Scaling unit

مصطفی نصرتی پور

دانشگاه بهار

**واحدهای مقیاس در اندروید**:

در توسعه اپلیکیشن‌های اندرویدی، استفاده صحیح از واحدهای مقیاس بسیار مهم است تا اطمینان حاصل شود که رابط کاربری اپلیکیشن شما در دستگاه‌های مختلف با اندازه‌های صفحه نمایش متفاوت، به خوبی نمایش داده شود و تجربه کاربری یکسانی را برای همه کاربران فراهم کند.

**چرا واحدهای مقیاس مهم هستند؟**

* **تنوع دستگاه‌ها:** دستگاه‌های اندرویدی با اندازه‌های صفحه نمایش، رزولوشن و تراکم پیکسل‌های بسیار متنوعی وجود دارند.
* **انطباق با اندازه‌های مختلف:** با استفاده از واحدهای مقیاس، می‌توانیم اطمینان حاصل کنیم که عناصر رابط کاربری (مانند دکمه‌ها، متن‌ها و تصاویر) در همه دستگاه‌ها با نسبت‌های مناسب نمایش داده شوند.
* **بهبود تجربه کاربری:** استفاده صحیح از واحدهای مقیاس باعث می‌شود که اپلیکیشن ما حرفه‌ای‌تر به نظر برسد و کاربر بتواند به راحتی با آن تعامل داشته باشد.

**واحدهای مقیاس اصلی در اندروید**

**dp (density-independent pixel):** این واحد مقیاس، مستقل از تراکم پیکسل دستگاه است و برای اندازه گیری ابعاد UI استفاده می‌شود. یک dp در دستگاه‌هایی با تراکم پیکسل متفاوت، اندازه فیزیکی متفاوتی خواهد داشت.

**sp (scaled pixel)**: این واحد مقیاس شبیه به dp است، اما برای اندازه گیری اندازه فونت استفاده می‌شود. اندازه فونت sp با توجه به تنظیمات اندازه فونت دستگاه تغییر می‌کند.

**px (pixel)**: این واحد مقیاس، مستقیماً به یک پیکسل روی صفحه نمایش اشاره دارد. از این واحد باید با احتیاط استفاده شود، زیرا اندازه یک پیکسل در دستگاه‌های مختلف متفاوت است.

**چه زمانی از چه واحدی استفاده کنیم؟**

* **:dp** برای اندازه گیری ابعاد UI مانند عرض و ارتفاع View ها، Padding ها و Margin ها.
* **:sp** برای اندازه گیری اندازه فونت‌ها.
* **:px** در موارد بسیار خاص و زمانی که می‌خواهید اندازه دقیق یک عنصر را کنترل کنید.

**مثال عملی**

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textSize="16sp"

android:padding="16dp"

android:text="Hello, World!" />

در کد بالا:

**android:textSize="16sp":** ندازه فونت متن 16 اسپ خواهد بود و با توجه به تنظیمات کاربر تغییر می‌کند.

**android:padding="16dp":** حاشیه داخلی TextView 16 dp خواهد بود و در دستگاه‌های مختلف متناسب با تراکم پیکسل تنظیم می‌شود.

**از px اجتناب کنید:** تا حد امکان از استفاده از px اجتناب کنید، زیرا این واحد مقیاس باعث می‌شود که اپلیکیشن شما در دستگاه‌های مختلف با مشکل نمایش مواجه شود.

**از ابزارهای طراحی استفاده کنید:** از ابزارهای طراحی مانند Android Studio استفاده کنید که به شما کمک می‌کنند تا ابعاد UI را به صورت بصری طراحی کنید و از واحدهای مقیاس صحیح استفاده کنید.

**تست کنید:** اپلیکیشن خود را روی دستگاه‌های مختلف با اندازه‌های صفحه نمایش متفاوت تست کنید تا اطمینان حاصل کنید که رابط کاربری به درستی نمایش داده می‌شود.

