

What is the difference between px, dip, dp, and sp?

Asked 14 years, 11 months ago Modified 8 months ago

Viewed 1.5m times  Part of [Mobile Development](#) Collective



What is the difference between the units of measure px, dip, dp, and sp?

6411



MD

android

android-layout

user-interface

dimension



units-of-measurement



Share

Improve this question

Follow

edited Jul 11, 2022 at 5:53



[Mateen Ulhaq](#)

27.1k ● 21 ● 117 ● 152

asked Jan 8, 2010 at 3:23




[capecrawler](#)

68.1k ● 7 ● 33 ● 34

83 this is use full difference between px, dip, dp and sp in android [[developer.android.com/guide/topics/resources/...](https://developer.android.com/guide/topics/resources/)]
— [NagarjunaReddy](#) May 25, 2012 at 9:34


- 30 [This nifty converter demonstrates it best](#), in my opinion. It's also extremely useful for exporting sprites from Photoshop or designing your layout for a physical dimension.
– [Paul Lammertsma](#) Jul 1, 2014 at 9:28

[How to programmatically convert between px, dp, and sp](#) – [Suragch](#)  Jul 15, 2017 at 10:40

from android developers site

developer.android.com/guide/practices/screens_support.html

– [Gangadhar Lahane](#) Sep 13, 2017 at 7:41

- 4 Material design with pixel-density material.io/design/layout/...
– [Samiran](#) Oct 4, 2018 at 5:19 

34 Answers

Sorted by:

Highest score (default)



1

2

Next



From the [Android Developer Documentation](#):

6233



1.

px

Pixels - corresponds to actual pixels on the screen.

2.

in

Inches - based on the physical size of the screen.

1 Inch OR 2.54 centimeters

3.

mm

> **Millimeters** - based on the physical size

of the screen.

4.

pt

> **Points** - $\frac{1}{72}$ of an inch based on the physical size of the screen.

5.

dp or dip

> **Density-independent Pixels** - an abstract unit that is based on the physical density of the screen. These units are relative to a **160** dpi screen, so one dp is one pixel on a **160** dpi screen. The ratio of dp-to-pixel will change with the screen density, but not necessarily in direct proportion. Note: The compiler accepts both "dip" and "dp", though "dp" is more consistent with "sp".

6.

sp

> [Scaleable Pixels](#) OR [scale-independent pixels](#) - this is like the dp unit, but it is also scaled by the user's font size preference. It is recommended you use this unit when specifying font sizes, so they will be adjusted for both the screen density and the user's preference. Note, the Android documentation is inconsistent on what **sp** actually stands for, one [doc](#) says "scale-independent pixels", the [other](#) says "scaleable pixels".

From [Understanding Density Independence In Android](#):

Density Bucket	Screen Density	Physical Size	Pixel Siz
ldpi	120 dpi	0.5×0.5 in	$0.5 \text{ in} \times 120 \text{ dpi} = 60 \times 60 \text{ px}$
mdpi	160 dpi	0.5×0.5 in	$0.5 \text{ in} \times 160 \text{ dpi} = 80 \times 80 \text{ px}$
hdpi	240 dpi	0.5×0.5 in	$0.5 \text{ in} \times 240 \text{ dpi} = 120 \times 120 \text{ px}$
xhdpi	320 dpi	0.5×0.5 in	$0.5 \text{ in} \times 320 \text{ dpi} = 160 \times 160 \text{ px}$
xxhdpi	480 dpi	0.5×0.5 in	$0.5 \text{ in} \times 480 \text{ dpi} = 240 \times 240 \text{ px}$

Density Bucket	Screen Density	Physical Size	Pixel Size
xxxhdpi	640 dpi	0.5 x 0.5 in	0.5 in * 640 dpi = 320x320 px

Unit	Description	Units Per Physical Inch	Density Independent
px	Pixels	Varies	No
in	Inches	1	Yes
mm	Millimeters	25.4	Yes
pt	Points	72	Yes
dp	Density Independent Pixels	~160	Yes
sp	Scale Independent Pixels	~160	Yes

More info can be also be found in the [Google Design Documentation](#).

Share Improve this answer edited Apr 21 at 14:41

Follow

Why does dp has "no" in the table in the last column? Isn't it the same as mm, pt and in, but dp has constant size 1/160 inches. I mean you can change a device's screen and its dpi but the physical size of dp is going to be the same, isn't it ?
– [VanechikSpace](#) Aug 15, 2022 at 8:02 ✎



755

Pretty much everything about this and how to achieve the best support for multiple screens of different sizes and densities is very well documented here:



- [Supporting Multiple Screens](#)



Screen size

Actual physical size, measured as the screen's diagonal. For simplicity, Android groups all actual screen sizes into four generalized sizes: small, normal, large, and extra-large.

Screen density

The number of pixels within a physical area of the screen; usually referred to as dpi (dots per inch). For example, a "low" density screen has fewer pixels within a given physical area, compared to a "normal" or "high" density screen. For simplicity, Android groups all actual screen densities into six generalized densities: low, medium, high, extra-high, extra-extra-high, and extra-extra-extra-high.

Orientation

The orientation of the screen from the user's point of view. This is either landscape or portrait, meaning that the screen's aspect ratio is either wide or tall, respectively. Be aware that not only do different devices operate in different orientations by default, but the orientation can change at runtime when the user rotates the device.

Resolution

The total number of physical pixels on a screen. When adding support for multiple screens, applications do not work directly with resolution; applications should be concerned only with screen size and density, as specified by the generalized size and density groups.

Density-independent pixel (dp)

A virtual pixel unit that you should use when defining UI layout, to express layout dimensions or position in a density-independent way. The density-independent pixel is equivalent to one physical pixel on a 160 dpi screen, which is the baseline density assumed by the system for a "medium" density screen. At runtime, the system transparently handles any scaling of the dp units, as necessary, based on the actual density of the screen in use. The conversion of dp units to

screen pixels is simple: $px = dp * (dpi / 160)$.

For example, on a 240 dpi screen, 1 dp equals 1.5 physical pixels. You should always use dp units when defining your application's UI, to ensure proper display of your UI on screens with different densities.

If you are at all serious about developing an Android app for more than one type of device, you should have read the screens support development document at least once. In addition to that, it is always a good thing to know the actual number of active devices that have a particular screen configuration.

- [Screen Sizes and Densities](#)

Share Improve this answer

Follow

edited Jul 31, 2021 at 16:08



auspicious99

4,311 ● 1 ● 47 ● 66

answered Feb 25, 2011 at 12:56



Bruiser

11.9k ● 5 ● 35 ● 45



I will elaborate more on how exactly does dp convert to px:

394



- If running on an mdpi device, a 150×150 px image will take up 150×150 dp of screen space.
- If running on an hdpi device, a 150×150 px image will take up 100×100 dp of screen space.



- If running on an xhdpi device, a `150 x 150 px` image will take up `75 x 75 dp` of screen space.

The other way around: say, you want to add an image to your application and you need it to fill a `100 x 100 dp` control. You'll need to create different size images for supported screen sizes:

- `100 x 100 px` image for mdpi
- `150 x 150 px` image for hdpi
- `200 x 200 px` image for xhdpi

Share Improve this answer

edited Oct 12, 2023 at 4:17

Follow



[ankushlokhande](#)

1,738 ● 11 ● 27

answered Aug 2, 2012 at 14:15

[devmiles.com](#)

>devmiles 10k ● 5 ● 34 ● 48

How the fontsize is represened? in dp or sp? I have an app the text shows bigger in some modal. So it comes to the next line or fully occupying the spaces. How does this fontsize will be set to show as good in android – [sejn](#) Jan 25, 2021 at 10:55



Moreover you should have a clear understanding of the following concepts:

254

Screen size:



Actual physical size, measured as the screen's diagonal. For simplicity, Android groups all actual screen sizes into four generalized sizes: small, normal, large, and extra-large.

