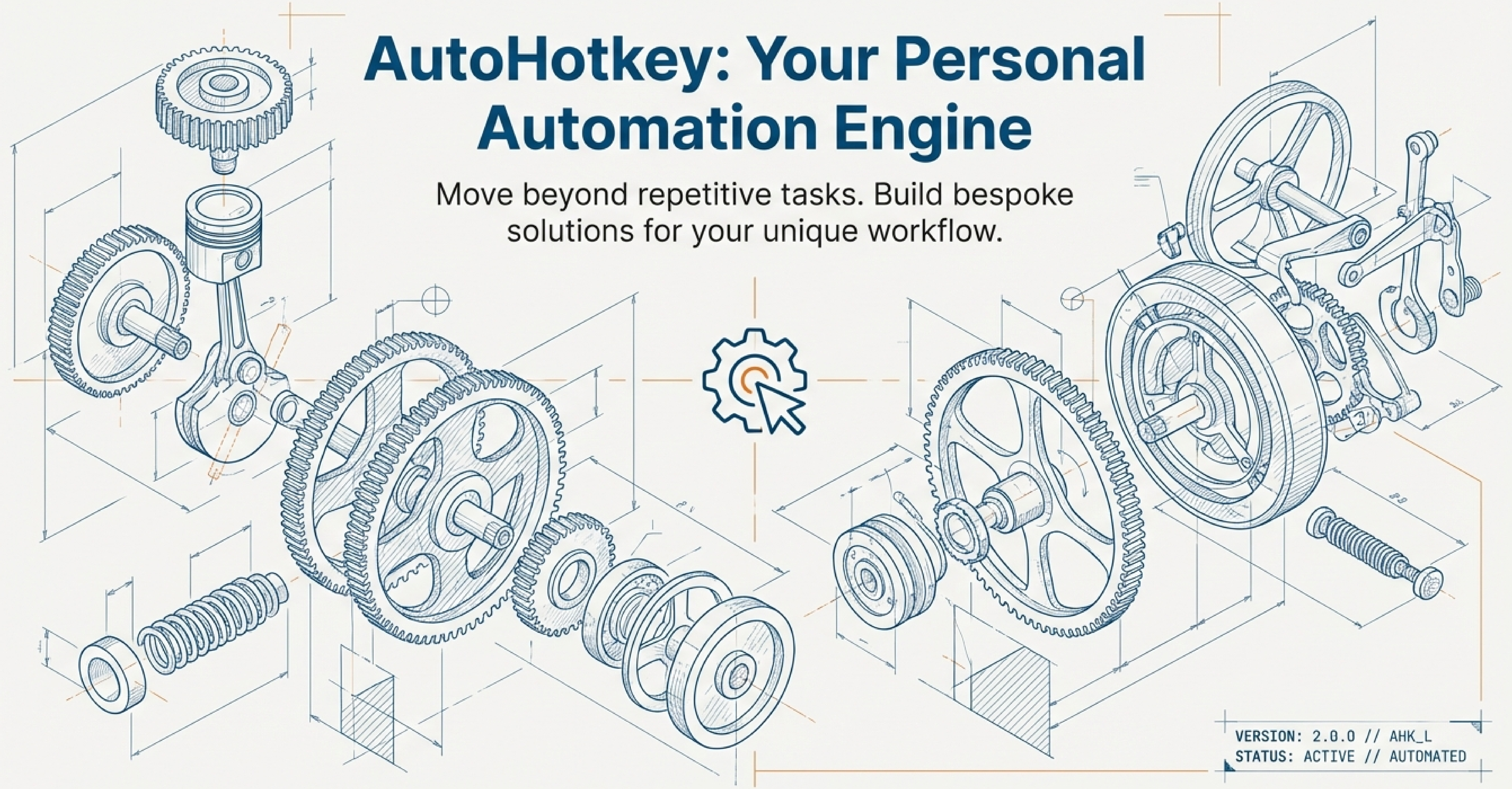
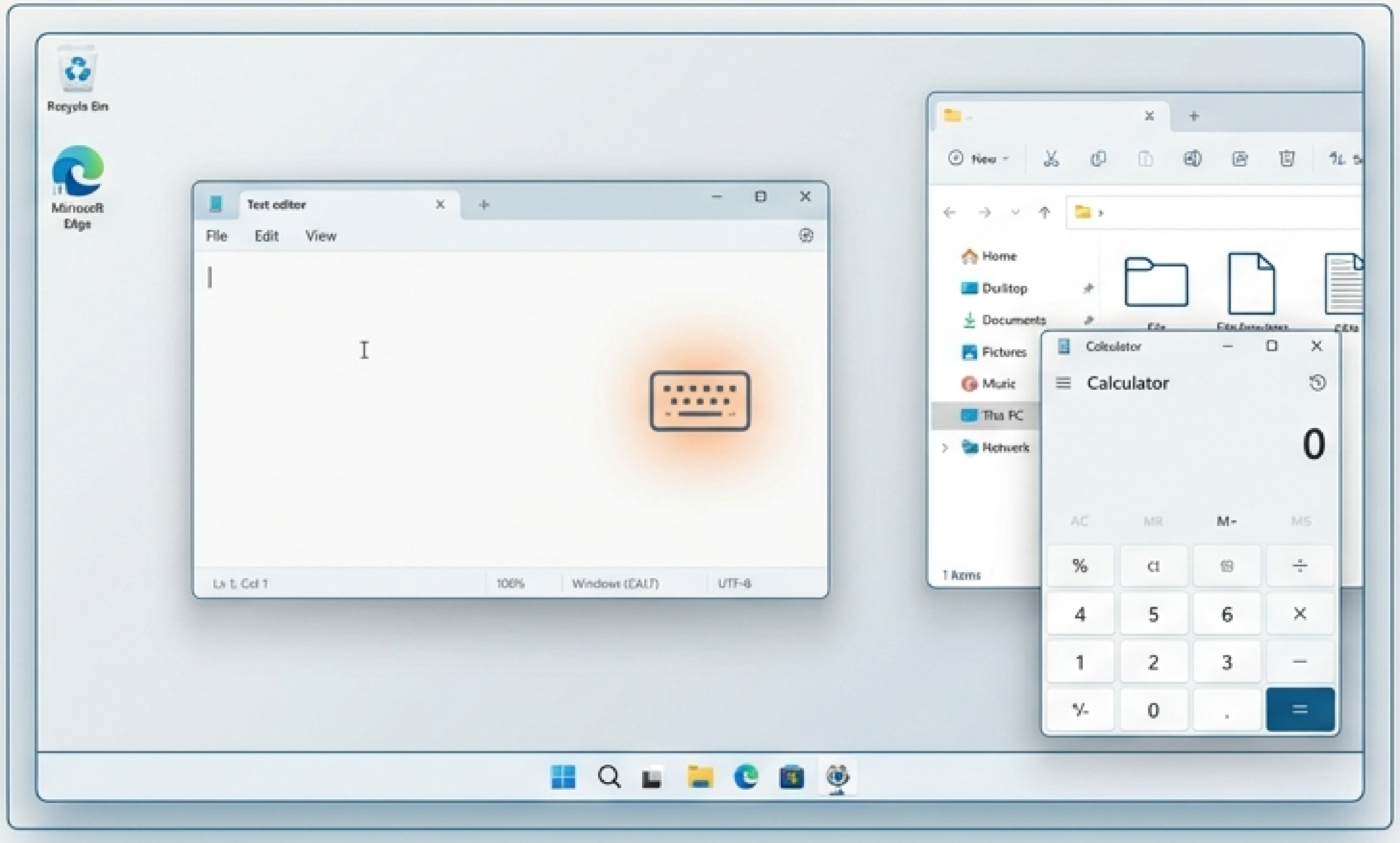
AutoHotkey: Your Personal Automation Engine

Move beyond repetitive tasks. Build bespoke solutions for your unique workflow.



