EBook Gratis

APRENDIZAJE AutoHotkey

Free unaffiliated eBook created from
Stack Overflow contributors.

#autohotkey

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: autohotkey

It is an unofficial and free AutoHotkey ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official AutoHotkey.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con AutoHotkey

Observaciones

AutoHotkey es un lenguaje de scripting personalizado gratuito y de código abierto para Microsoft Windows, inicialmente destinado a proporcionar atajos de teclado o teclas de acceso rápido, una macro- creación rápida y la automatización del software que permite a los usuarios de la mayoría de los niveles informáticos automatizar tareas repetitivas en cualquier aplicación de Windows. AutoHotkey puede ampliar o modificar fácilmente las interfaces de usuario (por ejemplo, anulando los comandos predeterminados de las teclas de control de Windows con sus equivalentes de Emacs). La instalación de Autohotkey incluye su propio archivo de ayuda extensa con una versión basada en web siempre actualizada.

Puede escribir macros de mouse o teclado , reasignar teclas , crear teclas de acceso rápido , expandir abreviaturas, cambiar el contenido del portapapeles y hacer que los ejecutables ejecuten scripts de teclas de acceso rápido en computadoras sin AutoHotkey instalado.

Versiones

AutoHotkey 1.0. * - también conocido retroactivamente como AutoHotkey Basic, Classic, Vanilla, etc.

(El desarrollo se suspendió en 2011; último establo: 2009)

Versión	Fecha de lanzamiento
v1.0.48.05	2009-09-26
v1.0.97.02	2011-04-14

AutoHotkey 1.1. * - anteriormente conocido como AutoHotkey_L.

(Estable y recibe actualizaciones regularmente)

Versión	Fecha de lanzamiento
v1.1.24.00	2016-05-22
v1.1.24.01	2016-08-02

AutoHotkey 2.0-a *

(Todavía en fase alfa)

Versión	Fecha de lanzamiento
v2.0-a069	2015-10-24
v2.0-a070	2015-11-09
v2.0-a071	2015-12-25
v2.0-a072	2015-12-25
v2.0-a073	2016-02-05
v2.0-a074	2016-03-11
v2.0-a075	2016-06-03

Examples

Instalación o configuración

De la documentación del sitio Autohotkey

1. Ir a la página de inicio de AutoHotkey .
2. Haga clic en Descargar , una vez descargado ejecute el ejecutable.
3. Durante la instalación de AutoHotkey, se le pedirá que elija entre UNICODE o ANSI. En resumen, probablemente querría elegir UNICODE. Tiene soporte para letras y números que no están en inglés (caracteres).
4. Continúa hasta que veas un botón Instalar.
5. Una vez hecho, genial!

Utilizar como software portátil

1. Ve a la página de descargas de AutoHotkey.
2. Busque la sección Portátil, elija entre UNICODE 32, 64 o ANSI y descargue.
3. Al elegir la carpeta de destino, elija un dispositivo externo correcto o no.
4. Ahora puede optar por asociar archivos .ahk con Autohotkey.exe
5. Crea un archivo de texto plano y dale la extensión .ahk
6. Luego haga clic con el botón derecho en el archivo .ahk en el explorador y haga clic en Propiedades.
7. En las propiedades del archivo, haga clic en el botón Cambiar junto a la opción "Se abre con".
    - Después de hacer clic en Cambiar, se le dará una lista de programas para abrir el

archivo, seleccione el programa que desea usar y luego haga clic en Aceptar o Aplicar.
    - Si el programa que desea seleccionar no se encuentra en la lista, haga clic en el botón Examinar, busque el archivo ejecutable de Autohotkey (.exe) y haga clic en Aceptar para seleccionar ese programa.
8. Ahora los archivos .ahk se ejecutarán como si se hubiera instalado autohotkey, ¡excelente!

