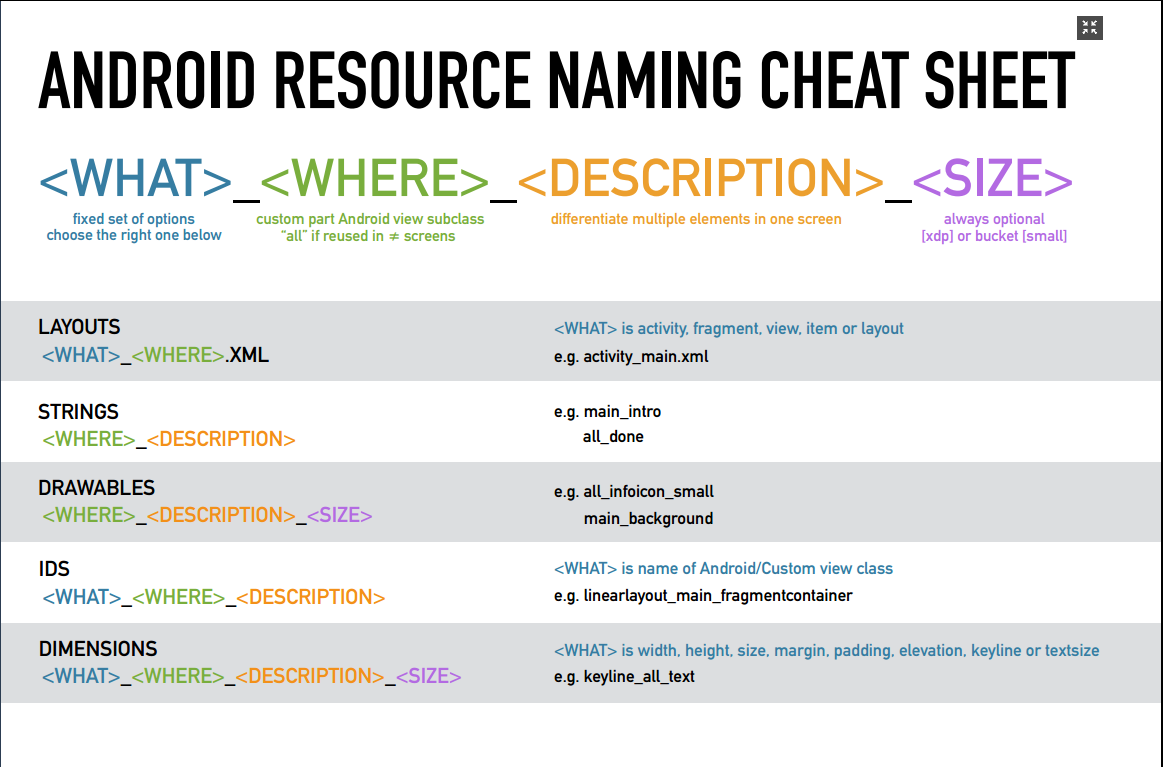
# **CHEAT SHEET**



# **BASIC PRINCIPLE**

All resource names follow a simple convention.

**<WHAT>\_<WHERE>\_<DESCRIPTION>\_<SIZE>**

**<WHAT>**   
indicates what the resource actually represents, often a standard Android view class.  
Limited options per resource type.  
(e.g. MainActivity → activity)

**<WHERE>**   
describes where it logically belongs in the app. Resources used in multiple screen use all,  
all others use the custom part of the Android view subclass they are in.  
(e.g. MainActivity → main, BookDetailFragment → bookdetail)

**<DESCRIPTION>**   
differentiates multiple elements in one screen.  
 (e.g. title, hint)

**<SIZE>** (OPTIONAL)   
either a precise size or size bucket. Optionally used for drawables and dimensions.  
(e.g. 24dp, small)

# **ADVANTAGES**

1. **Ordering of resource by screen**

The **WHERE** part describes what screen a resource belongs to. Hence it is easy to get all IDs, drawables, dimensions,... for a particular screen.

1. **Strongly typed resource IDs**

For resource IDs, the **WHAT** describes the class name of the xml element it belongs to. This makes is easy to what to cast your findViewById() calls to.

1. **Better resource organizing**

File browsers/project navigator usually sort files alphabetically. This means layouts and drawables are grouped by their **WHAT** (activity, fragment,...) and **WHERE** prefix respectively.

1. **More efficient autocomplete**

Because resource names are far more predictable, using the IDE's autocomplete becomes even easier. Usually entering the **WHAT** or **WHERE** is sufficient to narrow autocomplete down to a limited set of options.