Screen density:

The number of pixels within a physical area of the screen; usually referred to as dpi (dots per inch). For example, a "low" density screen has fewer pixels within a given physical area, compared to a "normal" or "high" density screen. For simplicity, Android groups all actual screen densities into four generalized densities: low, medium, high, and extra high.

Orientation:

The orientation of the screen from the user's point of view. This is either landscape or portrait, meaning that the screen's aspect ratio is either wide or tall, respectively. Be aware that not only do different devices operate in different orientations by default, but the orientation can change at runtime when the user rotates the device.

Resolution:

The total number of physical pixels on a screen. When adding support for multiple screens, applications do not work directly with resolution; applications should be concerned only with screen size and density, as specified by the generalized size and density groups.

Density-independent pixel (dp):

A virtual pixel unit that you should use when defining UI layout, to express layout dimensions or position in a density-independent way. The density-independent pixel is equivalent to one physical pixel on a 160 dpi screen, which is the baseline density assumed by the system for a "medium" density screen. At runtime, the system transparently handles any scaling of the dp units, as necessary, based on the actual density of the screen in use. The conversion of dp units to screen pixels is simple: $px = dp * (dpi / 160)$. For example, on a 240 dpi screen, 1 dp equals 1.5 physical pixels. You should always use dp units when defining your application's UI, to ensure proper display of your UI on screens with different densities.

Reference: [Android developers site](#)

Follow



Rahul

3,349 ● 2 ● 34 ● 44

answered Jul 26, 2013 at 21:19



Sazzad Hissain Khan

40.1k ● 41 ● 208 ● 298



224

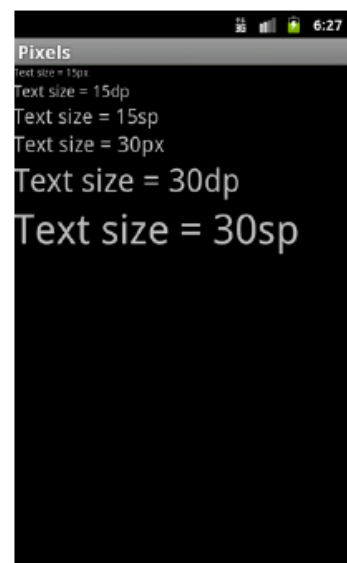
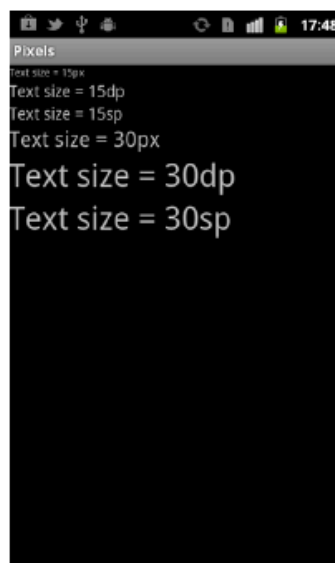
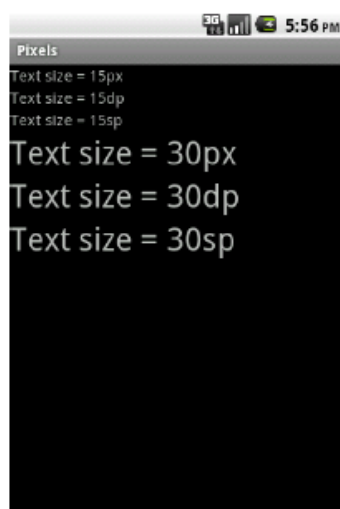
`dp` is `dip`. Use it for everything (margin, padding, etc.).

Use `sp` for {text-size} only.



To get the same size on different screen densities, Android translates these units into pixels at runtime, so there is no tricky math for you to do.

See the difference between `px`, `dp` and `sp` on different screen sizes.



Source: [Android Programming: The Big Nerd Ranch Guide](#)

Share Improve this answer

edited Aug 11, 2015 at 20:53

Follow

answered Oct 17, 2014 at 12:47



Mina Gabriel

25k ● 26 ● 99 ● 128



Definitions

174

px or dot is a *pixel* on the physical screen.



dpi are pixels per inch on the physical screen and represent the density of the display.



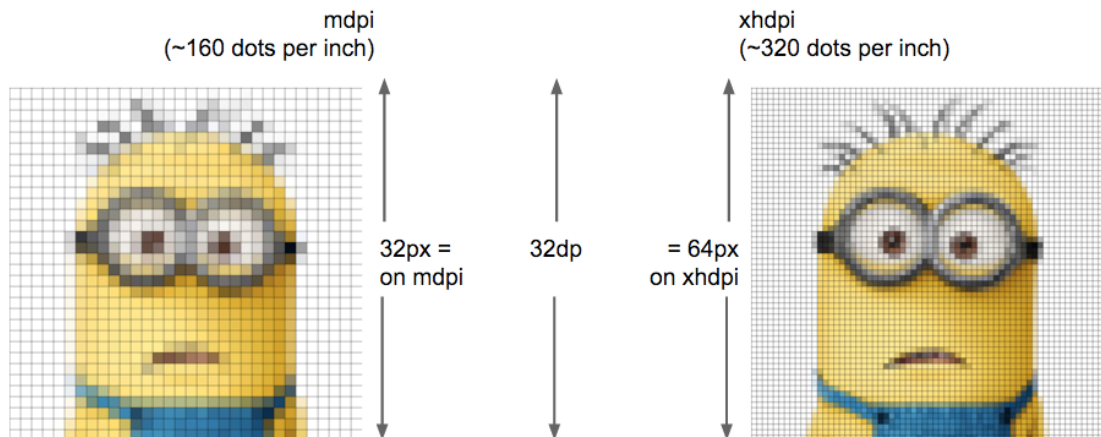
Android gives [alias names to several densities](#)

- ldpi (low) ~120dpi
- mdpi (medium) ~160dpi
- hdpi (high) ~240dpi
 - **most devices in 2015 are here**
- xhdpi (extra-high) ~320dpi
 - Apple iPhone 4/5/6, Nexus 4
- xxhdpi (extra-extra-high) ~480dpi
 - Nexus 5

- xxxhdpi (extra-extra-extra-high) ~640dpi

dip or **dp** are *density-independant pixels*, i.e. they correspond to more or less pixels depending on the physical density.

- 1dp = 1px on mdpi



sp or **sip** is a *scale-independant pixel*. They are scaled when the *Large Text* option is turned on in [Settings](#) >

[Accessibility](#)

- 1sp = 1dp
- 1sp = 1.2dp with accessibility Large Text

What to use?

Use **sp** for Text size.

Use **dp** for everything else.