VERSION: 2.0.0 // AHK_L
STATUS: ACTIVE // AUTOMATED

Go from a 5-Click Process to a 1-Key Solution



This entire sequence—opening a folder, launching a program, and typing text—was triggered by a single custom hotkey. This is the power you can build.

The Three Building Blocks of Automation

TRIGGERS

The event that starts your automation. This can be a key you press (a **Hotkey**) or a word you type (a **Hotstring**).

ACTIONS

The work your automation performs. This includes sending keystrokes, clicking the mouse, running programs, opening websites, and more.

CONTEXT

The ‘brains’ of the operation. This allows you to make your Triggers work only in specific windows or situations, giving your automations intelligence and precision.

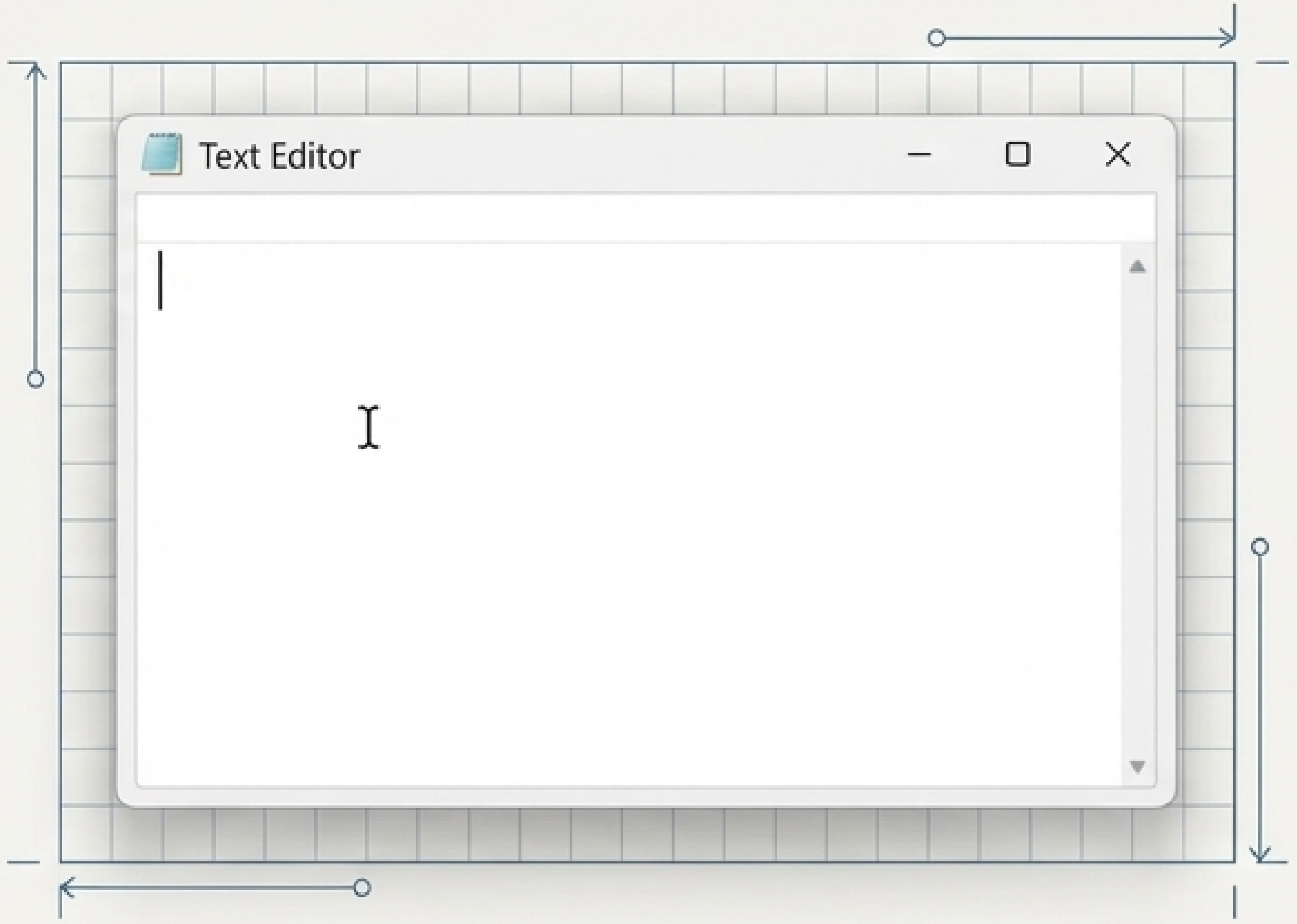
Your First Build: Trigger + Action for Effortless Text Expansion

THE BLUEPRINT

```
::btw::By the way
```

This **Hotstring** is a **TRIGGER**. When you type ‘btw' followed by a space or enter, AHK performs an **ACTION**: it replaces the trigger text with 'By the way'.

