# Tastenkombinationen einrichten mit VBA

Diese Dokumentation enthält kein Inhaltsverzeichnis. Schalten Sie in Ihrem PDF-Reader die Lesezeichen ein.

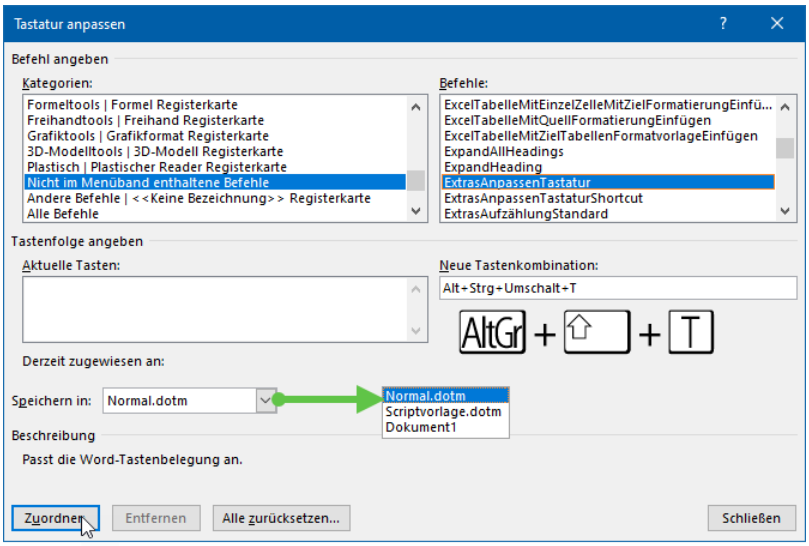
Word lässt sich fast vollständig per Tastatur bedienen. Die Tastenkombinationen funktionieren auch mit Word-Mobile, wenn Sie an das Mobilgerät eine Tastatur angeschlossen haben. Eine Auflistung aller eingebauten »Shortcuts« finden Sie in dieser Liste. Diese ist auf amerikanisches Tastaturlayout bezogen, die Kombinationen weichen in einigen Fällen von den deutschen ab. Die falschen Tastenkürzel werden in allerlei Netz-Quellen ungeprüft kolportiert.

Tastenbelegungen zu ändern, ist kein Luxus, sondern angesichts des Änderungswahns der Microsoft-Entwickler eine »Überlebensstrategie«. Gerade vor einigen Monaten wurde ohne triftigen Grund ein ganzes Paket an altbewährten Tastenkombinationen geändert. Sicher kann man auch nach Jahrzehnten noch umlernen, doch wenn eine Tastenkombination ersatzlos entfällt, hört der Spaß auf.

Allein dieser Akt mit zweifelhaftem Nutzen zeigt die Notwendigkeit, eigene Tastenkombinationen festlegen zu können.

## Tastatur anpassen per Dialog

Der für die Zuweisung von Tastenkombinationen zuständige Dialog ist nur auf Umwegen (über MENÜBAND ANPASSEN, FORMATVORLAGEN ÄNDERN, SYMBOLAUSWAHL oder MAKRO AUFZEICHNEN) zu erreichen. Deshalb kann es sinnvoll sein, eine Tastenkombination einzurichten, die direkt zu diesem Dialog führt. Der Screenshot zeigt die dafür erforderlich Auswahl an.




Setzen Sie den Cursor ins Feld NEUE TASTENKOMBINATION und drücken Sie die gewünschte Tastenkombination. (Falls zu Ihrer Tastenkombination AltGr gehört, wird das als Alt+Strg dargestellt. Funktioniert aber dennoch auch mit AltGr.)

## Tasten mittels VBA belegen

Nicht alle Tastenkombinationen lassen sich mit dem Dialog TASTATUR ANPASSEN einrichten. Kombinationen *nur* mit ⇧ zum Beispiel ignoriert das Feld NEUE TASTENKOMBINATION.