Follow

edited Mar 2, 2016 at 9:39

answered Jan 10, 2016 at 19:13



rds

26.9k ● 20 ● 110 ● 135



157



I have calculated the formula below to make the conversions `dpi` to `dp` and `sp`

Size (px)	dpi	$dp = [px / (dpi)] * 160$	$sp = [px / (dpi)] * 160$
1 10px	Ldpi (120)	13.33	13.33
2 10px	Mdpi (160)	10	10
3 10px	Hdpi (240)	~7	~7
4 10px	Xhdpi (320)	5	5

Share Improve this answer

edited Apr 6, 2017 at 10:24

Follow

community wiki

4 revs, 4 users 50%

chaitanya

11 It should be ppi instead of dpi – Prateek Jan 29, 2014 at 17:02



[Source 1](#)

143

[Source 2](#)



[Source 3](#): (data from source 3 is given below)



These are dimension values defined in XML. A dimension is specified with a number followed by a unit of measure. For example: 10px, 2in, 5sp. The following units of measure are supported by Android:

dp

Density-independent Pixels - An abstract unit that is based on the physical density of the screen. These units are relative to a 160 dpi (dots per inch) screen, on which 1dp is roughly equal to 1px. When running on a higher density screen, the number of pixels used to draw 1dp is scaled up by a factor appropriate for the screen's dpi. Likewise, when on a lower density screen, the number of pixels used for 1dp is scaled down. The ratio of dp-to-pixel will change with the screen density, but not necessarily in direct proportion. Using dp units (instead of px units) is a simple solution to making the view dimensions in your layout resize properly for different screen densities. In other words, it provides consistency for the real-world sizes of your UI elements across different devices.

sp

Scale-independent Pixels - This is like the dp unit, but it is also scaled by the user's font size preference. It is recommended that you use this unit when specifying font sizes, so they will be adjusted for both the screen density and the user's preference.

pt

Points - 1/72 of an inch based on the physical size of the screen.

px

Pixels - Corresponds to actual pixels on the screen. This unit of measure is not recommended because the actual representation can vary across devices; each device may have a different number of pixels per inch and may have more or fewer total pixels available on the screen.

mm

Millimeters - Based on the physical size of the screen.

in

Inches - Based on the physical size of the screen.

Note: A dimension is a simple resource that is referenced using the value provided in the name attribute (not the name of the XML file). As such, you can combine dimension resources with other simple resources in one XML file, under one element.

Share Improve this answer

Follow

edited Sep 9, 2021 at 17:03



Rahul

3,349 ● 2 ● 34 ● 44

answered Dec 27, 2013 at 9:21



sms247

4,514 ● 5 ● 36 ● 46

11 Also: what is the added value of this answer? There doesn't seem to be anything that hasn't already been mentioned in other answers. – [laalto](#) Dec 27, 2013 at 9:28



132

Basically the only time where px applies is one px, and that's if you want exactly one pixel on the screen like in the case of a divider:



On >160 dpi, you may get 2-3 pixels,

On >120 dpi, it rounds to 0.



Share Improve this answer

Follow

edited Mar 31, 2018 at 10:05



RK RK

1,294 ● 5 ● 14 ● 18

answered Sep 20, 2012 at 21:34



Joe Plante

6,368 ● 2 ● 31 ● 23



109



Where to use what & relationship between px & dp?

Density-independent pixel (dp)

A virtual pixel unit that you should use when defining UI layout, to express layout dimensions or position in a density-independent way. As described above, the density-independent pixel is equivalent to one physical pixel on a 160 dpi screen, which is the baseline density assumed by the system for a "medium" density screen. At runtime, the system transparently handles any scaling of the dp units, as necessary, based on the actual density of the screen in use. The conversion of dp units to screen pixels is simple:

$$\text{px} = \text{dp} * (\text{dpi} / 160).$$

For example, on a 240 dpi screen, 1 dp equals 1.5 physical pixels. You should always use dp units when defining your application's UI, to ensure proper display of your UI on screens with different densities.

Understanding pixel to dp and vice versa is very essential (especially for giving exact dp values to creative team)

$dp = px * 160 / dpi$

MDPI = 160 dpi || Therefore, on MDPI 1 px = 1 dp
For example, if you want to convert 20 pixel to dp, us
 $dp = 20 * 160 / 160 = 20$.
So, 20 pixel = 20 dp.

HDPI = 240 dpi - So, on HDPI 1.5 px = 1 dp
XHDPI = 320 dpi - So, on XHDPI 2 px = 1 dp
XXHDPI = 480 dpi - So, on XXHDPI 3 px = 1 dp

For example, let us consider Nexus 4.
If 24 pixels to be converted to dp and if it is a Nexus
can
convert it to dp easily by the following calculation :
 $dp = 24 * 160 / 320 = 12$ dp
Screen dimension:
768 x 1280 pixel resolution (320 ppi or 320dpi)
Optional (screen size):
4.7" diagonal

- Try to get all pixel values in even numbers from the creative team. Otherwise precision lose will happen while multiplying with 0.5.

px

It is explained above. Try to avoid in layout files. But there are some cases, where px is required. for example, ListView divider line. px is better here for giving a one-pixel line as a divider for all across screen resolutions.

sp

Use sp for font sizes. Then only the font inside the application will change while device fonts size changes

(that is, Display -> Fonts on Device). If you want to keep a static sized font inside the app, you can give the font dimension in dp. In such a case, it will never change. Developers may get such a requirement for some specific screens, for that, developers can use dp instead of sp. In all other cases, sp is recommended.

