Flutter 94: 初识 MediaQuery

阿策神奇

当我们同时为手机和平板适配编写 app 针对不同屏幕尺寸进行 UI 布局或当用户偏好设置较大字号或是想要最大限度等减少动画等；此时就需要 MediaQuery 来帮我们获取所用设备的信息以及用户设置的偏好信息； MediaQuery 媒体询问

MediaQuery

MediaQuery 一直存在于 WidgetsApp 和 MaterialApp 中，MediaQuery 继承自 InheritedWidget 是一个单独的 Widget，但一般通过 MediaQuery.of(context) 来获取相关信息；

当相关信息发生变化，例如屏幕旋转等时，屏幕中 Widget 会重新构建，以保持最新状态；我们可以通过 MediaQuery 构造函数和提供的静态方法手动设置对应的相关信息；

1. MediaQuery()

```
const MediaQuery({
    Key key,
    @required this.data,
    @required Widget child,
```

```
})
```

2. MediaQuery.removePadding() 删除内边距

```
factory MediaQuery.removePadding({
    Key key,
    @required BuildContext context,
    bool removeLeft = false,
    bool removeTop = false,
    bool removeRight = false,
    bool removeBottom = false,
    @required Widget child,
})
```

3. MediaQuery.removeViewInsets() 删除视图内边距

```
factory MediaQuery.removeViewInsets({
    Key key,
    @required BuildContext context,
    bool removeLeft = false,
    bool removeTop = false,
    bool removeRight = false,
    bool removeBottom = false,
    @required Widget child,
})
```

MediaQueryData

MediaQueryData 包含关于媒介的相关信息；一般通过

MediaQuery.of(context) 获取；

```
const MediaQueryData({
    this.size = Size.zero,
    this.devicePixelRatio = 1.0,
    this.textScaleFactor = 1.0,
    this.platformBrightness = Brightness.light,
    this.padding = EdgeInsets.zero,
    this.viewInsets = EdgeInsets.zero,
    this.systemGestureInsets = EdgeInsets.zero,
    this.viewPadding = EdgeInsets.zero,
    this.physicalDepth = double.maxFinite,
    this.alwaysUse24HourFormat = false,
    this.accessibleNavigation = false,
    this.invertColors = false,
    this.highContrast = false,
    this.disableAnimations = false,
    this.boldText = false,
});
```

1. size

size 为媒介的尺寸大小，以逻辑像素为单位；

```
print('屏幕 Size -> ${MediaQuery.of(context).size}');

print('按钮 Size -> $
{_itemExpandedKey.currentContext.size}');
print('文字 Size -> ${_itemTextKey.currentContext.size}');
print('文字 Size -> $
{MediaQuery.of(_itemTextKey.currentContext).size}');
```

```
I/flutter (11488): Current Button 1 click --> start
I/flutter (11488): 屏幕 Size -> Size(392.7, 872.7)
I/flutter (11488): Current Button 1 click --> end
I/flutter (11488): Current Button 2 click --> start
I/flutter (11488): 按钮 Size -> Size(190.4, 48.0)
I/flutter (11488): 文字 Size -> Size(67.0, 23.0)
I/flutter (11488): 文字 Size -> Size(392.7, 872.7)
I/flutter (11488): Current Button 2 click --> end
```

2. devicePixelRatio

devicePixelRatio 为像素密度；与设备物理像素有关，与横竖屏等无关；

```
print('屏幕像素比 -> $
{MediaQuery.of(context).devicePixelRatio}');
```

```
I/flutter (11488): Current Button 4 click --> start
I/flutter (11488): 屏幕像素比 -> 2.75
I/flutter (11488): Current Button 4 click --> end
```