Si tiene instalado Chocolatey, ejecute el siguiente comando como usuario administrador.

    choco instalar autohotkey

Alternativamente, se puede construir desde el código fuente. Vea aquí para más detalles: https://github.com/Lexikos/AutoHotkey_L/

Hola Mundo

Mostrar un "¡Hola mundo!" en el cuadro de mensaje.

```
MsgBox, Hello World!
```

Mostrar un "¡Hola mundo!" en la descripción.

```
#Persistent
Tooltip, Hello World!
```

Mostrar un "¡Hola mundo!" Mensaje en la edición de la barra de herramientas.

```
#Persistent
TrayTip,, Hello World!
```

Imprime "Hola, Mundo" en Salida estándar (stdout).

```
FileAppend, % "Hello, World", *
```

Mostrar "Hello World" en una GUI

```
Gui, Add, Text,, Hello World!
Gui, Show, w200 h200
return

GuiClose:
ExitApp
```

Consigue un efecto similar a SplashTextOn

```
Gui, +AlwaysOnTop +Disabled -SysMenu +Owner  ; +Owner avoids a taskbar button.
Gui, Add, Text,, Some text to display.
Gui, Show, NoActivate, Title of Window  ; NoActivate avoids deactivating the currently active
window.
```

Cómo crear un script

Una vez que tenga AutoHotkey instalado, probablemente querrá que haga cosas. AutoHotkey no es magia, todos deseamos que lo fuera, pero no lo es. Así que tendremos que decirle qué hacer. Este proceso se llama "Scripting".

1. Haga clic derecho en su escritorio.
2. Encuentra "Nuevo" en el menú.
3. Haga clic en "AutoHotkey Script" dentro del menú "Nuevo".
4. Dale un nuevo nombre al script. Nota: Debe terminar con una extensión .ahk. Ex. MyScript.ahk
5. Encuentra el archivo recién creado en tu escritorio y haz clic derecho.
6. Haga clic en "Editar Script".
7. Una ventana debería haber aparecido, probablemente Bloc de notas. Si es así, ¡ÉXITO!

Así que ahora que ha creado un script, necesitamos agregar cosas al archivo. Para obtener una lista de todos los comandos, funciones y variables incorporados, consulte la sección 5. Aquí hay una secuencia de comandos muy básica que contiene una tecla de acceso directo que escribe texto con el comando Enviar cuando se presiona la tecla de acceso rápido.

```
^j::
   Send, My First Script
Return
```

Vamos a obtener más en profundidad más adelante. Hasta entonces, aquí hay una explicación del código anterior.

- La primera línea. `^j::` es la tecla de acceso directo. `^` significa `CTRL` , `j` es la letra j. Cualquier cosa a la izquierda de `::` son las teclas que necesita presionar.
- La segunda línea. `Send, My First Script` es cómo `SEND` pulsaciones de teclas. `SEND` es el comando, se escribirá cualquier cosa después de la coma (,).
- La tercera línea. `Return` El regreso se convertirá en tu mejor amigo. Literalmente, DETIENE que el código no vaya más lejos, a las líneas de abajo. Esto evitará muchos problemas cuando empieces a tener muchas cosas en tus scripts.

8. Guarda el archivo.
9. Haga doble clic en el archivo / icono en el escritorio para ejecutarlo. Abra el bloc de notas o (cualquier cosa que pueda escribir) y presione Ctrl y J.
10. ¡Hip hip hurra! Su primer guión está hecho. Obtenga algunos bocadillos de recompensa y luego vuelva a leer el resto de este tutorial.

Lea Empezando con AutoHotkey en línea:
https://riptutorial.com/es/autohotkey/topic/3532/empezando-con-autohotkey

Capítulo 2: Abra un archivo en un script

Introducción

Diferentes formas de abrir un archivo para trabajar en un script.

Examples

Abra un archivo a través del Explorador de Windows

Dentro de la secuencia de comandos, use la primera línea para almacenar la primera variable (en este ejemplo, `%1%`) con un nombre para tratar. Ejemplo: `OpenWithFile = %1%`

Una vez que abra un archivo con este script a través de Windows (haga clic con el botón derecho en cualquier archivo en MS Windows y elija "Abrir con ...", luego seleccione la versión **compilada** del script, como script.exe). El nombre del archivo elegido será almacenado en esta variable y, por lo tanto, el script podrá trabajar con ella. Ejemplo:

```
OpenWithFile = %1%
if OpenWithFile !=
{
FileRead, content, %OpenWithFile%
msgbox %content%
return
}
```

Abrir un archivo a través del cuadro de diálogo Seleccionar archivo

El siguiente ejemplo crea una Gui con un solo botón que trae el cuadro de diálogo Seleccionar archivo.

```
Gui, Loader: New
Gui, Loader: Add, Button, Default Center w220 vLOAD, LOAD
Gui, Loader: Show, AutoSize Center, Loader

return

LoaderButtonLOAD:
FileSelectFile, LoadedFile, , , ,

if ErrorLevel=1
{
return
}

else
{
    FileRead, content, %LoadedFile%
    msgbox %content%
}
```

```
return
```