Share Improve this answer

edited Jun 20, 2020 at 9:12

Follow



Community Bot

1 • 1

answered Jul 9, 2014 at 5:46



Arunjyothis

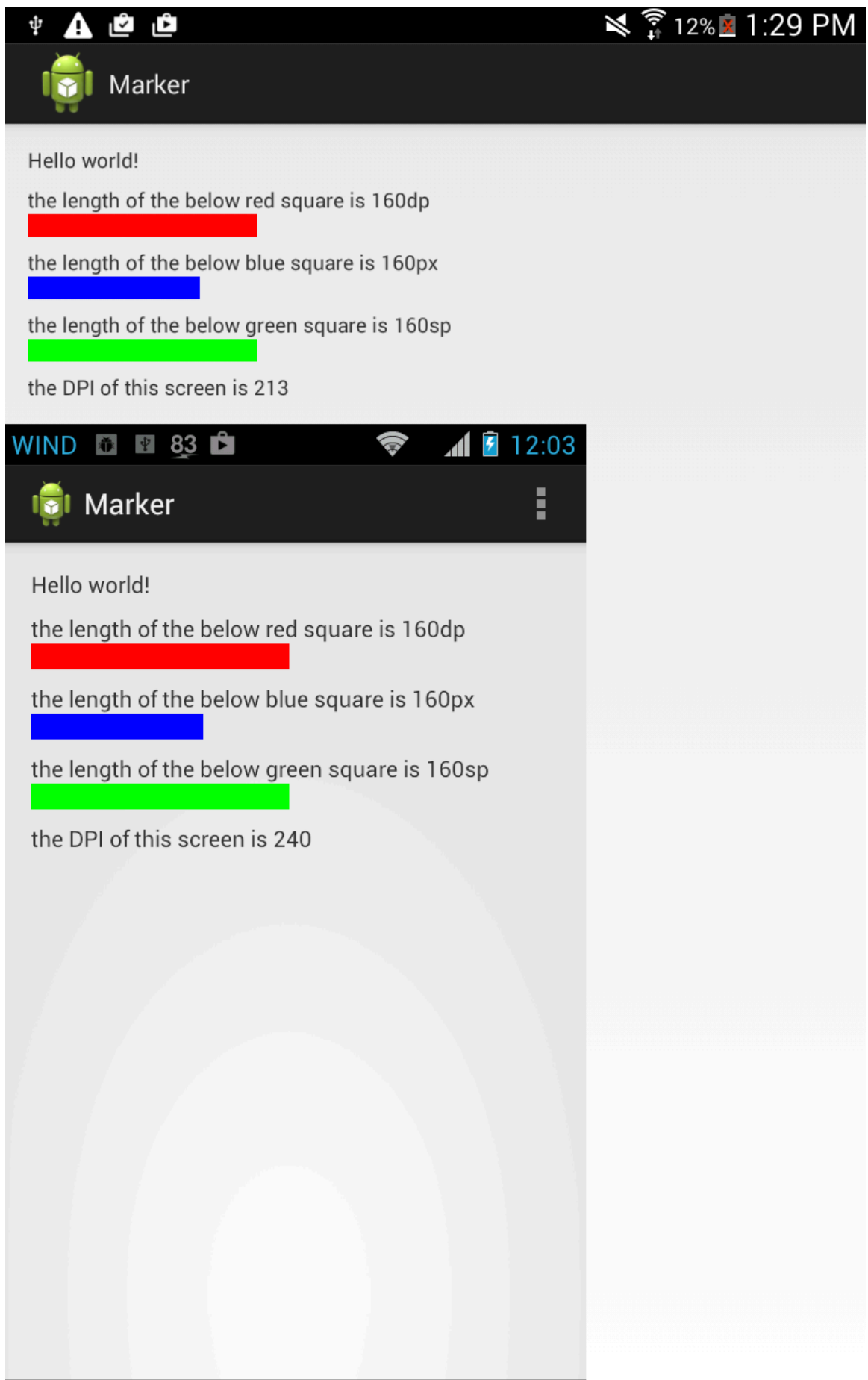
1,446 • 1 • 10 • 11



101



You can see the difference between px and dp from the below picture, and you can also find that the px and dp could not guarantee the same physical sizes on the different screens.



Share Improve this answer

Follow

answered Feb 13, 2015 at 18:50



Zephyr

6,331 ● 2 ● 35 ● 33

-
- 3 @EnesBattal, I think because the dp isn't an acute physical size, it is a approximate value. Quoting from CapTech : "dp - This is a density independent unit, however the physical size of a single "dp" is only approximately the same on every screen density. There are approximately 160 "dp" in an inch. A scaling factor, depending on the density bucket of the device, is applied to convert "dp" to the number of pixels at 160 dpi. The number of pixels a single "dp" translates to varies depending on the pixel on screen density and the density bucket the device falls into." – [Zephyr](#) Sep 17, 2015 at 14:12
-
- 3 @RuchirBaronia, I think the DP or DIP is still there inside apk, because the apk doesn't know which kind of screen density it will run with yet, so the device independence should be still kept. – [Zephyr](#) Nov 25, 2015 at 18:18
-



88



Anything related with the size of text and appearance must use `sp` or `pt` . Whereas, anything related to the size of the controls, the layouts, etc. must be used with `dp` .

You can use both `dp` and `dip` at its places.



Share Improve this answer

[edited Jul 1, 2015 at 14:30](#)

Follow

answered Aug 9, 2013 at 13:38



[Atish Agrawal](#)

2,877 ● 1 ● 25 ● 39



82



I would only use dp.

There is a lot of talk about using "sp" for font sizes, and while I appreciate the point, I don't think that it is the right thing to do from a design point of view. You can end up breaking your design if the user has some wonky font size selection, and the user will end up blaming *the app*, and not their own life choices.

Also, if you take an sp-font app on a 160 dpi tablet, you will find that *everything* scales up... but your font, which is going to look tiny in comparison. It isn't a good look.

While the idea of "sp" fonts has a good heart, it is a poor idea. Stick with dp for everything.

Share Improve this answer

Follow

edited Jun 6, 2015 at 14:42



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Nov 22, 2014 at 16:56



bharal

16.1k ● 39 ● 122 ● 201

-
- 1 You know that the font scale factor applied to `sp` is a factor, right? Anything that affects `dp` will also affect `sp`. That said, it's still preferable to specify font sizes using `dp` instead of `sp` if your layout is very tight and the larger sizes won't fit - better to have text smaller than what the user wants than a completely messed up layout. But in the first instance you should always strive to respect the user's font size preference - even the biggest setting is not THAT big. – [Karu](#) Feb 26, 2017 at 22:06
-



sp = scale independent pixel

77

dp = dip = density independent pixels



dpi = dots per inch



We should avoid to use **sp**.



We should use **dp** to support multiple screens.

Android supports different screen resolutions

- ldpi (low) ~120 dpi
- mdpi (medium) ~160 dpi
- hdpi (high) ~240 dpi
- xhdpi (extra-high) ~320 dpi
- xxhdpi (extra-extra-high) ~480 dpi
- xxxhdpi (extra-extra-extra-high) ~640 dpi

An 120 dp ldpi device has 120 pixels in 1 inch size.

The same for other densities...

We as software engineers should use this conversion formula:

$$\text{pixel} = \text{dp} * (\text{density} / 160)$$

So 240 dpi device's 1 dp will have = $1 * (240/160) = 3/2 = 1.5$ pixels.

And 480 dpi device's 1 dp will have = $1 * (480/160) = 3$ pixels.

Using this 1.5 and 3 pixels knowledge, a software engineer can design layouts for different densities.

To check screen parameters of any device:

```
DisplayMetrics metrics = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(metrics);

Toast.makeText(
    this,
    "4:" + metrics.heightPixels + "," + metrics.densityDpi,
    Toast.LENGTH_LONG).show();
```

Share Improve this answer

Follow

edited Jun 20, 2020 at 9:12



Community Bot

1 • 1

answered Mar 11, 2015 at 6:33



Kushal

8,488 • 10 • 64 • 83

Very good tutorial for understanding is :

<http://vinsol.com/blog/2014/11/20/tips-for-designers-from-a-developer> – Kushal Mar 16, 2015 at 10:06

- 6 "We should avoid to use sp" Why is that? You should use sp when dealing with font sizes because it takes into account the user's preferred text size,

developer.android.com/training/multiscreen/screendensities.html – ci_ Mar 20, 2015 at 10:05

I have answered relative to layout perspective.. Please read on link you provided "so you should use this measurement unit when defining text size (but never for layout sizes)."

– Kushal Mar 20, 2015 at 10:22

- 1 The question was "Difference between px, dp, dip and sp in Android?" your answer said "We should avoid to use sp". There was no mention of "layout perspective" anywhere.

– ci_ Mar 20, 2015 at 10:28

Yes.. dp and dip are same... used interchangeably... Is my answer solve your query? – Kushal Nov 24, 2015 at 6:36



69



Difference between `dp` and `sp` units mentioned as "*user's font size preference*" by the answers copied from official documentation can be seen at run time by changing `Settings->Accessibility->Large Text` option.

`Large Text` option forces text to become `1.3` times bigger.

```
private static final float LARGE_FONT_SCALE = 1.3f;
```

This might be well of course vendor dependent since it lies in [packages/apps/Settings](#).

Share Improve this answer

Follow

answered Oct 14, 2013 at 8:44



[auselen](#)

28.1k ● 8 ● 76 ● 117



61



dpi -

- Dots per inches
- Measuring the pixel density of the screen.

px - pixel

- For mapping screen pixels

pt - points

- About 1/72 of an inch, with respect to physical screen size.

in - inch - with respect to physical screen size(1 inch = 2.54 cm).

mm- millimeter - with respect to physical screen size.

sp - scale-independent pixel.