تغییر اندازه فونت یک TextView در زمان اجرا:

TextView textView = findViewById(R.id.myTextView);

float scaledDensity = getResources().getDisplayMetrics().scaledDensity;

int newTextSize = (int) (16 \* scaledDensity); // 16sp

textView.setTextSize(TypedValue.COMPLEX\_UNIT\_PX, newTextSize);

محاسبه اندازه یک View بر اساس عرض صفحه:

DisplayMetrics displayMetrics = new DisplayMetrics(); getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);

int screenWidth = displayMetrics.widthPixels;

int viewWidth = (int) (screenWidth \* 0.8); // 80%

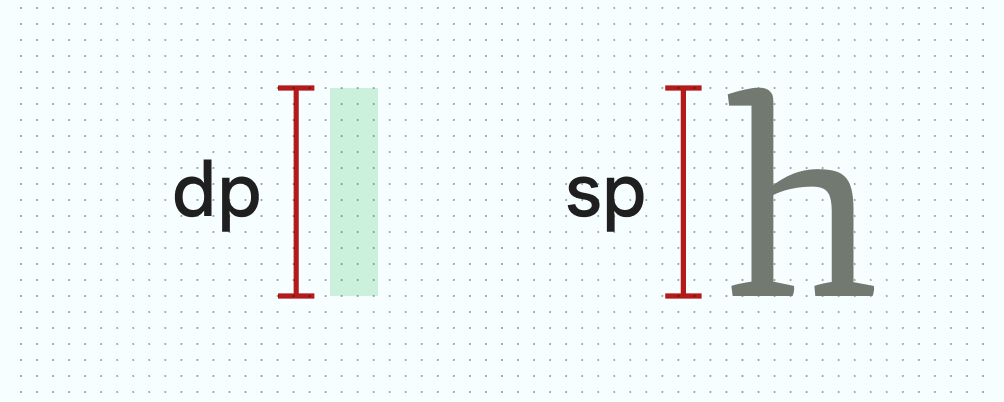
view.setLayoutParams(new LinearLayout.LayoutParams(viewWidth, ViewGroup.LayoutParams.WRAP\_CONTENT));

**Grids and units :**

Density independent pixels (dp) and scalable pixels (sp) are essential for building layouts and presenting fonts that respond uniformly to the wide range of screen densities, size classes, form factors, and aspect ratios that make up Android devices.

## Takeaways

* If using a baseline grid, stick to measurements of 4 and 8.
* Notate specs in dp and sp, instead of pixels.
* Export bitmap/raster graphics for all buckets.
* Design with a responsive mindset with different size classes, resolutions, and aspect ratios in mind.
* **Density-independent pixels (dp)**: density-independent pixels are flexible units that scale to have uniform dimensions on any screen. They are based on the physical density of the screen. These units are relative to a 160 dpi (dots per inch) screen, on which 1 dp is roughly equal to 1 px.
* **Scalable pixels (sp)**: Scalable pixels serve the same function as dp, but for fonts. The default value of an sp is the same as the default value for a dp. The Android system calculates the actual font size to use based on the device and the user's preference set in the Settings app of their Android device.

****

The primary difference between these units of measurement is that scalable pixels preserve a user's font settings. Users who have larger text settings for accessibility see font sizes match their text size preferences. See how to [change font size](https://developer.android.com/jetpack/compose/text#changing-size) in Compose.

Android uses these units to help scale and translate across the range of devices and resolutions.

## Density buckets

High-density screens have more pixels per inch than low-density ones. As a result, UI elements of the same pixel dimensions appear larger on low-density screens, and smaller on high-density screens. This is why you should not declare measurements in pixels.

Android groups ranges of screen densities into "buckets" and uses them to deliver the optimal set of assets to your device. The most commonly used density buckets are mdpi, hdpi, xhdpi, xxhdpi, and xxxhdpi (nodpi and anydpi refer to a bucket that does not scale per device resolution, typically used for vector drawables) each correspond to a resource file of your app.