Die VBA-Methode `Keybindings` dagegen ist nicht eingeschränkt, doch ihre Syntax ist kryptisch. Die Tastenzuordnung wie oben im Screenshot lautet in VBA:

```
KeyBindings.Add KeyCode:=BuildKeyCode(wdKeyControl, wdKeyAlt, wdKeyShift, wdKeyT),
KeyCategory:=wdKeyCategoryCommand, Command:=KeyboardCustomizationWord)
```

## BuildKeyCode

Beim Druck auf eine Taste sendet das Keyboard eine Nummer an das Betriebssystem.

Die Klammer nach *BuildKeyCode* enthält die Codenummern der einzelnen Tasten, durch Kommata getrennt. Zur besseren Lesbarkeit gibt es für die meisten Tastennummern Word-Konstanten, die mit `wdKey` beginnen und auf eine verständliche Tastenbezeichnung enden, im Beispiel also die Tasten Strg+Alt+⇧+T.
Die von Microsoft veröffentlichte Liste der wdKey-Konstanten ist unvollständig. Wie Sie dort nicht verzeichnete Tastennummern ermitteln, finden Sie im Abschnitt »Keycodes QWERTY vs. QWERTZ«.

## KeyCategory

Mit *KeyCategory* legen Sie fest, was mit dieser Tastenkombination in Word eingefügt oder aufgerufen wird.

| Konstante | Wert | Aufruf |
|---|---|---|
| wdKeyCategoryNil | -1 | keine Zuweisung |
| wdKeyCategoryDisable | 0 | Taste(nkombination) deaktivieren |
| wdKeyCategoryCommand | 1 | Befehl/Funktion/Makro |
| wdKeyCategoryMacro | 2 | Makro |
| wdKeyCategoryFont | 3 | Schriftart |
| wdKeyCategoryAutoText | 4 | AutoText |
| wdKeyCategoryStyle | 5 | Formatvorlage |
| wdKeyCategorySymbol | 6 | Symbol |
| wdKeyCategoryPrefix | 7 | Präfix |

## Command

Mit *Command* letztendlich bestimmen Sie konkret, welche Aktion ausgelöst wird, wobei der hier zugewiesene Inhalt zur *KeyCategory* passen muss.

# Speicherort der Tastenkombination

KeyBindings werden im Dokument oder in einer Dokumentvorlage gespeichert. Der Speicherort entscheidet über den Wirkungsbereich der Tastenbelegung.

Deshalb muss den KeyBindings.Add-Zeilen mit `Application.CustomizationContext` eine Definition vorangestellt werden.

| CostumizationContext | Tastenkombination verfügbar … |
|---|---|
| NormalTemplate | in allen Word-Instanzen |
| andere Vorlage (siehe unten) | nur in Dokumenten, die diese Vorlage verwenden |
| Document | nur in diesem Dokument |

Nach den `KeyBindings` ist die eingangs definierte Datei zu schließen.

**Beispiel**

Nahezu alle Programme verwenden Strg+⇧+S für die Funktion SPEICHERN UNTER, nur Word nicht, hier steht Strg+⇧+S für FORMAT ÜBERNEHMEN; SPEICHERN UNTER ruft man mit F12 auf. `KeyBindings` schafft Abhilfe:

```
Sub TastenkombiSpeichernUnter()
With Application
   .CustomizationContext = NormalTemplate
   .KeyBindings.Add KeyCode:= BuildKeyCode(wdKeyControl, wdKeyShift, wdKeyS), _
   KeyCategory:=wdKeyCategoryCommand, Command:="FileSaveAs"
   NormalTemplate.Save
End With
End Sub
```