- Based on user`s font size preference.
- Font should be in 'sp'.

dip -

- $dip == dp$
- Density independent pixel.
- It varies based on Screen Density.
- In 160 dpi screen, 1 dp = 1 pixel.
- Use dp except the text font size.

In standard, dp and sp are used. sp for font size and dp for everything else.

Formula for conversion of units:

```
px = dp * ( dpi / 160 );
```

Density Bucket -> Screen Display => Physical Size

ldpi -> 120 dpi => 0.5 x 0.5 in
60x60 px

mdpi -> 160 dpi => 0.5 x 0.5 in
80x80 px

hdpi -> 240 dpi => 0.5 x 0.5 in
120x120 px

xhdpi -> 320 dpi => 0.5 x 0.5 in
160x160 px

xxhdpi -> 480 dpi => 0.5 x 0.5 in
240x240 px

xxxhdpi -> 640 dpi => 0.5 x 0.5 in
320x320 px

Share Improve this answer

edited Nov 24, 2015 at 8:32

Follow

answered Jan 16, 2015 at 7:44



Arunendra

2,164 ● 26 ● 22

As per the documentation The compiler accepts both "dip" and "dp", though "dp" is more consistent with "sp".

– [Arunendra](#) Nov 24, 2015 at 8:15



57



Please read the answer from the community wiki. Below mentioned is some information to be considered in addition to the above answers. Most Android developers miss this while developing apps, so I am adding these points.



sp = scale independent pixel



dp = density independent pixels

dpi = density pixels

I have gone through the above answers...not finding them exactly correct. sp for text size, dp for layout bounds - standard. But **sp for text size will break the layout if used carelessly** in most of the devices.

sp take the text size of the device, whereas dp take that of device density standard(never change in a device)
Say 100sp text can occupy 80% of the screen or 100% of the screen depending on the font size set in the device

← Display

Brightness level

Wallpaper

Font size

Small



Normal



Large



Huge



Cancel

You can use sp for layout bounds also, it will work :) **No standard app use sp for whole text**

Use sp and dp for text size considering UX.

- Don't use sp for text in the toolbar(can use android dimens available for different screen sizes with dp)
- Don't use sp for text in small bounded buttons, very smaller text, etc

Some people use huge FONT size in their phone for more readability, giving them small hardcoded sized text will be a UX issue. Put sp for text where necessary, but make sure it won't break the layout when the user changes his settings.

Similarly, if you have a single app supporting all dimensions, adding xxxhdpi assets increases the app size a lot. But now xxxhdpi phones are common so we have to include xxxhdpi assets at least for icons in the sidebar, toolbar, and bottom bar. It's better to move to vector images to have a uniform and better quality images for all screen sizes.

Also, note that people use custom fonts on their phones. So lack of a font can cause problems regarding spacing and all. Say text size 12sp for a custom font may take some pixels extra than the default font.

Refer to google developer site for screen densities and base density details for android.

https://developer.android.com/training/multiscreen/screen_densities

Share Improve this answer

edited Sep 9, 2021 at 17:01

Follow



Rahul

3,349 ● 2 ● 34 ● 44

answered May 26, 2016 at 5:32



shijin

3,046 ● 1 ● 25 ● 31



54



Screen Size in `Android` is grouped into categories `small`, `medium`, `large`, `extra large`, `double-extra` and `triple-extra`. Screen density is the number of pixels within an area (like an inch) of the screen. Generally, it is measured in dots-per-inch (dpi). Screen density is grouped as low, medium, high, and extra high. Resolution is the total number of pixels on the screen.

- **dp:** Density Independent Pixel, varies based on screen density. In 160 dpi screen, 1 dp = 1 pixel. Except for font size, use dp always.
- **dip:** dip == dp. In earlier Android versions dip was used and later changed to dp.
- **sp:** Scale Independent Pixel, scaled based on user's font size preference. Fonts should use sp.
- **px:** our usual standard pixel which maps to the screen pixel.
- **in:** inches, concerning the physical screen size.

- **mm:** millimeters, concerning the physical screen size.
- **pt:** 1/72 of an inch, concerning the physical screen size.

Formula for Conversion between Units

$$\text{px} = \text{dp} * (\text{dpi} / 160)$$

dp to px in device

The following example may help understand better. The scaling occurs based on bucket sizes of 120(ldpi), 160(mdpi), 240(hdpi), 320(xhdpi), 480(xxhdpi), and 640(xxxhdpi). The Google suggested ratio for designing is 3:4:6:8:12 for ldpi:mdpi:hdpi:xhdpi:xxhdpi

A 150px X 150px image will occupy,

- 150 dp X 150 dp screen space in mdpi
- 100 dp X 100 dp screen space in hdpi
- 75 dp X 75 dp screen space in xhdpi

You may use the following DPI calculator to fix your image sizes and other dimensions when you wish to have a uniform UI design on all Android devices.

DPI Calculator in Java

```
/*  
Program output  
LDPI: 165.0 X 60.0  
MDPI: 220.0 X 80.0  
HDPI: 330.0 X 120.0  
XHDPI: 440.0 X 160.0  
XXHDPI: 660.0 X 240.0  
XXXHDPI: 880.0 X 320.0  
*/
```

```
public class DPICalculator {  
  
    private final float LDPI = 120;  
    private final float MDPI = 160;  
    private final float HDPI = 240;  
    private final float XHDPI = 320;  
    private final float XXHDPI = 480;  
    private final float XXXHDPI = 640;  
  
    private float forDeviceDensity;  
    private float width;  
    private float height;  
  
    public DPICalculator(float forDeviceDensity, float width, float height) {  
        this.forDeviceDensity = forDeviceDensity;  
        this.width = width;  
        this.height = height;  
    }  
  
    public static void main(String... args) {  
        DPICalculator dpiCalculator = new DPICalculator(240, 160, 120);  
        dpiCalculator.calculateDPI();  
    }  
  
    private float getPx(float dp, float value) {  
        float px = dp * (value / forDeviceDensity);  
        return px;  
    }  
  
    private void calculateDPI() {
```

```

float ldpiW = getPx(LDPI,width);
float ldpiH =  getPx(LDPI,height);
float mdpiW = getPx(MDPI,width);
float mdpiH =  getPx(MDPI,height);
float hdpiW = getPx(HDPI,width);
float hdpiH =  getPx(HDPI,height);
float xdpiW = getPx(XHDPI,width);
float xdpiH =  getPx(XHDPI,height);
float xxdpiW = getPx(XXHDPI,width);
float xxdpiH =  getPx(XXHDPI,height);
float xxxdpiW = getPx(XXXHDPI,width);
float xxxdpiH =  getPx(XXXHDPI,height);

System.out.println("LDPI: " + ldpiW + " X " + ldpiH);
System.out.println("MDPI: " + mdpiW + " X " + mdpiH);
System.out.println("HDPI: " + hdpiW + " X " + hdpiH);
System.out.println("XHDPI: " + xdpiW + " X " + xdpiH);
System.out.println("XXHDPI: " + xxdpiW + " X " + xxdpiH);
System.out.println("XXXHDPI: " + xxxdpiW + " X " + xxxdpiH);
}
}

```

More Information refer to the following link.