THE RESULT

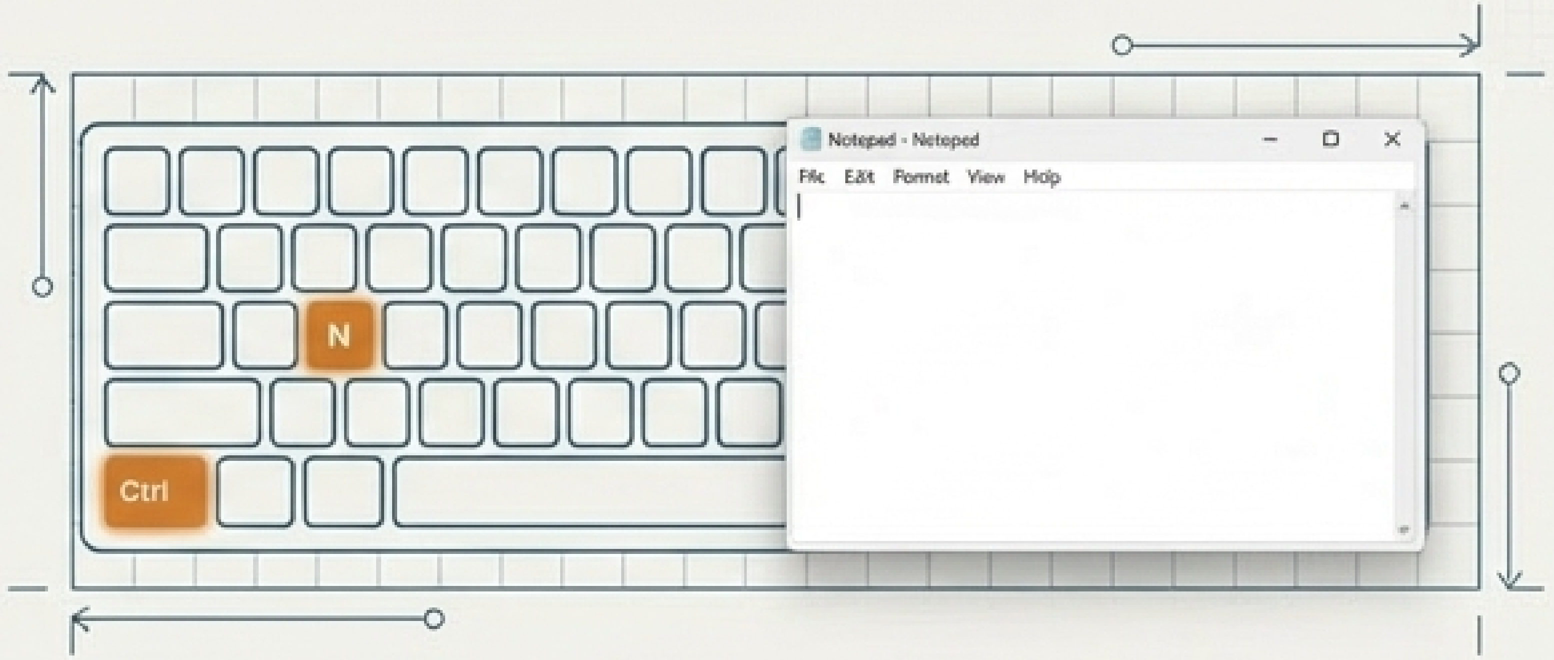


Taking Control: Use Hotkeys to Launch Programs and Websites Instantly

Launch a Program

```
^n::Run "notepad.exe"
```

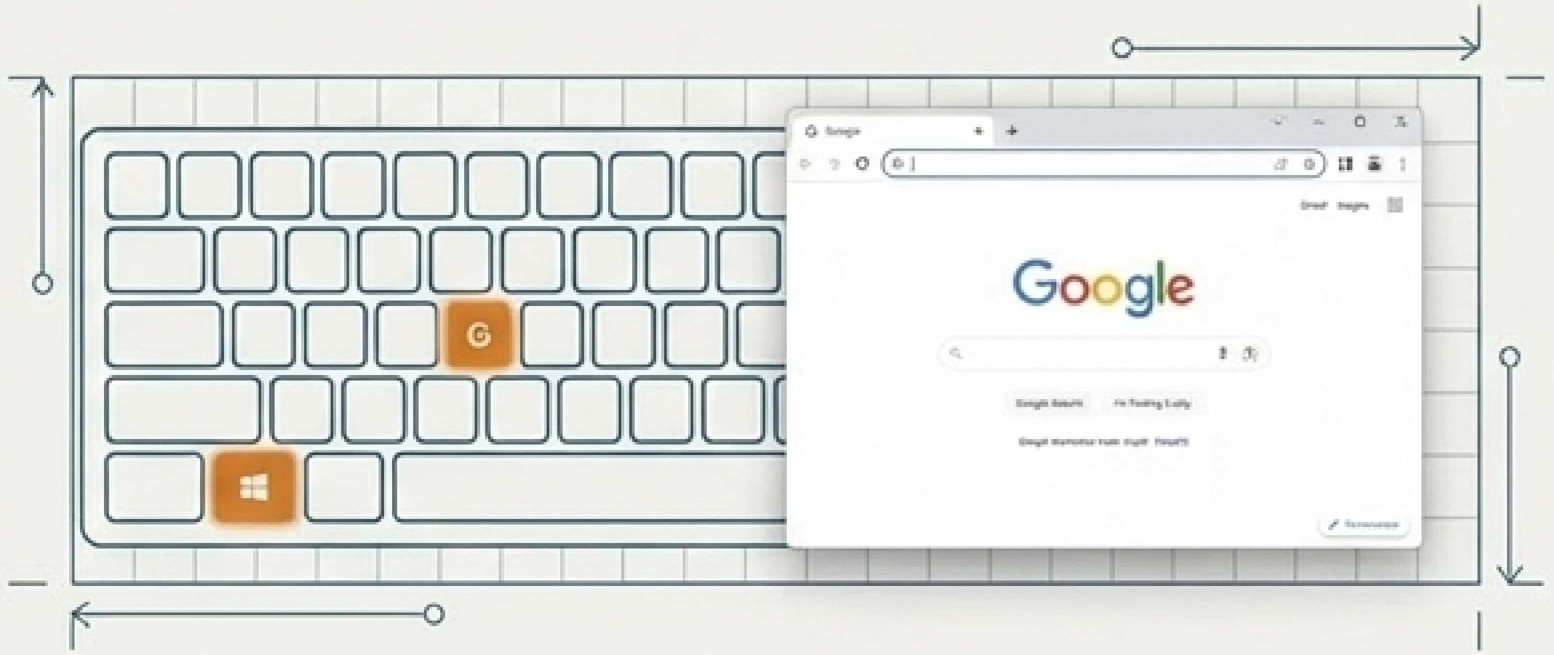
Note: `^` is the symbol for the `Ctrl` key. This hotkey is Ctrl+N.



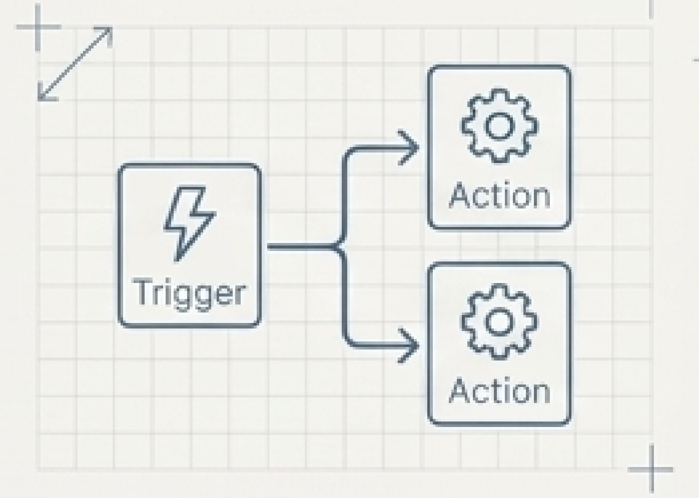
Open a Website

```
#g::Run "http://www.google.com"
```

Note: `#` is the symbol for the `Windows` key. This hotkey is Win+G.