1. **No more name conflicts**

Similar resources in different screens are either all or have a different **WHERE**. A fixed naming scheme avoids all naming collisions.

1. **Cleaner resource names**

Overall all resources will be named more logical, causing a cleaner Android project.

1. **Tools support**

This naming scheme could be easily supported by the Android Studio offering features such as: lint rules to enforce these names, refactoring support when you a **WHAT** or **WHERE**, better resource visualisation in project view,...

# **LAYOUTS**

Layouts are relative simple, as there usually are only a few layouts per screen. Therefore the rule can be simplified to:

**<WHAT>\_<WHERE>.XML**

Where **WHAT** is one of the following:

|  |  |
| --- | --- |
| Prefix | Usage |
| activity | content view for activity |
| fragment | view for a fragment |
| view | inflated by a custom view |
| item | layout used in list/recycler/gridview |
| layout | layout reused using the include tag |

Examples:

* **activity\_main**: content view of the MainActivity
* **fragment\_article\_detail**: view for the ArticleDetailFragment
* **view\_menu**: layout inflated by custom view class MenuView
* **item\_article**: list item in ArticleRecyclerView
* **layout\_actionbar\_backbutton**: layout for an actionbar with a backbutton

# **STRINGS**

The **WHAT** part for Strings is irrelevant. So either we use **WHERE** to indicate where the string will be used:

**<WHERE>\_<DESCRIPTION>**

or all if the string is reused throughout the app:

**all\_<DESCRIPTION>**

Examples:

* **articleDetail\_title**: title of ArticleDetailFragment
* **feedback\_explanation**: feedback explanation in FeedbackFragment
* **feedback\_name\_hint**: hint of name field in FeedbackFragment
* **all\_done**: generic “done” string

**WHERE** obviously is the same for all resources in the same view.

# **DRAWABLES**

The **WHAT** part for Drawables is irrelevant. So either we use **WHERE** to indicate where the string will be used:

**<WHERE>\_<DESCRIPTION>\_<SIZE>**

or all if the string is reused throughout the app:

**all\_<DESCRIPTION>\_<SIZE>**

Optionally you can add the **SIZE** argument, which can be an actual size “24dp” or a size qualifier “small”.

Examples:

* **articleDetail\_placeholder**: placeholder in ArticleDetailFragment
* **all\_info\_icon**: generic info icon
* **all\_info\_icon\_large**: large version of generic info icon
* **all\_info\_icon\_24dp**: 24dp version of generic info icon

# **IDs**

For IDs, **WHAT** is the class name of the xml element it belongs to. Next is the screen the ID is in, followed by an optional description to distinguish similar elements in one screen.

**<WHAT>\_<WHERE>\_<DESCRIPTION>**

Examples:

* **tablayout\_main**: TabLayout in MainActivity
* **imageview\_menu\_profile**: profile image in custom MenuView
* **textview\_articleDetail\_title**: title TextView in ArticleDetailFragment

# **DIMENSIONS**

Apps should only define a limited set of dimensions, which are constanly reused. This makes most dimensions all by default.

Therefore you should mostly use:

**<WHAT>\_all\_<DESCRIPTION>\_<SIZE>**

and optionally use the screen specific variant:

**<WHAT>\_<WHERE>\_<DESCRIPTION>\_<SIZE>**

Where **WHAT** is one of the following:

|  |  |
| --- | --- |
| Prefix | Usage |
| width | width in dp |
| height | height in dp |
| size | if width == height |
| margin | margin in dp |
| padding | padding in dp |
| elevation | elevation in dp |
| keyline | absolute keyline measured from view edge in dp |
| textsize | size of text in sp |

Note that this list only contains the most used **WHAT.** Other dimensions qualifiers like: rotation, scale,… are usually only used in drawables and as such less reused.

Examples:

* **height\_all\_toolbar**: height of all toolbars
* **keyline\_listtext:** listitem text is aligned at this keyline
* **textsize\_all\_medium:** medium size of all text
* **size\_menu\_icon:** size of icon in menu
* **height\_menu\_profile\_image:** height of profile image in menu

# **COMPONENT NAME IN CODE**

In source code, to make the easier to read, we can use short name prefix for major components

* Button - btn
* EditText - et
* TextView - tv
* Checkbox - chk
* RadioButton - rdb
* ToggleButton - tb
* Spinner - spn
* ListView - lv
* GalleryView - gv
* LinearLayout -ll
* RelativeLayout – rl
* SeekBar – skb
* RatingBar - rtb