3. orientation

orientation 为横竖屏，**Orientation.landscape** 为横屏，

orientation 方向，landscape 横向，portrait 竖向的

Orientation.portrait 为竖屏；

```
print('横竖屏 -> ${MediaQuery.of(context).orientation}');
```

```
I/flutter (11488): Current Button 3 click --> start
I/flutter (11488): 横竖屏 -> Orientation.portrait
I/flutter (11488): Current Button 3 click --> end
I/flutter (11488): Current Button 3 click --> start
I/flutter (11488): 横竖屏 -> Orientation.landscape
I/flutter (11488): Current Button 3 click --> end
```

4. textScaleFactor

textScaleFactor 为每个逻辑像素的字体像素数，小菜理解为字体的像素比；注意，小菜设置了默认字体像素密度为标准的 1.2 倍之后调整设备系统字号，其 1.2 倍依旧是以标准字号为基础扩大 1.2 倍；

```
print('字体像素比 -> $
{MediaQuery.of(context).textScaleFactor}');

MediaQuery(data:
MediaQuery.of(context).copyWith(textScaleFactor: 1.2),
    child: Text('字体像素比 * 1.2', style: TextStyle(color:
Colors.white, fontSize: 16.0));
print('字体像素比 * 1.2 -> $
```

```
{MediaQuery.of(context).copyWith(textScaleFactor:
1.2).textScaleFactor}');
```

```
I/flutter (14264): Current Button 5 click --> start
I/flutter (14264): 字体像素比 -> 1.0
I/flutter (14264): Current Button 5 click --> end
I/flutter (14264): Current Button 6 click --> start
I/flutter (14264): 字体像素比 * 1.2 -> 1.2
I/flutter (14264): Current Button 6 click --> end
```

```
I/flutter (14801): Current Button 5 click --> start
I/flutter (14801): 字体像素比 -> 1.25
I/flutter (14801): Current Button 5 click --> end
I/flutter (14801): Current Button 6 click --> start
I/flutter (14801): 字体像素比 * 1.2 -> 1.2
I/flutter (14801): Current Button 6 click --> end
```

5. platformBrightness

platformBrightness 为当前设备的亮度模式；注意调整屏幕亮度并不会改变该模式，与当前系统支持的黑暗模式和明亮模式相关；

```
print('亮度模式 -> $
{MediaQuery.of(context).platformBrightness}');
```

```
I/flutter (18240): Current Button 7 click --> start
I/flutter (18240): 亮度模式 -> Brightness.light
I/flutter (18240): Current Button 7 click --> end
```

6. alwaysUse24HourFormat

alwaysUse24HourFormat 为当前设备是否为 **24** 小时制；

```
print('24 小时制 -> $
{MediaQuery.of(context).alwaysUse24HourFormat}');
```

```
I/flutter (18240): Current Button 8 click --> start
I/flutter (18240): 24 小时制 -> false
I/flutter (18240): Current Button 8 click --> end
```

7. accessibleNavigation

accessibleNavigation 为是否使用 **TalkBack** 或 **VoiceOver** 之类的辅助功能与应用程序进行交互，用以辅助视力障碍人群；

```
print('亮度模式 -> $
{MediaQuery.of(context).accessibleNavigation}');
```

```
I/flutter (18240): Current Button 9 click --> start
I/flutter (18240): 辅助视力障碍 -> false
I/flutter (18240): Current Button 9 click --> end
```

8. invertColors

invertColors 为是否使用颜色反转，主要用于 **iOS** 设备；

```
print('颜色反转 -> ${MediaQuery.of(context).invertColors}');
```

```
I/flutter (18240): Current Button 10 click --> start
I/flutter (18240): 颜色反转 -> false
I/flutter (18240): Current Button 10 click --> end
```

9. highContrast

highContrast 为用户是否要求前景与背景之间的对比度高，主要用于 **iOS** 设备；

```
print('前后背景高对比度 -> $
{MediaQuery.of(context).highContrast}');
```

```
I/flutter (18240): Current Button 11 click --> start
I/flutter (18240): 前后背景高对比度 -> false
I/flutter (18240): Current Button 11 click --> end
```