Abra un archivo a través de Windows Arrastrar y soltar

Este ejemplo crea un nuevo evento Gui vacío para arrastrar y soltar:

```
Gui, Dropper: New
Gui, Dropper: Font, s10 w700
Gui, Dropper: Add, Text, y80 vText1, Drag the files here
Gui, Dropper: Show, w200 h200 Center, Dropper

return

DropperGuiDropFiles:
DroppedFile:=A_GuiEvent

    FileRead, content, %DroppedFile%
    msgbox %content%

return
```

Lea Abra un archivo en un script en línea: https://riptutorial.com/es/autohotkey/topic/9070/abra-un-archivo-en-un-script

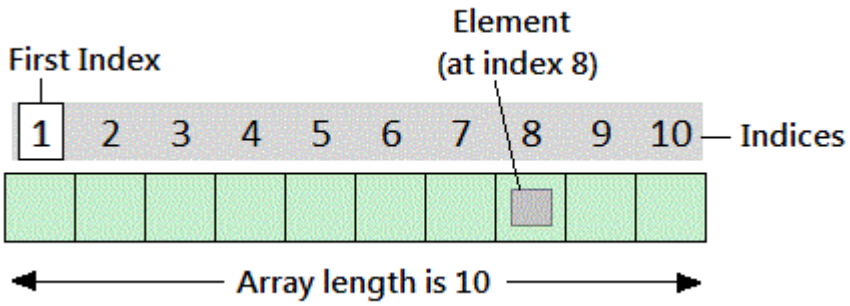
Capítulo 3: Arrays

Examples

Creando e inicializando arreglos simples

Introducción

Una *matriz* es un objeto contenedor que contiene una serie de valores. En la siguiente imagen puede ver una matriz con tamaño 10, el primer elemento indexado 1 y el último elemento 10.



Autohotkey ofrece algunas formas de definir y crear matrices.

- Array: = []
- Array: = Array ()

Creación e inicialización de matrices con N número de elementos

```
Array := [Item1, Item2, ..., ItemN]
Array := Array(Item1, Item2, ..., ItemN)
```

En Autohotkey, es posible tener matrices sin elementos:

```
Array := [] ; works fine.
```

Y luego se le pueden asignar elementos:

```
Array[0] := 1
```

El tamaño de la matriz se puede determinar utilizando un método llamado `length` :

```
msgbox % array.length()  ; shows 1 in this case.
```

Si la matriz no está vacía, **MinIndex** y **MaxIndex / Length** devuelven el índice más bajo y más alto actualmente en uso en la matriz. Como el índice más bajo es casi siempre 1, MaxIndex generalmente devuelve el número de elementos. Sin embargo, si no hay claves enteras, MaxIndex devuelve una cadena vacía mientras que Length devuelve 0.

Creando e inicializando arrays multidimensionales.

Puede crear una matriz multidimensional de la siguiente manera:

```
Array[1, 2] := 3
```

Puede crear e inicializar al mismo tiempo, y las matrices internas no tienen que ser de la misma longitud.

```
Array := [[4,5,6],7,8]
```

Las matrices de este tipo también se denominan matrices de matrices.

Llenando una matriz

```
; Assign an item:
Array[Index] := Value

; Insert one or more items at a given index:
Array.InsertAt(Index, Value, Value2, ...)

; Append one or more items:
Array.Push(Value, Value2, ...)
```

El valor de un índice para un elemento de matriz también puede ser un entero negativo (-1, 0, 1, 2, 3, 4, ...)

Eliminando elementos de una matriz

```
; Remove an item:
RemovedValue := Array.RemoveAt(Index)

; Remove the last item:
RemovedValue := Array.Pop()
```

Añadiendo métodos personalizados

anulando la función Array ()

AutoHotkey es un lenguaje de programación basado en prototipos , lo que significa que puede anular cualquier función / objeto incorporado en cualquier momento. Este ejemplo demuestra la función Array () que prevalece para agregar métodos dentro de un objeto de clase personalizado.

```
; Overrides Array()
Array(Args*) {
    Args.Base := _Array
    Return Args
}

; Custom Class Object with Methods
Class _Array {

    ; Reverses the order of the array.
    Reverse() {
        Reversed := []
        Loop % This.MaxIndex()
            Reversed.Push(This.Pop())
        Return Reversed
    }

    ; Sums all Integers in Array
    Sum(Sum=0) {
        For Each, Value In This
            Sum += Value Is Integer ? Value : 0
        Return Sum
    }
}

; Arr == ["Hello, World!", 4, 3, 2, 1]
Arr := [1, 2, 3, 4, "Hello, World!"].Reverse()

; SumOfArray == 10
SumOfArray := Arr.Sum()
```

Lea Arrays en línea: https://riptutorial.com/es/autohotkey/topic/4468/arrays

Capítulo 4: Campo de entrada

Introducción

Para obtener la entrada de un usuario y almacenarla en una variable, puede usar el comando InputBox. El script no continuará ejecutando comandos hasta que el usuario presione 'Aceptar' o 'Cancelar'.