Building a Workflow: One Trigger, Multiple Actions



THE BLUEPRINT

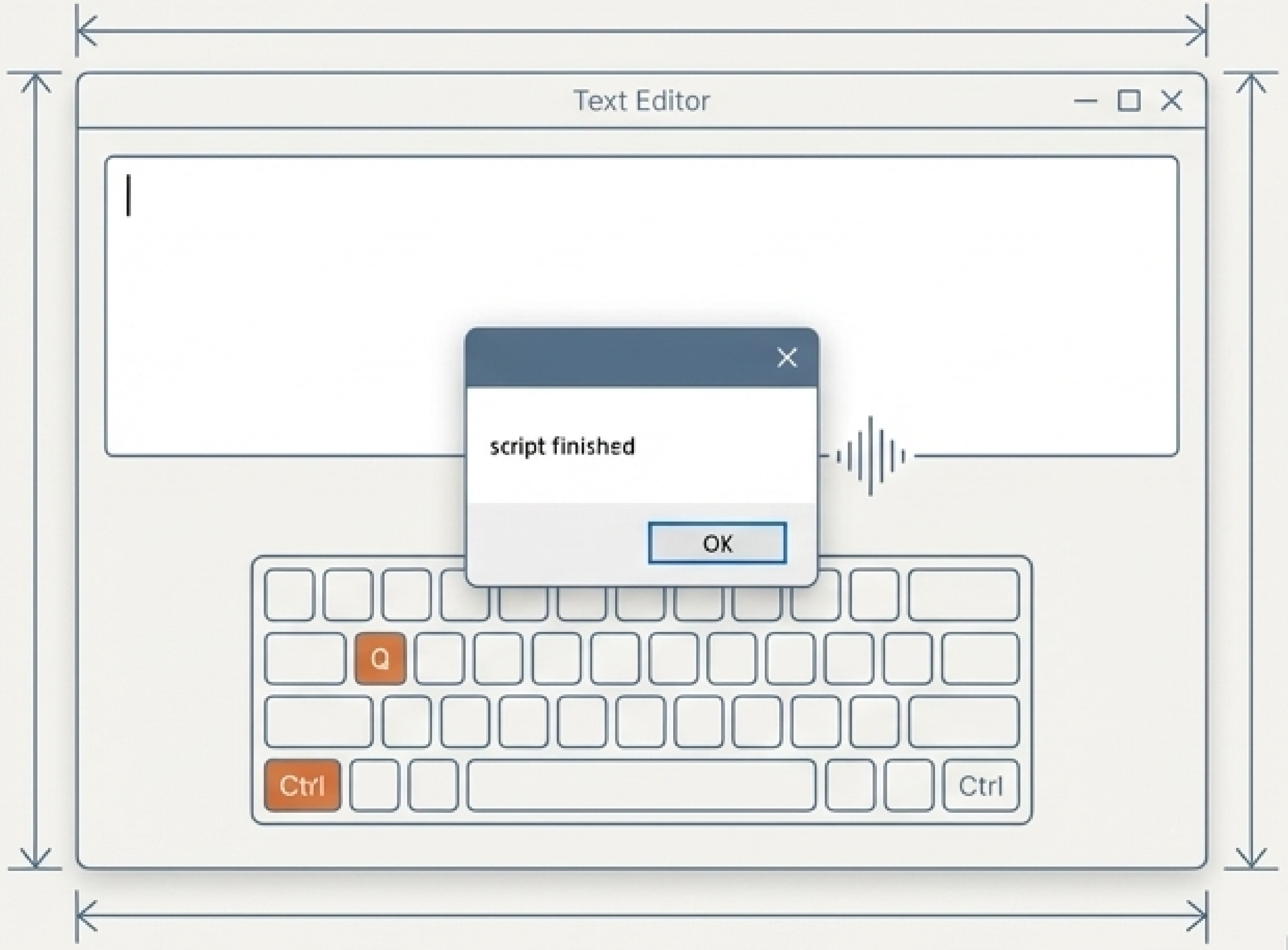
```
^q::
{
    Send("This is Anders Jensen")
    MsgBox("script finished")
}
Return
```

When you press `Ctrl+Q`, the script performs two actions in sequence:

1. It types the text “**This is Anders Jensen**”.
2. It then displays a message box that says “**script finished**”.

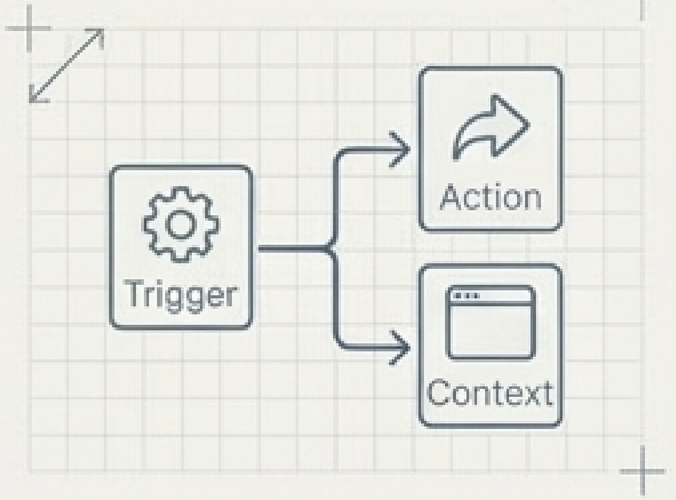
Each command needs to be on a separate line.

