Reto Meier Follow
Developer Advocate @ Google, software engineer, and author of Professional Android Application Dev...
Apr 15, 2016 · 5 min read

(About) 10 Things You (Probably) Didn't Know You Could do in Android Studio

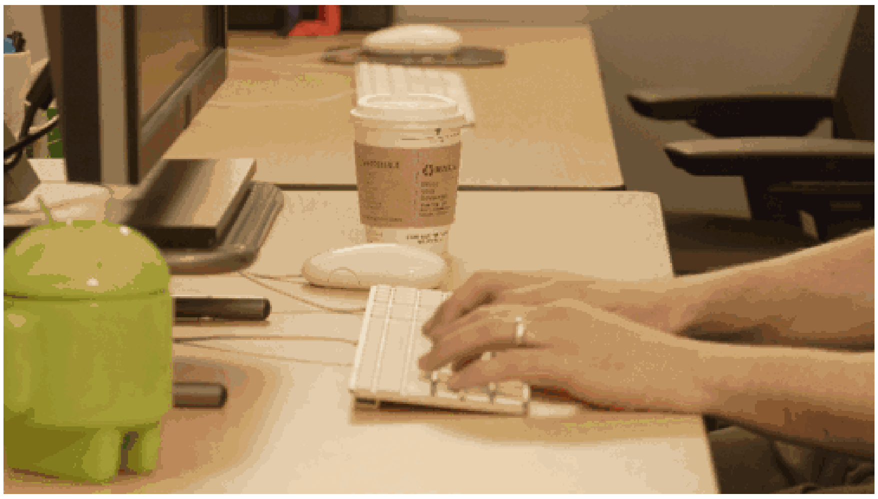
An Android Tool Time Pro Tip Roundup

We all have better things to do with our time than count the *exact* number of Android Studio pro-tips in a 3 minute video—check it out for yourself and see how many are new to you.

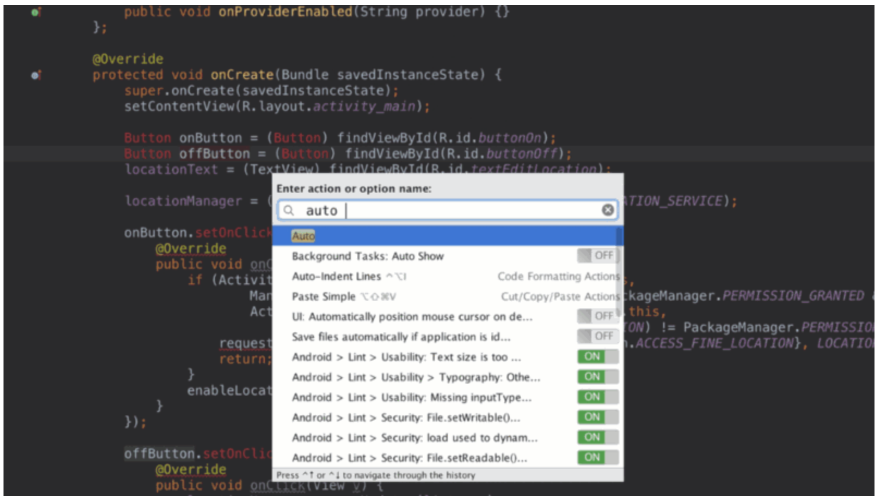
> *The video moves pretty quickly, so keep reading to see most of the tips and tricks featured in written / animated GIF form for easy reference.*

An over-dependence on using your mouse while coding can have results more serious than just lower productivity. These pro tips are designed to help you write less code, and make every keystroke count, so you can avoid situations like this one.

· · ·


Thanks Obama.

Most of these tips are available as part of IntelliJ, on which Android Studio is built. The most important shortcut to remember in Android Studio is CMD-SHIFT-A (or CTRL-SHIFT-A if you're on a Windows or Linux PC).



```
            public void onProviderEnabled(String provider) {}
        };

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            Button onButton = (Button) findViewById(R.id.buttonOn);
            Button offButton = (Button) findViewById(R.id.buttonOff);
            locationText = (TextView) findViewById(R.id.textEditLocation);

            locationManager = (                                      TION_SERVICE);

            onButton.setOnClick
                @Override
                public void onC
                    if (Activit                                        ,
                            Mar                                       kageManager.PERMISSION_GRANTED
                            Act                                       this,
                                                                      ON) != PackageManager.PERMISSIO
                        request                                       .ACCESS_FINE_LOCATION}, LOCATIO
                        return;
                    }
                    enableLocat
                }
            });

            offButton.setOnClic
                @Override
                public void onClick(View v) {
```