'Aceptar' cerrará la ventana y guardará la entrada del usuario 'Cancelar' cerrará la ventana, descartando la entrada del usuario

Parámetros

InputBox, OutputVar [, Título, Solicitud, OCULTAR, Ancho, Altura, X, Y, Tiempo de espera, Predeterminado]	Qué significa cada opción
OutputVar	La variable de entrada del usuario se guardará en
Título	El nombre del cuadro de entrada.
Rápido	Texto dentro del cuadro de entrada
ESCONDER	Muestra la entrada del usuario como asteriscos para ocultar la entrada; escriba HIDE para habilitar
Anchura	El ancho del cuadro de entrada
Altura	La altura del cuadro de entrada.
X	La cantidad de píxeles del borde izquierdo de la pantalla que la esquina superior izquierda del cuadro de entrada será
Y	La cantidad de píxeles desde el borde superior de la pantalla que la esquina superior izquierda del cuadro de entrada será
Se acabó el tiempo	Cierra automáticamente el cuadro de entrada y guarda la entrada del usuario después de este tiempo en milisegundos.
Defecto	El texto que aparecerá en el campo editable del cuadro de entrada cuando se abra

Observaciones

Un cuadro de entrada es un elemento de la GUI, por lo que se tratará como un elemento de la GUI.

Una lista de niveles de error para este comando:

Nivel de error	Lo que significa
0	El usuario presiona el botón 'OK'
1	El usuario presiona el botón 'Cancelar'
2	El cuadro de entrada expiró

Puede encontrar la página de este comando en la documentación de AutoHotkey aquí: https://autohotkey.com/docs/commands/InputBox.htm

Examples

Ejemplo de uso básico

```
InputBox, userinput
```

Esto almacenará lo que el usuario escribe en el cuadro de entrada en la variable llamada *userinput*

Contraseñas

```
InputBox, password, Enter your Password,, HIDE,, 100

Loop, {
  if (errorlevel = 1)
return

  if (password = "password") {
MsgBox, The password is correct.
    return
  } else if (password != "password") {
MsgBox, The password is incorrect.
InputBox, password, Enter your Password,, HIDE,, 100
  }
}
```

Esto comprobará si el usuario ha escrito "contraseña" en el cuadro de entrada. Si el usuario escribe el valor correcto, dirá "La contraseña es correcta". y cierre el cuadro de entrada. Si el usuario escribe el valor incorrecto, dirá "La contraseña es incorrecta". y vuelva a abrir el cuadro de entrada. Si el nivel de error es 1 (el usuario presionó cancelar), terminará la secuencia de comandos.

Lea Campo de entrada en línea: https://riptutorial.com/es/autohotkey/topic/9917/campo-de-entrada

Capítulo 5: Hola Mundo

Examples

Hola ejemplos del mundo

Mostrar un "¡Hola mundo!" en el cuadro de mensaje.

```
MsgBox, Hello World!
```

Mostrar "¡Hola mundo!" almacenado en variable mytring en el cuadro de mensaje.

```
MyString := "Hello World!"
MsgBox, %MyString%
```

Mostrar un "¡Hola mundo!" en la descripción.

```
#Persistent
Tooltip, Hello World!
```

Mostrar un "¡Hola mundo!" Mensaje en la edición de la barra de herramientas.

```
#Persistent
TrayTip,, Hello World!
```

Mostrar "¡Hola mundo!" en una ventana GUI.

```
Gui, Add, Text,, Hello World!
Gui, Add, Edit,, Hello World!
Gui, Show
return

GuiClose() {
    ExitApp
}
```

Impresiones "¡Hola mundo!" a salida estándar (stdout).

```
FileAppend, % "Hello, World!", *
```

Simular la escritura "Hola, mundo!" al presionar enter

```
Enter::Send, Hello World{!}
```

Crea un nuevo archivo llamado HelloWorld.txt

```
FileAppend,, HelloWorld.txt
```