THE RESULT



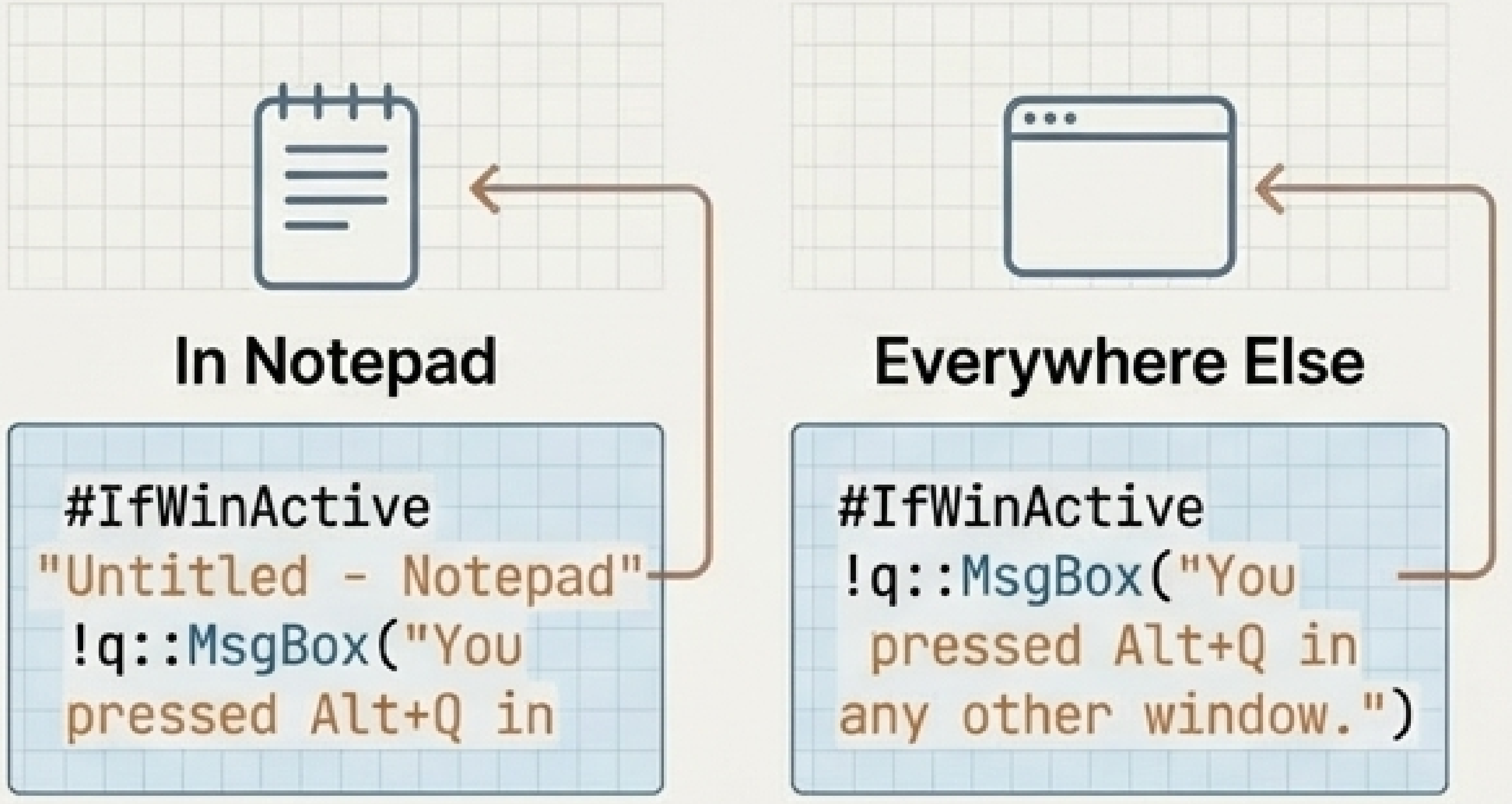
script finished

Adding Intelligence: Making Hotkeys Context-Aware



With the `#IfWinActive` directive, you can make a hotkey perform different actions depending on which window is currently active.

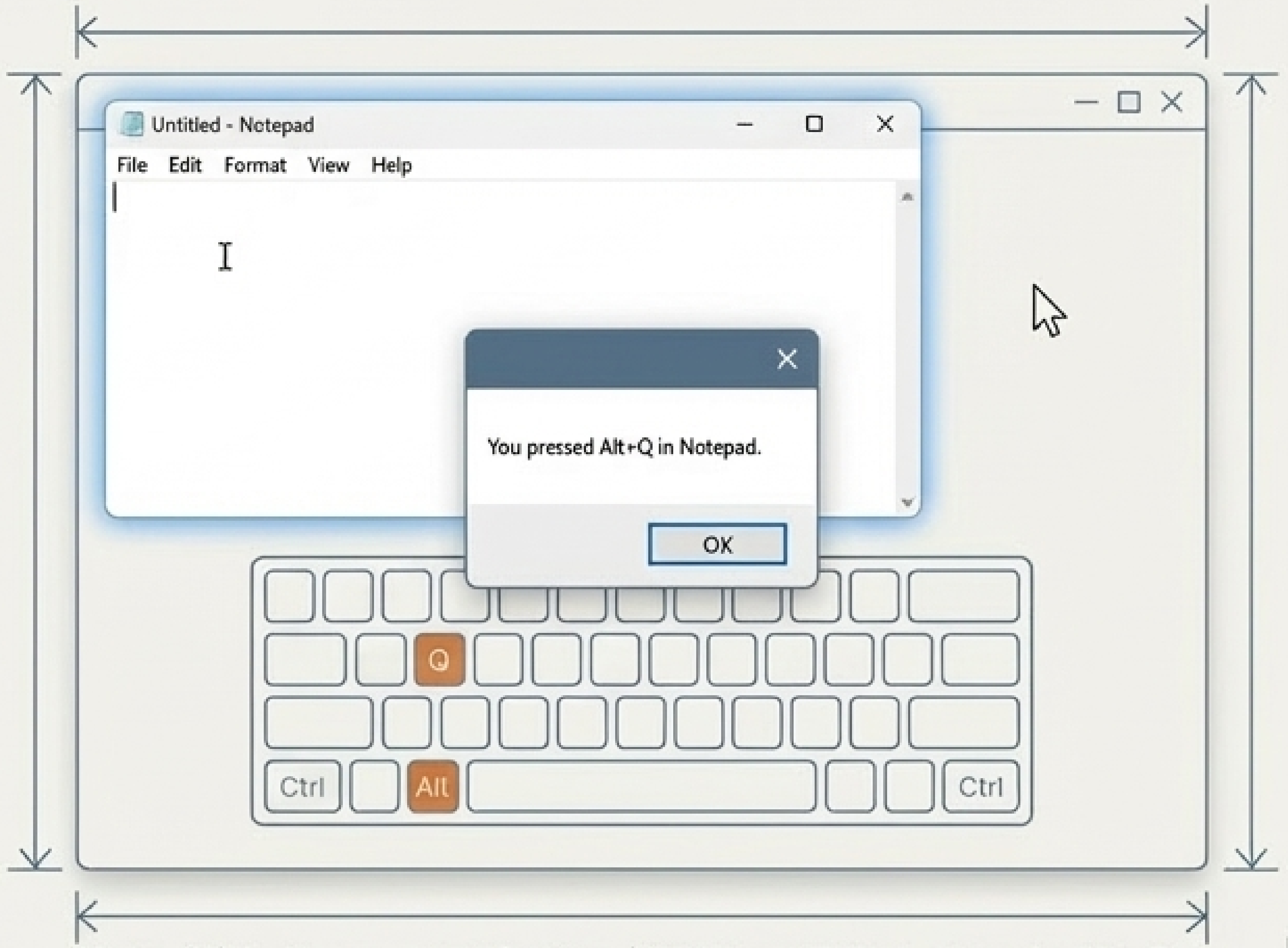
THE BLUEPRINT



```
#IfWinActive
"Untitled - Notepad"
!q::MsgBox("You
pressed Alt+Q in
```

```
#IfWinActive
!q::MsgBox("You
pressed Alt+Q in
any other window.")
```