## Keybindings anderen Dokumentvorlagen zuweisen

Grundsätzlich lässt sich das Ziel der Tastenkürzel mit
`Application.CustomizationContext = "Dokumentvorlage.dotm"`
beliebigen vom aktuellen Dokument verwendeten Dokumentvorlagen zuweisen. Häufig »zickt« Word dabei jedoch und verweigert den Zugriff. Sicherer fahren Sie mit der Indexnummer der Vorlage, die Sie mit folgenden Befehlen ermitteln:

```
Dim VorlNr As Long
VorlNr = 1 'Indexnummer der Vorlage
For Each aktTemp In Templates
If aktTemp.Name = "Dokumentvorlage.dotm" Then Exit For
VorlNr = VorlNr + 1
Next
```

Die so ermittelte lfd. Nummer verwenden Sie in der Zuweisung

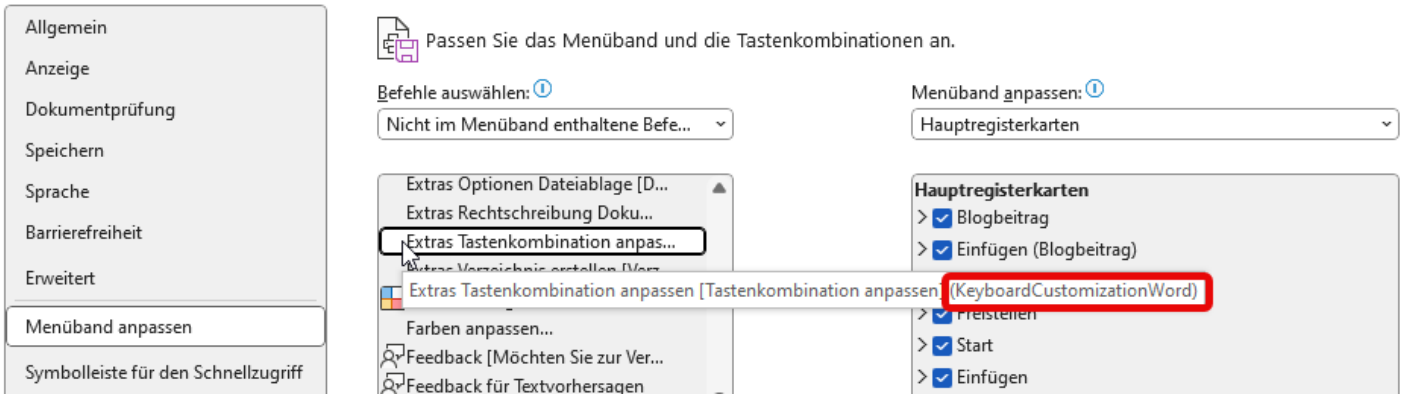
```
Application.CustomizationContext = Templates(VorlNr)
```

# Command ermitteln

Für den KeyBindings-Command-Eintrag ist die interne Funktionsbezeichnung erforderlich.

1. Rechtsklick ins Menüband | MENÜBAND ANPASSEN
2. Suchen Sie in der Liste BEFEHLE AUSWÄHLEN nach der gewünschten Funktion und verweilen Sie mit dem Mauszeiger darauf.

Eine QuickInfo mit der Befehlsbeschreibung erscheint. Der Text in Klammern ist der benötigte Befehlstext.




# Keycodes QWERTY vs. QWERTZ

Die `wdkey`-Konstanten haben zwei Handicaps:

- Sie beziehen sich auf die US-Tastatur, die sich nicht nur durch die Positionen von Z und Y von der in Deutschland, Österreich und der Schweiz unterscheidet. In den »Randbereichen« helfen die Namen der Konstanten deshalb wenig, weil nicht alle `wdKey`-Konstanten an die Tastaturlayouts angepasst wurden.
- Für einige Tasten wie Num, ↓, →|, AltGr gibt es überhaupt keine `wdKey`-Konstante.