10. disableAnimations

disableAnimations 为平台是否要求禁用或减少动画；

```
print('是否减少动画 -> $
{MediaQuery.of(context).disableAnimations}');
```

```
I/flutter (18240): Current Button 12 click --> start
I/flutter (18240): 是否减少动画 -> false
I/flutter (18240): Current Button 12 click --> end
```

11. boldText

boldText 为平台是否要求使用粗体；

```
print('是否使用粗体 -> ${MediaQuery.of(context).boldText}');
```

```
I/flutter (19849): Current Button 13 click --> start
I/flutter (19849): 是否使用粗体 -> false
I/flutter (19849): Current Button 13 click --> end
```

12. padding

padding 为屏幕内边距，一般是刘海儿屏或异形屏中被系统遮挡部分边距；

```
print('内边距 -> ${MediaQuery.of(context).padding}');
```

```
I/flutter (19849): Current Button 14 click --> start
I/flutter (19849): 内边距 -> EdgeInsets(0.0, 34.5, 0.0, 0.0)
I/flutter (19849): Current Button 14 click --> end
```

13. viewInsets

viewInsets 为键盘弹出时等遮挡屏幕边距，其中 **viewInsets.bottom** 为键盘高度；

```
print('键盘遮挡内边距 -> $
{MediaQuery.of(context).viewInsets}');
```

```
I/flutter (19849): Current Button 15 click --> start
I/flutter (19849): 键盘遮挡内边距 -> EdgeInsets.zero
I/flutter (19849): Current Button 15 click --> end
I/flutter (19849): Current Button 15 click --> start
I/flutter (19849): 键盘遮挡内边距 -> EdgeInsets(0.0, 0.0, 0.0, 337.5)
I/flutter (19849): Current Button 15 click --> end
```

14. systemGestureInsets

systemGestureInsets 为手势边距，如 **Android Q** 之后添加的向左滑动关闭页面等；

```
print('系统手势边距 -> $
{MediaQuery.of(context).systemGestureInsets}');
```

```
I/flutter (19849): Current Button 16 click --> start
I/flutter (19849): 系统手势边距 -> EdgeInsets(0.0, 34.5, 0.0, 0.0)
I/flutter (19849): Current Button 16 click --> end
```

15. viewPadding

viewPadding 小菜理解为视图内边距，为屏幕被刘海儿屏或异形屏中被系统遮挡部分，从 **MediaQuery** 边界的边缘计算；此值是保持不变；例如，屏幕底部的软件键盘可能会覆盖并占用需要底部填充的相同区域，因此不会影响此值；

```
print('系统手势边距 -> $
{MediaQuery.of(context).systemGestureInsets}');
```

```
I/flutter (19849): Current Button 17 click --> start
I/flutter (19849): 视图内边距 -> EdgeInsets(0.0, 34.5, 0.0, 0.0)
I/flutter (19849): Current Button 17 click --> end
```

16. physicalDepth

physicalDepth 为设备物理层级，小菜暂时还未想到对应的应用场景；

```
print('设备物理层级 -> $
{MediaQuery.of(context).physicalDepth}');
```

```
I/flutter (19849): Current Button 18 click --> start
I/flutter (19849): 设备物理层级 -> 1.7976931348623157e+308
I/flutter (19849): Current Button 18 click --> end
```

Tips

小菜在尝试获取其他子 **Widget Size** 时，有两点需要注意，首先要设置一个全局的 **GlobalKey** 来获取当前位置，**key** 需要为唯一的；第二通过 **GlobalKey().currentContext** 获取 **BuildContext** 上下文环境，从而获取对应尺寸；

```
var _itemExpandedKey = GlobalKey();
var _itemTextKey = GlobalKey();

Expanded(
    key: _itemExpandedKey,
    child: FlatButton(
        onPressed: () => _itemClick(2),
        child: Center(child: Text('按钮 Size', key:
_itemTextKey, style: TextStyle(color: Colors.white,
fontSize: 16.0))),
        color: Colors.purpleAccent.withOpacity(0.4)))
```

MediaQuery 案例尝试