Enter action or option name:

auto

```
Auto
Background Tasks: Auto Show                              OFF
Auto-Indent Lines ^⌥I                    Code Formatting Actions
Paste Simple ⌥⇧⌘V                          Cut/Copy/Paste Actions
UI: Automatically position mouse cursor on de...         OFF
Save files automatically if application is id...        OFF
Android > Lint > Usability: Text size is too ...         ON
Android > Lint > Usability > Typography: Othe...         ON
Android > Lint > Usability: Missing inputType...         ON
Android > Lint > Security: File.setWritable()...         ON
Android > Lint > Security: load used to dynam...         ON
Android > Lint > Security: File.setReadable()...         ON
Press ^↑ or ^↓ to navigate through the history
```

Use CMD-SHIFT-A or CTRL-SHIFT-A to find actions or options

After hitting that shortcut, you can just typing keywords and available actions and options will be available for your each selection. It's a great way to start using new features before you've memorize their shortcuts.

You can use a similar approach anywhere there's a long list of options. If you're trying to find a file in the project hierarchy, or select an option from large menu such as *Refactor This…*, just start typing and it'll start searching and filtering results.

Using Tab to Replace Existing Methods and Values with Autocomplete Selections

Pressing Tab replaces existing methods and values rather than just inserting a new one.

Bringing up autocomplete with CTRL-SPACE (or CTRL-SHIFT-SPACE for options of the expected type) is probably one of the most commonly used shortcuts in Android Studio.

And everyone's experienced the mild annoyance of using this technique to choose a new method, or select a different variable, hitting enter, and having the new selection get inserted in front of the existing one.

If you hit TAB instead of ENTER, it'll *replace* the existing method or value instead. You're welcome.

Two Text Selection Tricks

Up, down, left, right, and a combination of CTRL, SHIFT, and Fn covers most of your cursor navigation and selection needs—but the ALT modifier adds some new and unexpected flavor.

You can use ALT-UP and ALT-DOWN to extend and contract your selection by "node"—effectively letting you modify selection by scope.

Meanwhile, ALT-SHIFT-UP and ALT-SHIFT-DOWN will let you move a selection up and down respectively, conveniently moving other code out of the way without the need to resort to anything as pedestrian as cut and paste.

Postfix Code Completion and Live Templates

If I'd been paid a dollar for every for loop, if-statement, and log statement I've typed I'd… actually that might work out to be pretty close to accurate.

In the spirit of getting paid more for typing less, take advantage of postfix code completion and Live Templates to insert some of the most common code patterns with quick shortcuts.

Use postfix code completion lets you transform an already typed expression into another one.

You can create a for-loop over a list using the *.fori* shortcut, or a boolean variable (or expression) into an if statement using *.if* (or *.else*). You can use see all the valid postfixes available for a given context by typing CMD-J (or CTRL-J on Windows / Linux).

For more complex patterns, Live Templates let you use shortcuts that are available as autocompletion options that will insert templatized snippets into your code. For example, using the Toast shortcut makes it easy to add a new toast, where you only need to specify the text to display.