Ersatzweise können Sie die Werte der Tasten verwenden. Die doppelt vertretenen Tasten haben denselben Wert, auch Alt und AltGr. Sowohl Alt als auch AltGr haben beide den Wert 18; sie gelten als identische doppelte Tasten links und rechts, wie auch ⇧ und Strg, ebenso ↵ und Enter im Ziffernblock. Um AltGr zu codieren, muss sie im Code als Kombination Strg+Alt eingegeben werden, also `wdKeyControl, wdKeyAlt.`

## Tasten, die Sie nicht anders belegen sollten.

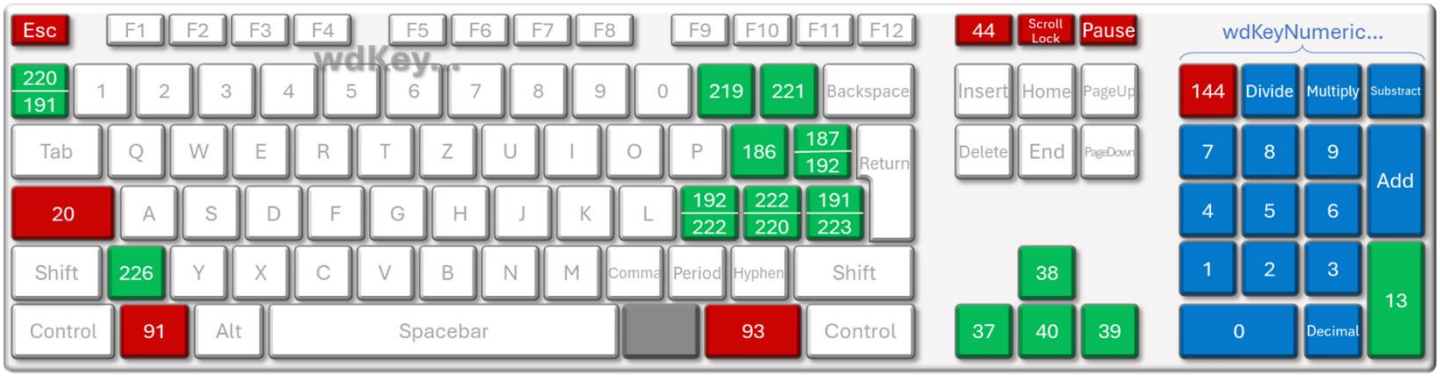
Esc und Pause werden von vielen Programmen als Abbruchfunktion genutzt.

, Roll und sind für systemnahe Funktionen im Einsatz.

und Num wirken unmittelbar auf die Tastatur.

Druck wird von den meisten Screenshot-Apps als Auslöser verwendet.

## Tastaturschema mit Ansprechcodes




Die Grafik zeigt die unterschiedlichen Ansprechmöglichkeiten:

| | |
|---|---|
| **Weiße Tasten** | können mit `wdKey___` angesprochen werden, wobei ___ für die Tastenbezeichnung steht. |
| **Blaue Tasten** | können mit `wdKeyNumeric___` angesprochen werden. |
| **Grüne Tasten** | können nur mit der Codenummer angesprochen werden. |
| **Rote Tasten** | sollten möglichst nicht umdefiniert werden. |
| **Graue Taste** | kann nur mittels `wdKeyContol, wdKeyAlt` angesprochen werden. |

**Geteilte grüne Tasten** zeigen oben den Code für das D/A-Keyboard, unten für das CH-Layout.

## Keycodes online ermitteln

Auf der Seite von Toptal ermitteln Sie den Keycode und mehr zu jeder Taste auf Ihrer Tastatur.

# Keybindings entfernen

Soll eine Tastenkombination entfernt werden, ist das mit dem Dialog TASTATUR ANPASSEN sehr umständlich. Dieses Makro vereinfacht das:

```
Sub KillKey()
Dim Count As Long
Count = 1
For Each aktTemp In Templates
    If aktTemp.Name = "Name der Vorlage" Then Exit For
    Count = Count + 1
Next
Application.CustomizationContext = Templates(Count)
FindKey(BuildKeyCode(wdKey___, wdKey___)).Clear
Templates(Count).Save
End Sub
```