****

To calculate dp:

dp = (width in pixels \* 160) / screen density

## **Grids**

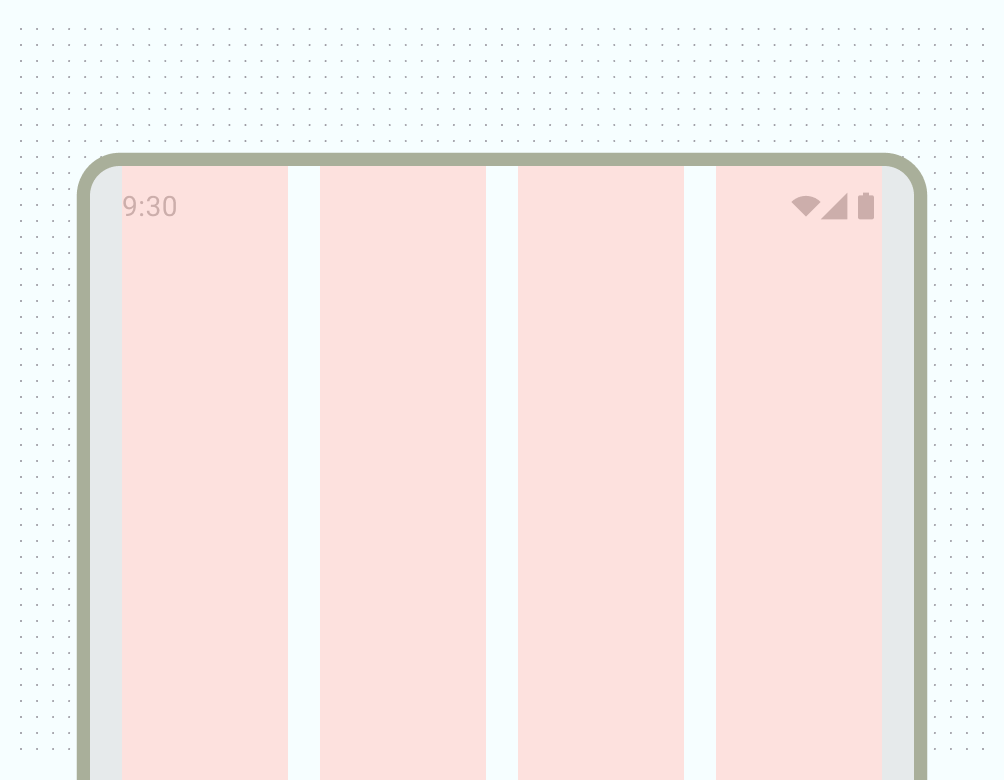
### Baseline grid

Building with an underlying grid helps create consistent spacing and alignment across your UI. Android UI utilizes an 8 dp grid for layout, components, and spacing.

Smaller elements such as icons, type, and some elements within components are best aligned to a 4 dp grid

### Column grid

Columns build a grid structure to provide vertical definition to a layout by dividing content within the body area. Content is placed in the areas of the screen that contain columns. Align with an underlying grid to align content, but should keep flexible sizing. Learn the basics on how to set up a column grid and apply content in

****

***Figure 4:****Four-column grid*

## **Size classes**

Window size classes are a set of opinionated viewport breakpoints that help you design, develop, and test responsive and adaptive application layouts. Android breaks window size classes into 3: Compact, Medium, and Expanded. Read more on [Window size classes](https://developer.android.com/develop/ui/compose/layouts/adaptive/window-size-classes).

### Aspect ratios

An aspect ratio is the proportion of an element's width to its height. Aspect ratios are written as width:height.

To maintain consistency in your layout, use a consistent aspect ratio on elements like images, surfaces, and screen size.

The following aspect ratios are recommended for use across your UI:

* 16:9
* 3:2
* 4:3
* 1:1
* 3:4
* 2:3

**منابع**

[**https://developer.android.com/design/ui/mobile/guides/layout-and-content/grids-and-units**](https://developer.android.com/design/ui/mobile/guides/layout-and-content/grids-and-units)