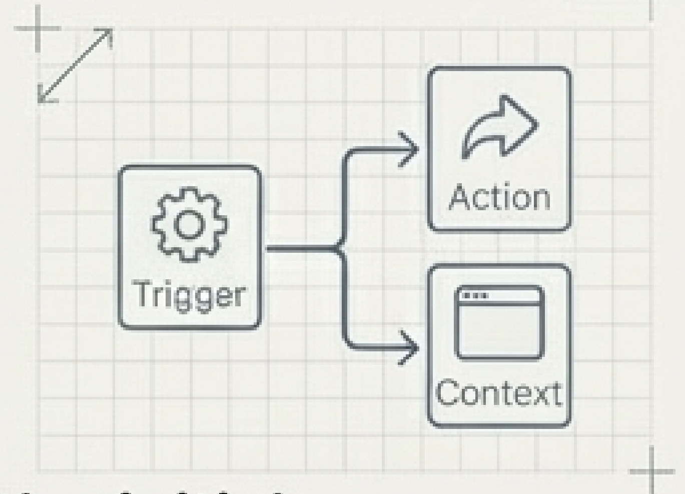
THE RESULT



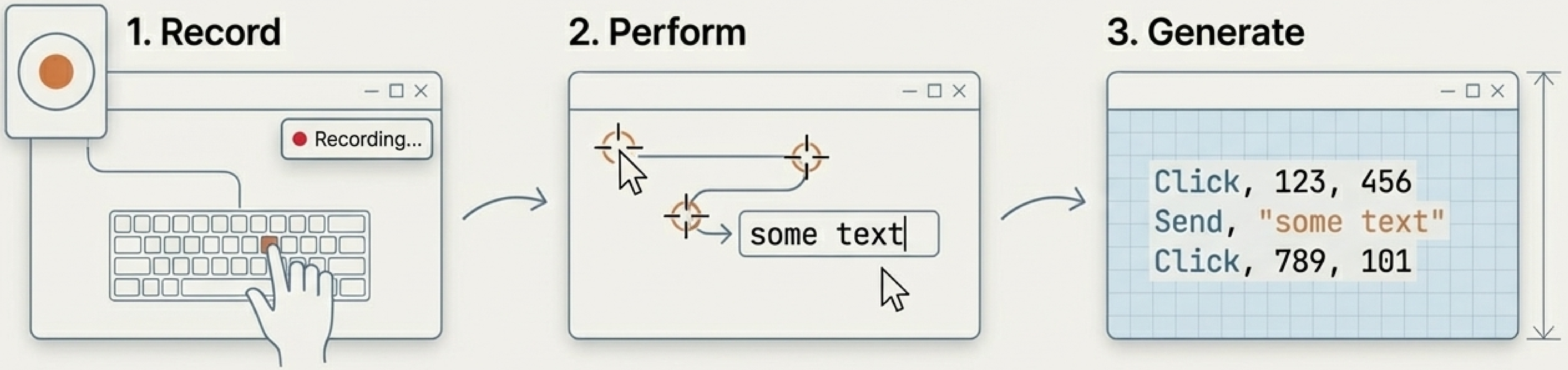
You pressed Alt+Q in Notepad.

Note: `!` is the symbol for the `Alt` key. This hotkey is Alt+Q.

The Accelerator: Using a Macro Recorder to Generate Complex Actions



For long, repetitive sequences of clicks and keystrokes, a macro recorder can write the initial script for you. Think of it as a code generator, not a replacement for understanding the code.



```
Click, 123, 456
Send, "some text"
Click, 789, 101
```

"The recorder codes the script for you. You may still want to adjust the script to do more advanced stuff, and you can still do that." - **Raeleus, Automation Expert**

The Modern Engine: Why AutoHotkey v2 is a Major Leap Forward

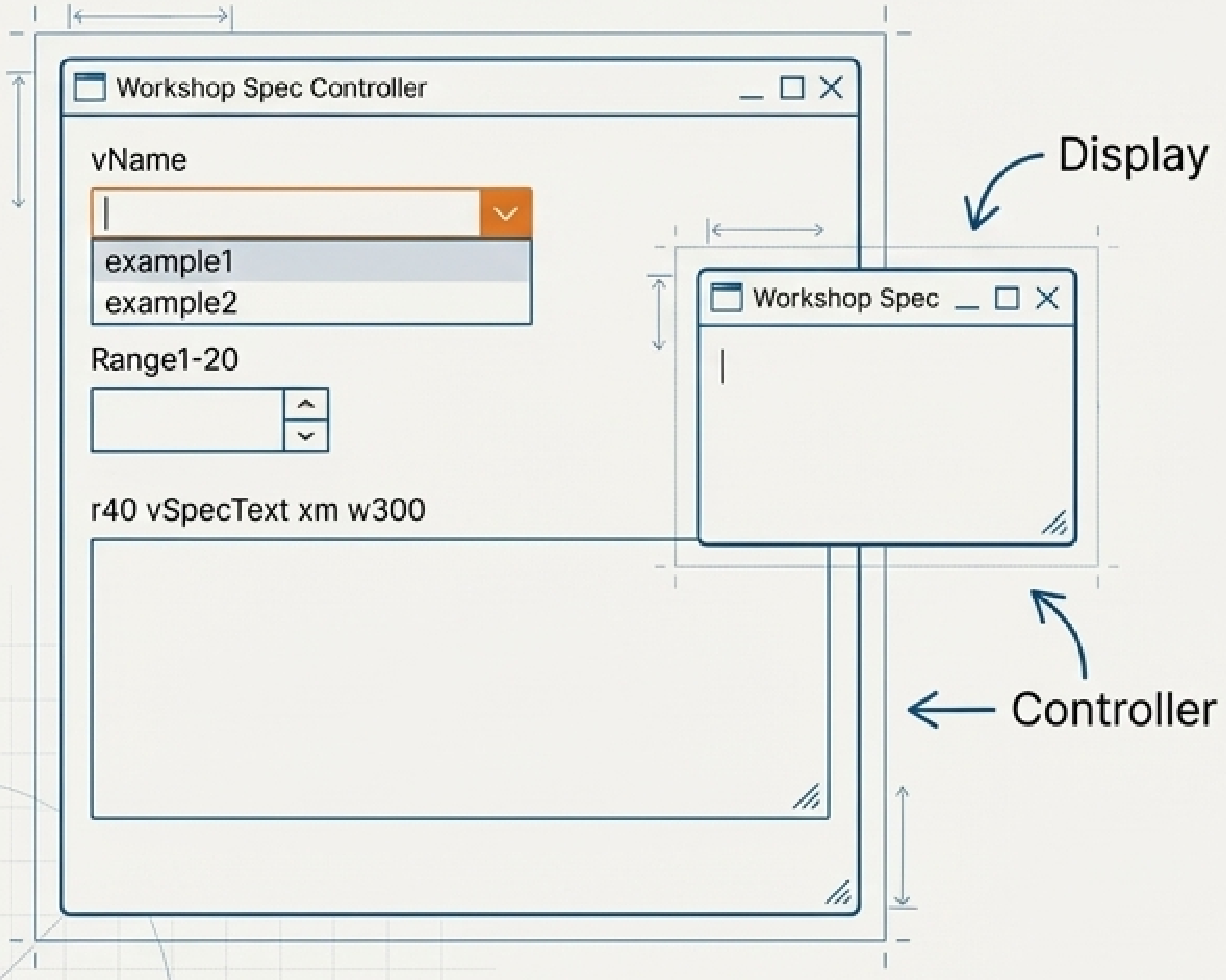
Key Idea: The release of AHK v2 was a fundamental redesign that fixed legacy issues and turned a quirky utility into a more consistent and powerful scripting language.