Al escribir la palabra "Hola" seguida de un espacio o ingresar, se reemplazará por "Hola mundo"

```
::Hello::Hello World
```

Lea Hola Mundo en línea: https://riptutorial.com/es/autohotkey/topic/3547/hola-mundo

Capítulo 6: Scripts de teclas de acceso rápido

Sintaxis

- atajos de teclado::
- ::abreviatura::
- Regreso

Parámetros

Atajos de teclado	Detalles
^	Tecla Ctrl
!	tecla Alt
+	Tecla Shift
#	Tecla de Windows
{entrar}	enviar tecla enter
{lengüeta}	enviar tecla de tabulación
*	comodín, cualquier tecla se puede presionar hacia abajo
~	La función nativa de la tecla no será bloqueada.
<símbolo	especifica la tecla de la izquierda (<+ es el turno de la izquierda)
> símbolo	especifica la clave correcta

Examples

Tecla de acceso directo

Para hacer una tecla de acceso rápido que envíe la secuencia de teclas 'Hola mundo' desde presionar `Ctrl` + `J` en la ventana activa (se puede demostrar en el bloc de notas, por ejemplo)

```
^j::
    Send, Hello World
Return
```

Cadena caliente

Para hacer un script para reemplazar una frase, use la `::abbreviation::` hotstring syntax. Por `btw` reemplazará `by the way` cada vez que ingrese por `btw` y luego la tecla de espacio.

```
::btw::by the way
```

Si quisiera hacer un script de inicio de sesión para que el inicio de sesión sea más rápido, podría hacer un script como este (el archivo no está cifrado, por lo que cualquier persona con acceso al archivo podrá ver la información en su script).

```
::lmi::user{tab}password{enter}
```

Pulsación múltiple

Para ejecutar un script cuando se presionan varias teclas, use las teclas `&` entre ellas.

```
Numpad0 & Numpad1::
    MsgBox You pressed 0 and 1
return
```

Teclas de acceso rápido sensibles al contexto y Hotstrings

Para crear una tecla de acceso rápido o cadena de activación que solo se activa cuando ciertas ventanas están activas o existen, puede poner una o varias de las siguientes *directivas* antes de la definición de la tecla de acceso rápido:

```
#IfWinActive [, WinTitle, WinText]
#IfWinExist [, WinTitle, WinText]
#IfWinNotActive [, WinTitle, WinText]
#IfWinNotExist [, WinTitle, WinText]
```

Ejemplo: desea `stackoverflow.com` que se enviará cada vez que escriba `so` (y un espacio en blanco después de eso) en Google Chrome, pero no hace caso de la cadena de acceso rápido en cualquier otra ventana.

```
#IfWinActive, ahk_class Chrome_WidgetWin_1
::so::stackoverflow.com
```

Al usar `#If [, Expression ]` , puede hacer un disparador de tecla de `#If [, Expression ]` rápido solo cuando una expresión arbitraria es verdadera, por ejemplo:

```
#If A_Hour < 9
F1::
    MsgBox, It is too early to ask for help!
return
```

Reasignar teclas

El siguiente ejemplo vuelve a asignar la clave `z` a `y` y viceversa, por ejemplo, si desea trabajar con el diseño QWERTY en un teclado QWERTZ.

```
z::y
y::z
```

Teclas de acceso rápido conmutables

El siguiente script ingresa cadenas predefinidas en las teclas de acceso rápido si el bloqueo de desplazamiento está activo. Esto puede ser útil si a menudo pegas un número de cadenas que se repiten. Tecla de acceso directo incluida para actualizar el script (por ejemplo, si necesita editar cadenas que se pueden pegar).

```
; refresh script hotkey
Numpad9::
    GetKeyState, state, ScrollLock, T
    if ( state = "D" )
        Reload
Return

Numpad1::
    GetKeyState, state, ScrollLock, T
    if ( state = "D" )
        Send,         Hello
Return

Numpad2::
    GetKeyState, state, ScrollLock, T
    if ( state = "D" )
        Send,         World
Return
;...
```

Lea Scripts de teclas de acceso rápido en línea:
https://riptutorial.com/es/autohotkey/topic/3853/scripts-de-teclas-de-acceso-rapido

Capítulo 7: Usar funciones en lugar de etiquetas.

