Android Studio Essentials

Your quick guide to mastering Android Studio with essential shortcuts, handy tips, and core usage patterns for faster development.

Essential Keyboard Shortcuts

Navigation & Search

`Ctrl + N` / `Cmd + O`	Find class by name.
`Ctrl + Shift + N` / `Cmd + Shift + O`	Find file by name (including resources).
`Ctrl + F` / `Cmd + F`	Find text within the current file.
`Ctrl + Shift + F` / `Cmd + Shift + F`	Find text in the entire project.
`Ctrl + G` / `Cmd + L`	Go to line number.
`Ctrl + E` / `Cmd + E`	Show recent files popup.
`Alt + Left/Right` / `Ctrl + Left/Right` or `Cmd + [` / `Cmd + ]`	Navigate back/forward through cursor positions.
`Ctrl + B` or `Ctrl + Click` / `Cmd + B` or `Cmd + Click`	Go to declaration or usage.
`Alt + F7` / `Option + F7`	Find usages of the symbol at the caret.

Editing & Code Generation

`Ctrl + Space`	Basic code completion.
`Ctrl + Shift + Space`	Smart code completion (filters by expected type).
`Alt + Enter` / `Option + Return`	Show intention actions and quick-fixes.
`Ctrl + O` / `Cmd + O` (within class)	Override methods.
`Ctrl + I` / `Cmd + I`	Implement methods.
`Ctrl + Alt + T` / `Cmd + Option + T`	Surround with (if, try/catch, etc.).
`Ctrl + D` / `Cmd + D`	Duplicate current line or selection.
`Ctrl + Y` / `Cmd + Backspace`	Delete line at caret.
`Ctrl + Alt + L` / `Cmd + Option + L`	Reformat code.

Build & Run

`Shift + F10` / `Ctrl + R`	Run the current configuration.
`Shift + F9` / `Ctrl + D`	Debug the current configuration.
`Ctrl + F9` / `Cmd + F9`	Make project (compile changed files).
`Ctrl + Shift + F10` / `Ctrl + Shift + R`	Run context configuration (e.g., a specific test).
`Ctrl + F5`	Rerun the last run configuration.
`Alt + Shift + F10` / `Ctrl + Option + R`	Select and run configuration.
`Alt + Shift + F9` / `Ctrl + Option + D`	Select and debug configuration.

Refactoring

`Shift + F6` / `Shift + F6`	Rename.
`Ctrl + Alt + M` / `Cmd + Option + M`	Extract Method.
`Ctrl + Alt + V` / `Cmd + Option + V`	Extract Variable.
`Ctrl + Alt + F` / `Cmd + Option + F`	Extract Field.
`Ctrl + Alt + C` / `Cmd + Option + C`	Extract Constant.
`Ctrl + Alt + P` / `Cmd + Option + P`	Extract Parameter.
`F6` / `F6`	Move.
`F5` / `F5`	Copy.

Tips, Tricks & Usage

Layout Editor Tips

Design + Blueprint View: Toggle between Design, Blueprint, or both views for layout construction (`B` key).
Constraint Inference: For `ConstraintLayout`, use the 'Infer Constraints' magic wand icon to automatically add basic constraints.

Attributes Panel: Quickly find attributes using the search bar in the Attributes panel. Star favorite attributes for quick access.
Tools Attributes: Use the `tools:` namespace (e.g., `tools:text="Sample"`) for placeholder data visible only in the Layout Editor, not in the final app.
Layout Validation: Use the Layout Validation tool (View > Tool Windows > Layout Validation) to preview layouts on different screen sizes and configurations simultaneously.
Convert View: Right-click a View in the Component Tree or Design surface to convert it to another type (e.g., `TextView` to `Button`).
Split View: Use the 'Split' mode (top-right icons) to see the XML code and the visual preview side-by-side.

Debugging Techniques

Conditional Breakpoints: Right-click a breakpoint and enter a condition (Java/Kotlin expression). The debugger will only stop if the condition evaluates to true.
Evaluate Expression: While debugging (`Alt + F8` / `Option + F8`), evaluate arbitrary code expressions in the current context.
Logpoints: Instead of stopping execution, right-click a line number and choose 'Log message to console'. Useful for tracing without pausing.
Analyze Stack Trace: Clickable links in the Logcat or Run window allow you to jump directly to the source code line mentioned in a stack trace.
Attach Debugger to Process: If the app is already running, use 'Run > Attach Debugger to Android Process' (`Shift + Alt + F9` / `Ctrl + Option + D` then select process) to start debugging without restarting.
View > Tool Windows > Debug: Access breakpoints, variables, watches, and the call stack.
Mark Object: In the Variables view, right-click an object and select 'Mark Object' to easily track its instances across different scopes or times.
Layout Inspector: While debugging, use 'Tools > Layout Inspector' to examine the view hierarchy and attributes of your running app's UI.

Version Control (Git)

`Ctrl + K` / `Cmd + K`	Commit changes.
`Ctrl + T` / `Cmd + T`	Update project (pull).
`Ctrl + Shift + K` / `Cmd + Shift + K`	Push commits.
`Alt + \` / `Ctrl + V`	Show VCS operations popup.
`Alt + 9` / `Cmd + 9`	Show Version Control tool window (Log, Local Changes, etc.).
Branch Management	Use the branch widget in the bottom-right status bar to view, create, checkout, merge, and manage branches.
Show History	Right-click a file or folder -> Git -> Show History.
Resolve Conflicts	Use the built-in merge tool (VCS -> Git -> Resolve Conflicts) for a visual conflict resolution interface.

Gradle & Build System

Sync Project with Gradle Files: Click the 'Sync Now' banner after build script changes, or use the elephant icon with a refresh arrow in the toolbar (`Ctrl + Shift + Alt + S` -> Project Structure -> Suggestions).
Gradle Tool Window: View > Tool Windows > Gradle. See tasks, dependencies, and run specific Gradle tasks.
Build Variants: Use the 'Build Variants' tool window (View > Tool Windows > Build Variants) to switch between debug/release builds and product flavors.
Project Structure Dialog: `Ctrl + Alt + Shift + S` / `Cmd + ;` to manage dependencies, SDK locations, build variants, signing configs, etc.
Analyze Dependencies: Right-click a module in the Gradle tool window and select 'Analyze Dependencies' or use the Project Structure dialog.
Clean Project: Build > Clean Project. Removes build artifacts.
Rebuild Project: Build > Rebuild Project. Cleans and then builds the entire project.
Use `gradlew` from Terminal: Access the embedded terminal (`Alt + F12`) and run Gradle commands directly (e.g., `./gradlew assembleDebug`).