"AHK v1 was very, very bad... v2 got rid of implicit strings... In v2 everything's just a function... v2 just uses try/catch blocks."
- Hillel Wayne

Feature	AHK v1 (The Old Way)	AHK v2 (The Modern Way)
Syntax	Inconsistent mix of commands and functions. `StringUpper, name, tmp`	Everything is a function. `tmp := StrUpper(name) // comments`
Strings	Implicit strings, leading to confusion. `MsgBox Hello %name%`	All literal strings must be in quotes. `MsgBox("Hello " . name) // comments`
Error Handling	Relied on a global `ErrorLevel` variable, which could be unreliable.	Uses modern `Try / Catch` blocks for robust error handling.
Variables	Confusing rules for when to use % signs.	Consistent rules. Variables in expressions don't need % signs.

The Architect's Vision: Building Custom GUI Tools

A custom controller GUI built by developer Hillel Wayne for managing his formal methods workshops.



"Two GUIs, communicating state, two global hotkeys, less than 50 lines of code. AHK is **wonderful**." - Hillel Wayne

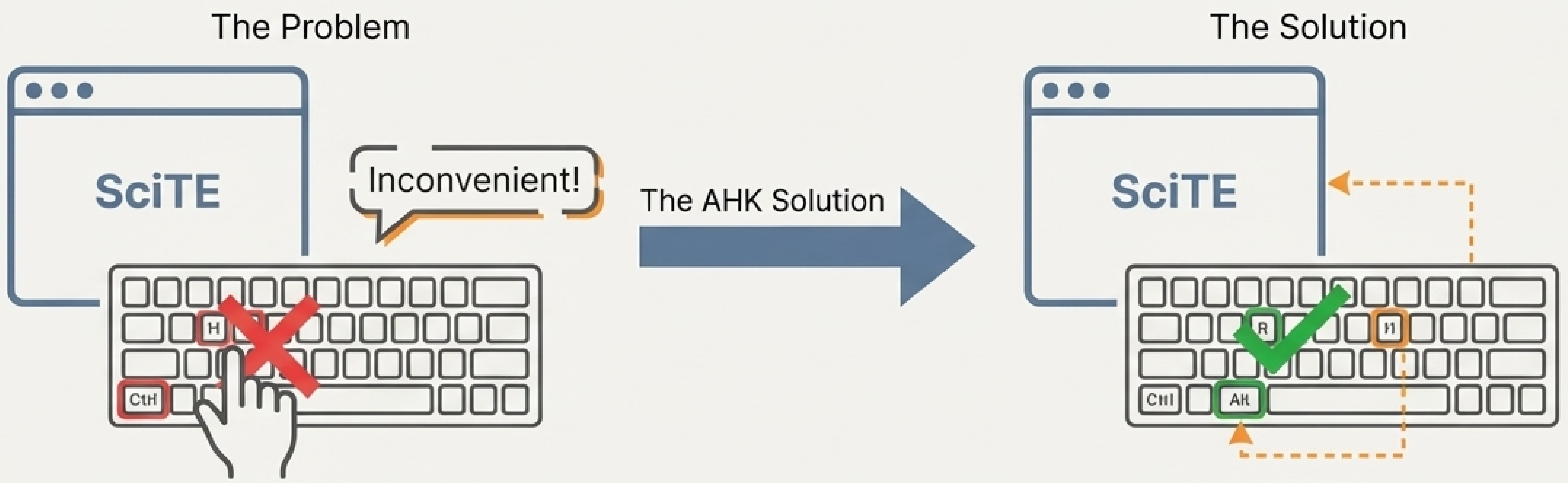
```
; Abridged example of the GUI creation code
Controller := Gui("AlwaysOnTop Resize", "Workshop Spec Controller")
Display := Gui("ToolWindow AlwaysOnTop +Owner" Controller.Hwnd, "Workshop Spec")

SpecNameCtrl := Controller.AddComboBox("vName", ["example1", "example2"])
SpecNumberCtrl := Controller.AddUpDown("Range1-20", 1)
cSpecText := Controller.AddEdit("r40 vSpecText xm w300")

Controller.Show("x2700 y500")
```

AutoHotkey isn't just for automation; it's for application development. You can build the exact tool you need.

The Integrator's Vision: Remapping and Refining Your Existing Tools



The Blueprint (The Code)

```
; Remaps Alt+R to send Ctrl+H, but only when SciTE is the active window
#IfWinActive "ahk_class SciTEWindow"
    !r::Send("^h")
#IfWinActive
```

You don't need to wait for a developer to add a feature. With AutoHotkey, you can modify the behavior of other programs to fit your needs perfectly. It acts as a universal customization layer for your entire Windows environment.

Your Blueprint: The Essential Starter Kit

The Right Editor

SciTE4AutoHotkey

It's an editor built specifically for AHK.

- Syntax highlighting and colour coding.
- 'Auto-assist' provides pop-up help for commands as you type.
- Integrated output panel shows you exactly where errors are in your script.
- Press `F1` on any command to instantly open its documentation.

Your Starter Template

```
; === Recommended Settings for All v2 Scripts ===
#Requires AutoHotkey v2.0
#SingleInstance Force
SetWorkingDir A_ScriptDir

; === Your Code Goes Below This Line ===
```

- `#Requires`: Ensures your script runs with the correct AHK version.
- `#SingleInstance Force`: Prevents duplicate copies of your script from running and automatically reloads the latest version.
- `SetWorkingDir`: Ensures the script looks for files in its own directory by default.