<http://javapapers.com/android/difference-between-dp-dip-sp-px-in-mm-pt-in-android/>

Share Improve this answer

Follow

edited Sep 9, 2021 at 17:06



Rahul

3,349 ● 2 ● 34 ● 44

answered Mar 4, 2016 at 5:36



Ravi Vaghela

3,430 ● 2 ● 24 ● 53



Here's the formula used by Android:

49

$$px = dp * (dpi / 160)$$



Where dpi is one of the following screen densities. For a list of all possible densities [go here](#)

It defines the "DENSITY_*" constants.

- ldpi (low) ~120dpi
- mdpi (medium) ~160dpi
- hdpi (high) ~240dpi
- xhdpi (extra-high) ~320dpi
- xxhdpi (extra-extra-high) ~480dpi
- xxxhdpi (extra-extra-extra-high) ~640dpi

Taken from [here](#).

This will sort out a lot of the confusion when translating between px and dp, if you know your screen dpi.

So, let's say you want an image of 60 dp for an hdpi screen then the physical pixel size of 60 dp is:

$$px = 60 * (240 / 160)$$

Share Improve this answer

Follow

edited Mar 9, 2018 at 6:24



Rohit Sharma

2,579 ● 6 ● 22 ● 40

answered Apr 30, 2015 at 4:14



Dan Borza

3,559 ● 3 ● 24 ● 16



46



- px - one pixel, same as to what is used in CSS, JavaScript, etc.
- sp - scale-independent pixels
- dip - density-independent pixels



Normally sp is used for font sizes, while dip is used (also called dp) for others.

Share Improve this answer

Follow

edited Jun 6, 2015 at 14:41



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Nov 13, 2014 at 16:53



DPC

624 ● 5 ● 10

In Android "px" means the size of a physical pixel on the screen. In CSS, 1 px = 1/96 of an inch. Not the same thing.

– Pedro Pedruzzi Nov 4, 2021 at 19:46 ✎



43



I've come across a good article about designing Android apps UI for different screen resolutions, and I'd like to leave it here just for somebody searching in this area. Yes, I know that it's somehow described in Google docs (and mentioned in the posts above), I read that but it was not good for me (yeah, I may be too stupid)). It remained unclear to me how to design layouts capable to handle





different screen sizes. I hate the DP concept and so on when I need to implement a "flexible" UI layout for different screens. (Hey iOS developers - yes, you're right it's a Storyboard concept).

Android has not bad UI concept, but lacks iOS Storyboard features, unfortunately. Designing flexible UI in Android is not an easy thing (at the best).

Here goes the article that helped me to understand what to do in Android to make layouts for different screen sizes:

[JMSTUDIO Blog:- Decide Android App Screen Size](#)

How to Design UI for Android Apps for Different Screen Size

To design an app UI for different screen sizes, our initial design has to meet a minimum required space for each screen size. Android defines a minimum size (in dp) for each generalized screen type. Here is an Android screen size guideline.

Minimum Screen Size for Android in dp (landscape)

Screen	Width	Height
xlarge	960dp	720dp
large	640dp	480dp
normal	470dp	320dp
small	426dp	320dp

When we get the screen size in dp, it is not enough for us to design the Android app UI. For each screen size, we need to prepare graphics

and bitmap images for each density. Here is an Android screen density guideline.

Android Density Guideline (dpi)

Density Name	DPI	Scaling Ratio
xhdpi	320	8
hdpi	240	6
mdpi	160	4
ldpi	120	3

For easy calculation, we can follow the 3:4:6:8 scaling ratio between the four generalized densities. If we create a 36×36 pixel picture for ldpi device, the rest densities pictures size will be 48×48 for mdpi, 72×72 for hdpi, and 96×96 for xhdpi.

How to Design Android Apps UI in Photoshop

Many designers have problems designing Android app UI in photoshop or another pixel based graphic design tools because of the density-independent unit, dp. Designers don't know how to map dp to pixel. Google also doesn't give a clear Android UI design guide for them, though they give a basic formula for dp and pixel translation.

As Android's definition, 1dp equal to 1px under 160 dpi device (mdpi). So we want to design an Android app for xlarge Android devices with mdpi density, we can define our UI size in pixel as 960 pixels in width and 720px in height; Follow the

same mapping rule, we can get following Android App screen size UI design guideline:

Android App Screen Size in Pixel Guideline				
	xlarge (960×720 dp)	large (640×480 dp)	normal (470×320 dp)	small (426×320 dp)
xhdp (320dpi)	1920×1440 pixel	1280×960 pixel	940×640 pixel	852×640 pixel
hdpi (240dpi)	1440×1080 pixel	960×720 pixel	705×480 pixel	639×480 pixel
mdpi (160dpi)	960×720 pixel	640×480 pixel	470×320 pixel	426×320 pixel
ldpi (120dpi)	720×540 pixel	480×360 pixel	352.5×240 pixel	319.5×240 pixel

ADDED: If you are interested in "flexible" UI too, have a look at this library: [An Android SDK that provides a new size unit - sdp \(scalable dp\). This size unit scales with the screen size](#) (this also mentioned in an answer here, about `SDP` library)

ADDED2 Google has finally understood the usefulness of the iOS Storeboard UI concept, and here goes `ConstraintLayout` for Android world: [Build a Responsive UI with ConstraintLayout](#)

Share Improve this answer

Follow

edited Sep 9, 2021 at 17:05



Rahul

3,349 ● 2 ● 34 ● 44

answered May 17, 2016 at 19:59



Mixaz

4,188 ● 1 ● 32 ● 56

1) `dp: (density independent pixels)`



34



The number of pixels represented in one unit of dp will increase as the screen resolution increases (when you have more dots/pixels per inch). Conversely on devices with lower resolution, the number of pixels represented in one unit of dp will decrease. Since this is a relative unit, it needs to have a baseline to be compared with. This baseline is a 160 dpi screen. This is the equation: $px = dp * (dpi / 160).$

2) **sp: (scale independent pixels)**

This unit scales according to the screen dpi (similar to dp) as well as the user's font size preference.

3) **px: (pixels)**

Actual pixels or dots on the screen.

For more details you can visit

[Android Developer Guide > Dimension](#)

[Android Developer Guide > Screens](#)

Share Improve this answer

Follow

edited Apr 2, 2018 at 13:58



RKRK

1,294 ● 5 ● 14 ● 18

answered May 10, 2015 at 12:59



31



Screen size in Android is grouped into categories `ldpi`, `mdpi`, `hdpi`, `xhdpi`, `xxhdpi` and `xxxhdpi`. **Screen density** is the amount of pixels within an area (like inch) of the screen. Generally it is measured in dots-per-inch (`dpi`).

PX(Pixels):

- our usual standard pixel which maps to the screen pixel. `px` is meant for absolute pixels. This is used if you want to give in terms of absolute pixels for width or height. Not recommended.

DP/DIP(Density pixels / Density independent pixels):

- `dip == dp`. In earlier Android versions `dip` was used and later changed to `dp`. This is alternative of `px`.
- Generally we never use `px` because it is absolute value. If you use `px` to set width or height, and if that application is being downloaded into different screen sized devices, then that view will not stretch as per the screen original size.
- `dp` is highly recommended to use in place of `px`. Use `dp` if you want to mention width and height to grow & shrink dynamically based on screen sizes.
- if we give `dp/dip`, android will automatically calculate the pixel size on the basis of 160 pixel

sized screen.