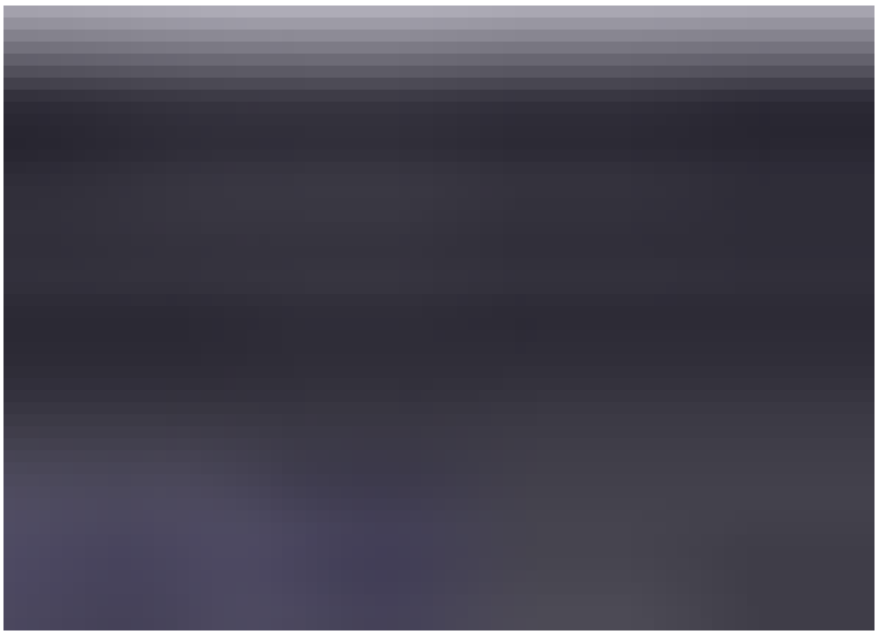
There are dozens of generic and Android specific Live Templates, including a selection of logging shortcuts—or you can create your own to simplify best-practice patterns or boilerplate within your own code.

Custom Rendering for Objects when Evaluating Expressions

When debugging code at runtime—looking at variable values at a breakpoint or evaluating expressions—objects are displayed using their .toString() values. If your variable is a String or a primitive type that works well, but for most objects it's not particularly helpful.

This is especially true of collections of objects, which are typically displayed as a list of "ClassName:HashValue". To understand the state, you need to dig into each object.

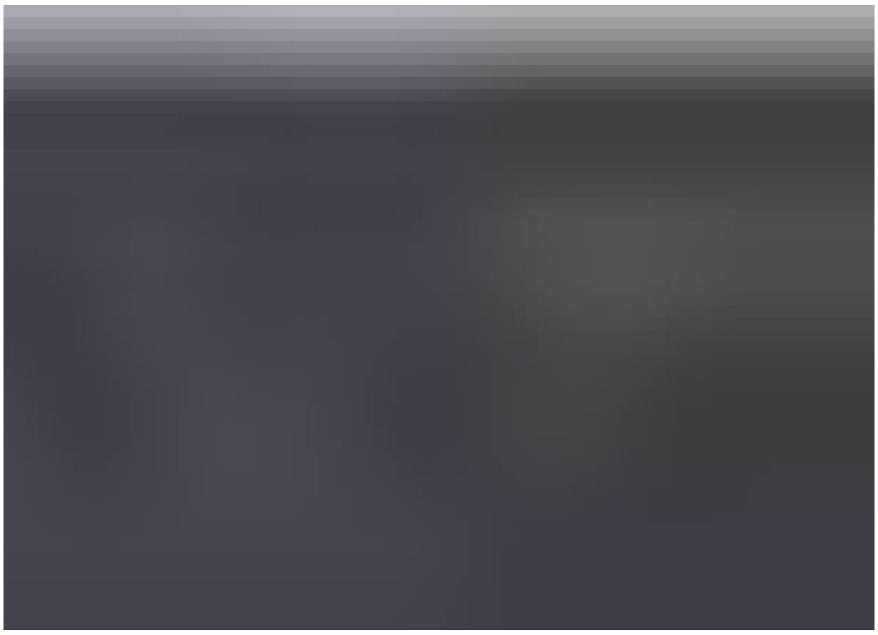
Instead, you can create a custom renderer for any object type.



Right click on the object, select "View as" → Create… and define the expression you want to use to render that object when debugging.

Structural Search, Replace, and Inspection

Structural search and replace let you search for, and replace (respectively), code patterns without resorting to Regular Expressions.


Structural Replace Inspections Let You Create Your Own Lint Checks with Quickfixes

Even more usefully, you can enable Structural Search Inspection. You can then save a Structural Search Templates that will flag code that matches that pattern warning, displaying the hint text you provide.

Use it to flag anti-patterns within your code (or code you're reviewing).

Even more powerfully, you can create Structural Replace Templates. Like Structure Search Templates, the pattern will be flagged as a warning—but in this case, the replacement code will be offered as a quick fix!

This is perfect for creating quick fixes for deprecated code, or common anti-patterns in code your reviewing, or which is being submitted by other team members.

. . .

There are hundreds of tips and tricks to make your Android Studio experience faster, more productive, and (mostly) mouse-free. Subscribe to Android Developers on YouTube, and tune in to Android Tool Time for more Android Tool Time Pro Tips.