NotebookLM

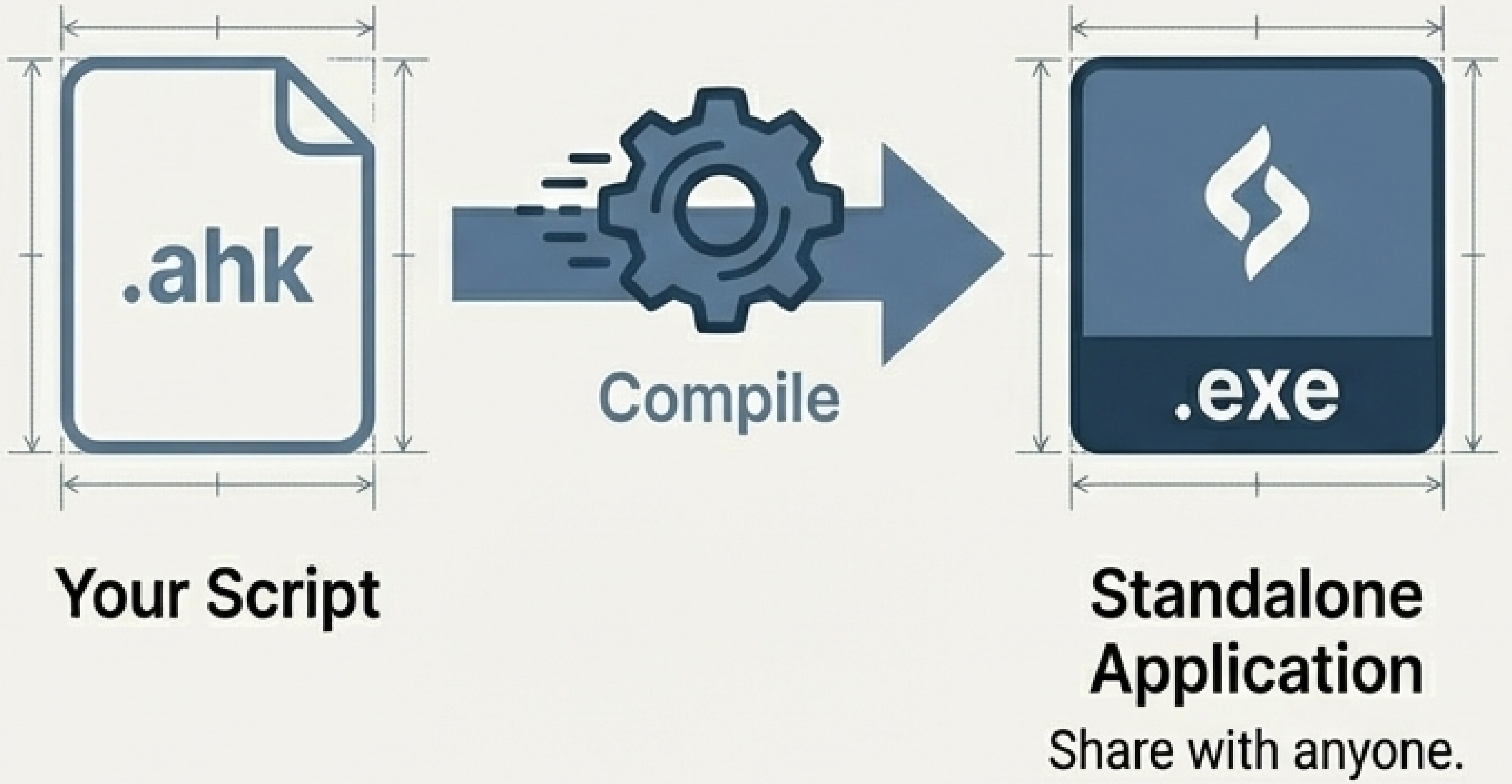
From Script to Application: Sharing Your Creations

The Goal:

Package your `.ahk` script into a single, portable `.exe` file that can run on any Windows computer, even one without AutoHotkey installed.

The Process is Simple

1. Finalize your script (MyScript.ahk).
2. Right-click the file in Windows Explorer.
3. Select 'Compile Script'.
4. An executable file (`MyScript.exe`) is created in the same folder.



The Library: Continue Your Journey

A curated list of the best resources for mastering AutoHotkey.

Official Documentation (The Foundation)

- **AutoHotkey v2 Docs**: The definitive source of truth.
- **Key Pages to Bookmark**: The complete Command List, Function List, and Key List.

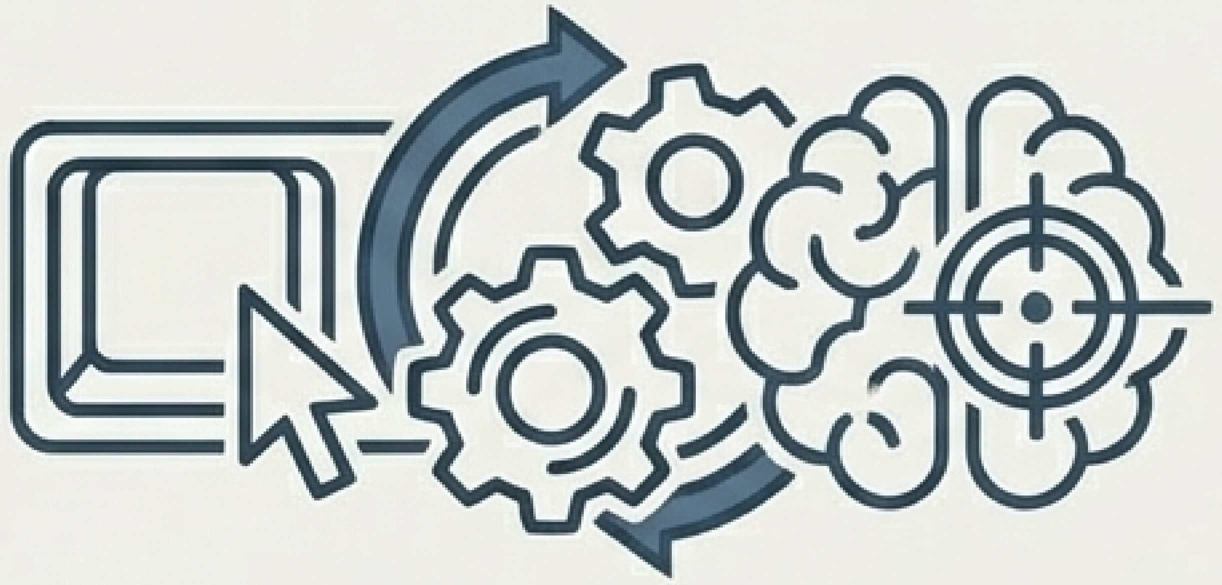
Community Wisdom (Live Support)

- **The Official AHK Forums**: A massive repository of solved problems and advanced scripts.
- **The AHK Discord Server**: Get real-time help and chat with other automation enthusiasts.

Guided Learning (Tutorials & Courses)

- **YouTube Channels**: the-Automator / AUTOHOTKEY Gurus for practical tutorials and deep dives.
- **Books & Blogs**: Jack Dunning's books for step-by-step projects and Hillel Wayne's blog for advanced concepts.

You are the Architect.



Learning AutoHotkey isn't about memorizing a list of commands. It's about acquiring a new way to interact with your computer.

You now have the fundamental building blocks—the Triggers, the Actions, and the Context—to solve your own problems.

Go build your solution.