SP(Scale independent pixels):

- scaled based on user's font size preference. Fonts should use `sp`.
- when mentioning the font sizes to fit for various screen sizes, use `sp`. This is similar to `dp`. Use `sp` especially for font sizes to grow & shrink dynamically based on screen sizes

Android Documentation says:

when specifying dimensions, always use either `dp` or `sp` units. A `dp` is a density-independent pixel that corresponds to the physical size of a pixel at 160 `dpi`. An `sp` is the same base unit, but is scaled by the user's preferred text size (it's a scale-independent pixel), so you should use this measurement unit when defining text size

Share Improve this answer

answered Oct 7, 2015 at 8:47

Follow



Rajesh Jadav

12.9k ● 5 ● 55 ● 79



Pixel density

25

Screen pixel density and resolution vary depending on the platform. Device-independent pixels and scalable



pixels are units that provide a flexible way to accommodate a design across platforms.



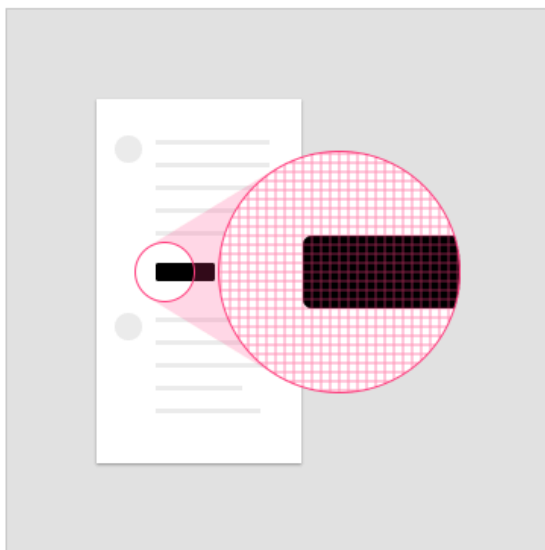
Calculating pixel density

The number of pixels that fit into an inch is referred to as pixel density. High-density screens have more pixels per inch than low-density ones...

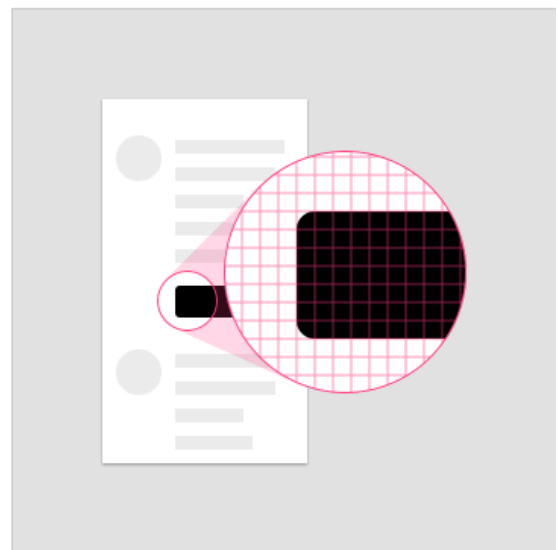
The number of pixels that fit into an inch is referred to as pixel density. High-density screens have more pixels per inch than low-density ones. As a result, UI elements of the same pixel dimensions appear larger on low-density screens, and smaller on high-density screens.

To calculate screen density, you can use this equation:

**Screen density = Screen width (or height) in pixels /
Screen width (or height) in inches**



High-density display



Lower density display

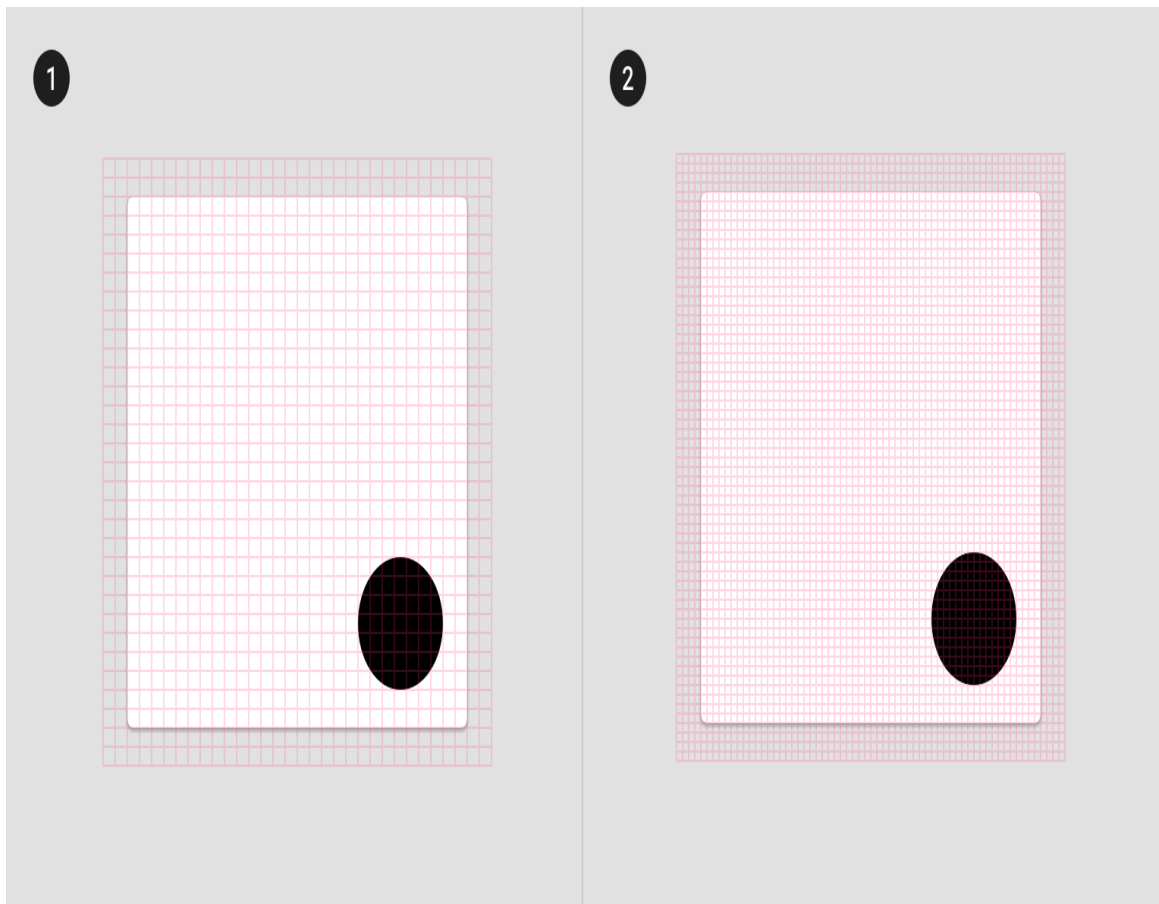
Density independence

Screen pixel density and resolution vary depending on the platform. Device-independent pixels and scalable pixels are units that provide a flexible way to accommodate a design across platforms.

Calculating pixel density The number of pixels that fit into an inch is referred to as *pixel density*. High-density screens have more pixels per inch than low-density ones...

Density independence refers to the uniform display of UI elements on screens with different densities.

Density-independent pixels, written as *dp* (pronounced “*dips*”), are flexible units that scale to have uniform dimensions on any screen. Material UIs use density-independent pixels to display elements consistently on screens with different densities.



1. Low-density screen displayed with density independence
2. High-density screen displayed with density independence

Read full text <https://material.io/design/layout/pixel-density.html>

Share Improve this answer

Follow

edited Sep 9, 2021 at 17:02



Rahul

3,349 ● 2 ● 34 ● 44

answered Aug 31, 2018 at 8:20



lava-lava

1,009 ● 10 ● 20



24



The screen of a mobile phone is made up of thousands of tiny dots known as **pixels (px)**. A pixel is the smallest element which goes to make the picture. The more the number of pixels to make a picture or wording, the sharper it becomes and makes the smartphone screen more easily readable.