Observaciones

AutoHotkey solía confiar mucho en las etiquetas hasta la versión 1.1.20. Su confianza en las etiquetas tenía desventajas muy serias. La principal es que las etiquetas generalmente se ejecutan en el ámbito global, lo que significa que cualquier variable definida dentro de una etiqueta estará disponible globalmente. Esto suena bien hasta que te das cuenta de que, por ejemplo, no puedes usar bibliotecas de otras personas sin asegurarte de que sus variables no interfieran con las tuyas.
Trabajar en el ámbito global cuando no es necesario es simplemente una mala práctica.

Así que aquí es donde entran las funciones. A partir de la versión 1.1.20, cada comando de AutoHotkey que acepta un nombre de etiqueta como parámetro, ahora alternativamente acepta un nombre de función.

Examples

Ejemplo muy básico que demuestra el uso de funciones en SetTimer.

```
;Sends the keystroke for the letter "a" every 3 seconds.
#Persistent
SetTimer, SendLetterA, 3000
return

SendLetterA() {
    Send, a
}
```

Uso avanzado de SetTimer: llamar a la misma función con diferentes parámetros

Este es un ejemplo de algo que hubiera sido imposible con las etiquetas. Si ejecuta la misma etiqueta varias veces al mismo tiempo y dependen de las variables que se están definiendo dentro de ellas, es muy probable que interfieran y causen un comportamiento inesperado.

Aquí está cómo hacerlo con funciones:

```
; This script will switch between showing "Hello 1" and "Hello 2"

#Persistent
DisplayMessage_Hello1 := Func("DisplayMessage").bind("Hello 1")
SetTimer, %DisplayMessage_Hello1%, 2000

Sleep, 1000
```

```
DisplayMessage_Hello2 := Func("DisplayMessage").bind("Hello 2")
SetTimer, %DisplayMessage_Hello2%, 2000

DisplayMessage(messageToDisplay) {
    TrayTip ; remove other traytips
    TrayTip, Message to display:, %messageToDisplay%
}
```

Aquí es **cómo no hacerlo** (con etiquetas):

```
;This script will never display the message "Hello 1". It will always show "Hello 2".

#Persistent
messageToDisplay := "Hello 1"
SetTimer, DisplayMessage, 2000

Sleep, 1000

messageToDisplay := "Hello 2"
SetTimer, DisplayMessage, 2000

DisplayMessage:
    TrayTip ; remove other traytips
    TrayTip, Message to display:, %messageToDisplay%
Return
```

Gui con funciones en lugar de etiquetas.

Ejemplo que muestra cómo crear Guis usando funciones en lugar de etiquetas.

```
Gui, Add, Button, gCtrlEvent vButton1, Button 1
Gui, Add, Button, gCtrlEvent vButton2, Button 2
Gui, Add, Button, gGoButton, Go Button
Gui, Add, Edit, vEditField, Example text
Gui, Show,, Functions instead of labels

CtrlEvent(CtrlHwnd:=0, GuiEvent:="", EventInfo:="", ErrLvl:="") {
    GuiControlGet, controlName, Name, %CtrlHwnd%
    MsgBox, %controlName% has been clicked!
}
GoButton(CtrlHwnd:=0, GuiEvent:="", EventInfo:="", ErrLvl:="") {
    GuiControlGet, EditField
    MsgBox, Go has been clicked! The content of the edit field is "%EditField%"!
}

GuiClose(hWnd) {
    WinGetTitle, windowTitle, ahk_id %hWnd%
    MsgBox, The Gui with title "%windowTitle%" has been closed!
    ExitApp
}
```

Teclas rápidas con funciones en lugar de etiquetas.

Ejemplos de uso de funciones con teclas de acceso rápido:

```
Hotkey, a, MyFunction ; Calls MyFunction() when a is pressed

MyFunction() {
    MsgBox You pressed %A_ThisHotkey%.
}
```

O:

```
a::MyFunction()

MyFunction() {
    MsgBox You pressed %A_ThisHotkey%.
}
```

Bandeja de acciones de menú con funciones.

```
#Persistent

Menu, Tray, NoStandard ; remove default tray menu entries
Menu, Tray, Add, MyDefaultAction, OnDefaultTrayAction ; add a new tray menu entry
Menu, Tray, Add, Exit, Exit ; add another tray menu entry
Menu, Tray, Default, MyDefaultAction ;When doubleclicking the tray icon, run the tray menu
entry called "MyDefaultAction".


OnDefaultTrayAction() {
    MsgBox, You double clicked the tray icon of this script or you clicked the MyDefaultAction
entry!
}

Exit() {
    MsgBox, You clicked the Exit entry! The script will close itself now.
    ExitApp
}
```

Ejemplo general de funciones vs etiquetas

Demostraré el uso básico del uso de funciones frente al uso de etiquetas + gosub.
En este ejemplo, implementaremos una funcionalidad simple para agregar dos números y almacenarlos en una variable.

