData _mediaQueryData;
  static double screenWidth;
  static double screenHeight;
  static double rpx;
  static double px;

  static void initialize(BuildContext context, {double standardWidth = 750})
  {
    _mediaQueryData = MediaQuery.of(context);
    screenWidth = _mediaQueryData.size.width;
    screenHeight = _mediaQueryData.size.height;
    rpx = screenWidth / standardWidth;
    px = screenWidth / standardWidth * 2;
  }

  // 按照像素来设置
  static double setPx(double size) {
    return HYSizeFit.rpx * size * 2;
  }

  // 按照rpx来设置
  static double setRpx(double size) {
    return HYSizeFit.rpx * size;
  }
}
```

初始化HYSizeFit类的属性:

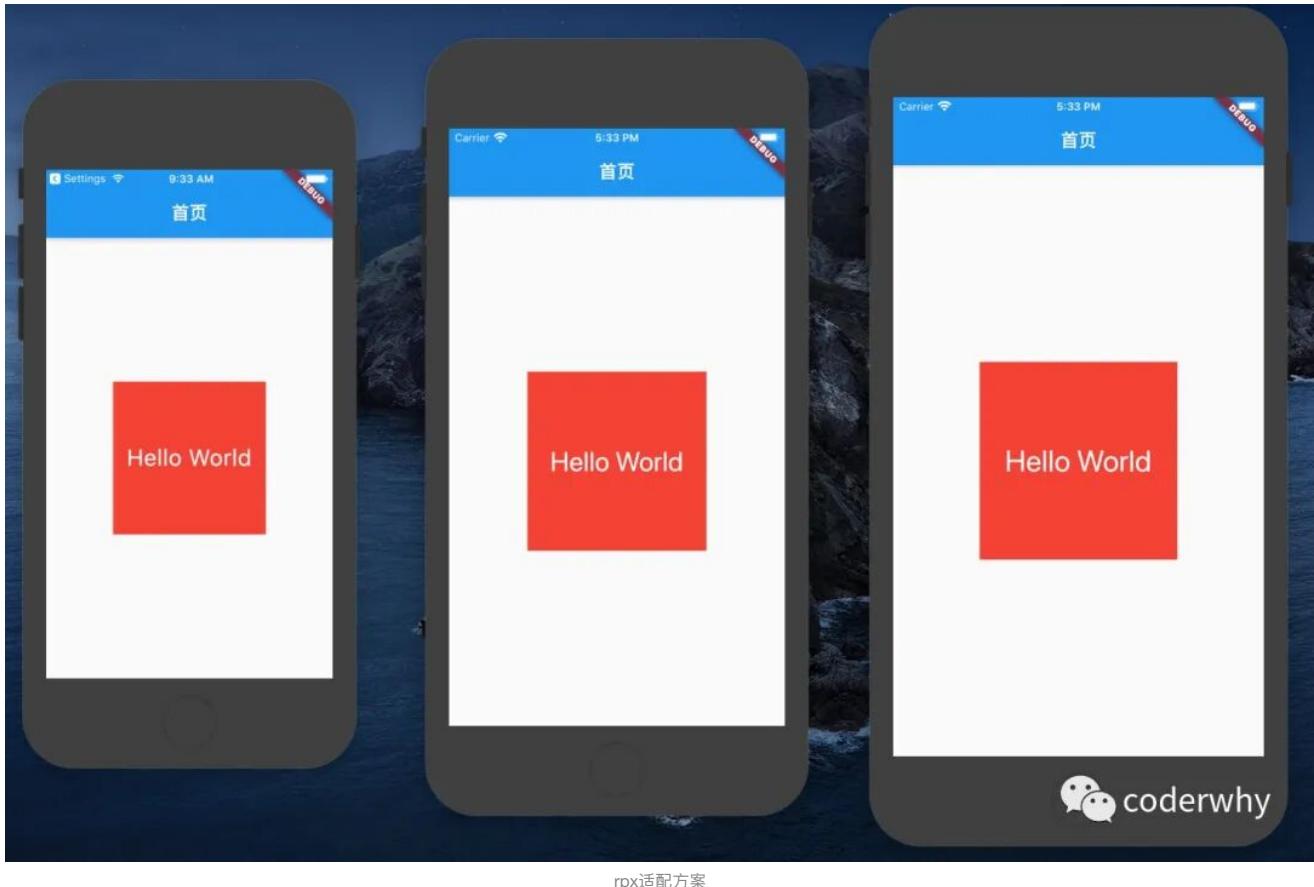
- 注意: 必须在已经有MaterialApp的Widget中使用context, 否则是无效的

```
class HYHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // 初始化HYSizeFit
    HYSizeFit.initialize(context);
    return null;
  }
}
```

使用rpx来完成屏幕适配:

```
class HYHomePage extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    HysizeFit.initialize(context);
    return Scaffold(
      appBar: AppBar(
        title: Text("首页"),
      ),
      body: Center(
        child: Container(
          width: HysizeFit.setPx(200),
          height: HysizeFit.setRpx(400),
          color: Colors.red,
          alignment: Alignment.center,
          child: Text("Hello World", style: TextStyle(fontSize: HysizeFit.setPx(30), color: Colors.white)),
        ),
      ),
    );
  }
}
```

我们来看一下实现效果：



rpx适配方案

屏幕适配也可以使用第三方库：flutter_screenutil

- https://github.com/OpenFlutter/flutter_screenutil

备注：所有内容首发于公众号，之后除了Flutter也会更新其他技术文章，TypeScript、React、Node、uniapp、mpvue、数据结构与算法等等，也会更新一些自己的学习心得等，欢迎大家关注



coderwhy

微信扫描二维码，关注我的公众号



公众号