Screen resolution is measured in terms of number of pixels on the screen. Screen resolution is a commonly-used specification when buying a device, but it's actually not that useful when designing for Android because thinking of screens in terms of pixels ignores the notion of physical size, which for a touch device is really really important.

Density independent pixel (dp or dip) allow the designer to create assets that appear in a expected way, no matter the resolution or density of target device.

A density independent pixel (dp or dip) is equal to one pixel at the baseline density or 160 dpi (dots per inch).

$$1 \text{ px}/1\text{dp} = 160 \text{ dpi}/160 \text{ dpi}$$

$$2 \text{ px}/1\text{dp} = 320 \text{ dpi}(2\text{x})/160 \text{ dpi}$$

where,

dpi is dots per inch

So, at 320 dpi, 1 dp is equal to 2 px.

Formula

$$\text{px/dp} = \text{dpi}/160\text{dpi}$$

Dots per inch (dpi) is a measure of the sharpness (that is, the density of illuminated points) on a display screen. The dots per inch for a given picture resolution will differ based on the overall screen size since the same number of pixels are being spread out over a different space.

Working with density independent pixels help us to deal with a situation like where you have two devices with same pixel resolution, but differing amount of space. Suppose in a case, a tablet and phone has the same pixel resolution 1280 by 800 pixels (160 dpi) and 800 by 1280 pixels (320 dpi) respectively.

Now because a tablet is at baseline density (160 dpi) its physical and density independent pixels sizes are the same, 1280 by 800. The phone on the other hand has a higher pixel density, so it has half as many density independent pixels as physical pixels. So a phone has 400 by 640 density independent pixels. So using a density-independent pixel makes it easier to mentally picture that tablet has much more space than the phone.

Similarly, if you have two devices with similar screen size, but different pixel density, say one is 800 by 1280 pixels (320 dpi), and the other is 400 by 640 pixels (160 dpi), we don't need to define totally different layouts for these two devices as we can measure assets in terms of density independent pixel which is same for both devices.

800 by 1280 pixels (320dpi)=400 by 640 density
independent pixel (dp)

400 by 640 pixels (160 dpi)=400 by 640 density
independent pixel (dp)

Scale independent pixels(sp) is the preferred unit for font size. For accessibility purposes, Android allows users to customize their device's font size. Users that have trouble reading text can increase their device's font size. You can normally find this option in the display setting on your phone or tablet under font size. It's often also available through the accessibility settings.

With scale independent pixels, 16 sp is exactly the same as 16 dp when the device's font size is normal or 100%. But when device's font size is large, for example 125%, 16 sp will translate to 20 dp or 1.25 times 16.

If you use dp as the unit for font size, then that piece of text has a specific physical size no matter if the user has customized device's font size. Using sp units will make a better experience for people with impaired eyesight.

Reference: [Udacity](#), Google

Share Improve this answer

Follow

edited Oct 18, 2017 at 5:31



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Jan 29, 2017 at 10:00



Android Developer

9,633 ● 20 ● 90 ● 145



20



sp: scale independent pixel

You should use it with texts because it is automatically scaled according to the font size that is being used by the user in his device.



px: pixel or picture element is the single point on the screen

Share Improve this answer

Follow

edited Mar 9, 2018 at 8:59



Rohit Sharma

2,579 ● 6 ● 22 ● 40

answered Jul 1, 2016 at 16:37



A.Saini

219 ● 2 ● 2



18

Pixels(px) – corresponds to actual pixels on the screen. This is used if you want to give in terms of absolute pixels for width or height.



Density-independent Pixels (dp or dip) – an abstract

unit that is based on the physical density of the screen.

These units are relative to a 160 dpi screen, so one dp is one pixel on a 160 dpi screen. The ratio of dp-to-pixel will change with the screen density, but not necessarily in direct proportion. Note: The compiler accepts both “dip” and “dp”, though “dp” is more consistent with “sp”.

Scale-independent Pixels(sp) – this is like the dp unit, but it is also scaled by the user’s font size preference. It is recommended you use this unit when specifying font sizes, so they will be adjusted for both the screen density and user’s preference.

Always use dp and sp only. sp for font sizes and dp for everything else. It will make UI compatible for Android devices with different densities. You can learn more about pixel and dp from

<https://www.google.com/design/spec/layout/units-measurements.html#units-measurements-density-independent-pixels-dp->

Source URL:- <http://www.androidtutorialshub.com/what-is-the-difference-between-px-dp-dip-sp-on-android/>

Share Improve this answer

Follow

edited Sep 9, 2021 at 17:04



Rahul

3,349 ● 2 ● 34 ● 44

answered May 10, 2017 at 7:37



lalit vasan

450 ● 3 ● 6



16



I want to provide an easy way to understand `dp`. In fact, I think `dp` is the easiest one to understand. `dp` is just a physical length unit. It's of the same dimension as `mm` or `inch`. It's just convenient for us to write `50dp`, `60dp` rather than `50/160 inch` or `60/160 inch`, because one `dp` is just `1/160 inch` whatever the screen size or resolution is.


The only problem is that, the android dpi of some screens are not accurate. For example, a screen classified to 160dpi may have 170dpi indeed. So the computation result of `dp` is fuzzy. It should be approximately the same as `1/160 inch`.

Share Improve this answer

edited Apr 2, 2016 at 3:35

Follow

answered Apr 2, 2016 at 3:19

 [ProtossShuttle](#)
1,673 ● 21 ● 41



13



SDP - a scalable size unit - basically it is not a unit, but dimension resources for different screen size.

Try the [sdp](#) library from Intuit. It's very handy to solve unit problems, and you can *quickly support multiple screens*.

Usage



`android:paddingBottom="@dimen/_15sdp"` for positive and
`android:layout_marginTop="@dimen/_minus10sdp"` for
negative sdp sdp

It has equivalent value in dp for each size in `values-sw<N>dp` folders (sw = smallestWidth).

Attention

Use it carefully! In most cases you still need to design a different layout for tablets.

Example

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="@dimen/_minus10sdp"
    android:paddingBottom="@dimen/_15sdp"
    android:orientation="horizontal" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:includeFontPadding="false"
        android:text="♡"
        android:textColor="#ED6C27"
        android:textSize="@dimen/_70sdp"
        android:textStyle="bold" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:includeFontPadding="false"
        android:text="U"
        android:textColor="@android:color/
        android:textSize="@dimen/_70sdp" /
</LinearLayout>
```

You can use `db` for text size, but I prefer [ssp](#) for text size.

For more details, check [the library GitHub page](#).

Share Improve this answer

Follow

edited Oct 18, 2017 at 5:26



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Dec 27, 2016 at 14:17



Qamar

5,135 ● 3 ● 32 ● 51



The ratio of **dp-to-pixel** will change with the screen density, but not necessarily in direct proportion.

8



Note: The compiler accepts both "**dip**" and "**dp**", though "**dp**" is more consistent with "**sp**".



Scale-independent Pixels - this is like the **dp** unit, but it is also scaled by the user's font size preference.

Share Improve this answer

Follow

answered Mar 3, 2017 at 18:59



Narendra Sorathiya

3,830 ● 2 ● 36 ● 38

1

2

Next



Highly active question. Earn 10 reputation (not counting the [association bonus](#)) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.