Con funciones:

```
c := Add(3, 2) ; function call
MsgBox, Result: %c%

Add(a, b) { ; This is a function. Put it wherever you want, it doesn't matter.
    ; the a and b inside of this function are set by the function call above
    Return a+b ; the function will return the result of the expression "a+b"
}
```

Con etiquetas (por favor no hagas eso):

```
a := 3
b := 2
GoSub, Add ; execute the label "Add" then jump back to the next line here
MsgBox, Result: %c%
Return ; without this, the label would be executed again for no reason.

Add: ; This is a label. Please put them at the bottom of your script and use "Return" in a
line above.
    c := a+b
Return
```

OnClipboardCambiar con fucntion / label

El siguiente código está tomado de la documentación oficial de AutoHotkey:

Implementación de la función:

```
#Persistent
OnClipboardChange("ClipChanged")
return

ClipChanged(Type) {
    ToolTip Clipboard data type: %Type%
    Sleep 1000
    ToolTip  ; Turn off the tip.
}
```

Implementación de la etiqueta:

```
#Persistent
return

OnClipboardChange:
ToolTip Clipboard data type: %A_EventInfo%
Sleep 1000
ToolTip  ; Turn off the tip.
return
```

Ejemplo de Gui más complicado con múltiples vistas de lista que utilizan la misma función de devolución de llamada de eventos

Esta secuencia de comandos muestra cómo recibir eventos complicados de GUI de diferentes controles en la misma función de devolución de llamada de eventos. Usaremos dos controles ListView para eso.
Ahora, cada vez que se detecta una acción en uno de esos controles ListView, queremos una descripción precisa de lo que sucedió y tenemos que iniciar sesión en un control de edición en la misma GUI.

```
Gui, Add, ListView, gListCtrlEvent vMyFirstListView AltSubmit -ReadOnly R10 w310,
ColumnTitle1|ColumnTitle2|ColumnTitle3
Gui, Add, ListView, gListCtrlEvent vMySecondListView AltSubmit -ReadOnly R10 w310,
ColumnTitle1|ColumnTitle2|ColumnTitle3
Gui, Add, Text, w310, Action Log
```

```
Gui, Add, Edit, vLog R7 w310,
Gui, Show,, Functions instead of labels

; Create example entries for the first ListView
Gui, ListView, MyFirstListView
Loop, 10 {
    LV_Add("", "Column-1 | Row-" A_Index ,  "Column-2 | Row-" A_Index,  "Column-3 | Row-"
A_Index)
}
LV_ModifyCol()

; Create example entries for the second ListView
Gui, ListView, MySecondListView
Loop, 10 {
    LV_Add("", "Column-1 | Row-" A_Index ,  "Column-2 | Row-" A_Index,  "Column-3 | Row-"
A_Index)
}
LV_ModifyCol()


ListCtrlEvent(ctrlHwnd:=0, guiEvent:="", eventInfo:="", errLvl:="") {
    GuiControlGet, ctrlName, Name, %CtrlHwnd%
    whatHappened := "Action detected!`n"
    whatHappened .= "Control handle: " ctrlHwnd "`n"
    whatHappened .= "Control name: " ctrlName "`n"

    If (guiEvent = "DoubleClick") {
        whatHappened .= "`nThe user has double-clicked within the control."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "R") {
        whatHappened .= "`nThe user has double-right-clicked within the control."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "ColClick") {
        whatHappened .= "`nThe user has clicked a column header."
        whatHappened .= "`n> Column number: " eventInfo
    } Else If (guiEvent = "D") {
        whatHappened .= "`nThe user has attempted to start dragging a row or icon."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "d") {
        whatHappened .= "`nThe user has attempted to start right-click-dragging a row or
icon."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "e") {
        whatHappened .= "`nThe user has finished editing the first field of a row."
        whatHappened .= "`n> Row number: " eventInfo
    } Else If (guiEvent = "Normal") {
        whatHappened .= "`nThe user has left-clicked a row."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "RightClick") {
        whatHappened .= "`nThe user has right-clicked a row."
        whatHappened .= "`n> Focused row number: " eventInfo
    } Else If (guiEvent = "A") {
        whatHappened .= "`nA row has been activated."
        whatHappened .= "`n> Row number: " eventInfo
    } Else If (guiEvent = "C") {
        whatHappened .= "`nThe ListView has released mouse capture."
    } Else If (guiEvent = "E") {
        whatHappened .= "`nThe user has begun editing the first field of a row."
        whatHappened .= "`n> Row number: " eventInfo
    } Else If (guiEvent = "F") {
        whatHappened .= "`nThe ListView has received keyboard focus."
```

```
    } Else If (guiEvent = "f") {
        whatHappened .= "`nThe ListView has lost keyboard focus."
    } Else If (guiEvent = "I") {
        whatHappened .= "`nItem changed. (A row has changed by becoming selected/deselected,
checked/unchecked, etc.)"
        whatHappened .= "`n> Row number: " eventInfo
    } Else If (guiEvent = "K") {
        whatHappened .= "`nThe user has pressed a key while the ListView has focus."
        whatHappened .= "`n> Key pressed: " GetKeyName(Format("vk{:x}", eventInfo))
    } Else If (guiEvent = "M") {
        whatHappened .= "`nItem changed. (A row has changed by becoming selected/deselected,
checked/unchecked, etc.)"
        whatHappened .= "`n> Row number: " eventInfo
    } Else If (guiEvent = "S") {
        whatHappened .= "`nMarquee. The user has started to drag a selection-rectangle around
a group of rows or icons."
    } Else If (guiEvent = "s") {
        whatHappened .= "`nThe user has finished scrolling the ListView."
    }
    GuiControlGet, Log
    GuiControl,, Log, % whatHappened "`n----------------------`n" Log
}

GuiClose(hWnd) {
    WinGetTitle, windowTitle, ahk_id %hWnd%
    MsgBox, The Gui with title "%windowTitle%" is going to be closed! This script will exit
afterwards!
    ExitApp
}
```

Lea Usar funciones en lugar de etiquetas. en línea:
https://riptutorial.com/es/autohotkey/topic/4062/usar-funciones-en-lugar-de-etiquetas-

Capítulo 8: Variables y funciones incorporadas

Observaciones

AutoHotkey viene con muchas funciones y variables integradas que pueden usarse en cualquier lugar dentro de un script.
Para una lista completa que incluye explicaciones, vea:

- Lista de variables incorporadas
- Lista de funciones incorporadas

Examples

Determinación del tiempo de inactividad del usuario

```
if(A_TimeIdlePhysical > 60000) { ; 60,000 milliseconds
    WinClose, ahk_class Chrome_WidgetWin_1
    MsgBox, Google Chrome was closed due to user inactivity.
}
```

Esta comprobación podría realizarse periódicamente, por ejemplo, utilizando `SetTimer` .

Insertar automáticamente el nombre del día de la semana actual

Este ejemplo inserta / envía el día completo del nombre completo de la semana (por ejemplo, *domingo*) cada vez que se `presiona Ctrl` + `Alt` + `D` :

```
^!d::Send, %A_DDDD%
```

Extraer partes de cadena utilizando RegEx

```
myDebt := 9000
index  := RegExMatch("You owe me $42", "\$(\d+)", dollars)
if(index > 0) { ; indices are usually 1-based in AHK
    myDebt += dollars1
    MsgBox, Current debt: %myDebt%
}
```

Resultado:

> Deuda actual: 9042

Recortar una cuerda

```
myString := "  hello, Trim()! "
trimmed  := Trim(myString)
FileAppend, % trimmed "`n", TrimmedStrings.txt
```

Tenga en cuenta que `Trim()` no manipulará la cadena original, sino que devolverá una nueva que se debe almacenar o generar en algún lugar.

Lea Variables y funciones incorporadas en línea:
https://riptutorial.com/es/autohotkey/topic/4469/variables-y-funciones-incorporadas

Creditos

S. No	Capítulos	Contributors
1	Empezando con AutoHotkey	blackholyman, Community, depperm, Forivin, Joe DF, Shyam Sundar Shankar, tlm, Vijay
2	Abra un archivo en un script	freestock.tk
3	Arrays	blackholyman, errorseven
4	Campo de entrada	Meh
5	Hola Mundo	errorseven, Forivin, Goerman, mikew, vasili111
6	Scripts de teclas de acceso rápido	depperm, MCL, user5226582
7	Usar funciones en lugar de etiquetas.	Forivin
8	Variables y funciones incorporadas	MCL