Typinator™

User's Guide

Version 7.8
November 2018

Contents

License agreement

Note: If you use Typinator under the terms of a site license, the following statements do not apply for you. Please ask your system administrator about the terms and conditions of the site license agreement.

Ergonis Software grants the customer a non-exclusive and non-transferable license to use Typinator™ (the "software") as long as the customer complies with the terms of this agreement. With a single-user license, you (the customer) may use two copies of the software on two computers that are owned by you. A multi-computer license may be purchased to allow the software to be used on more than two computers. A family pack license allows you to install and use the software on up to five computers as long as those computers are used only by family members living in the same private household. A family pack license does not extend to business or commercial users. The Typinator™ documentation and the software are copyrighted and all rights are reserved. The software and the information in the manual are subject to change without notice. The manual and the software may not, in whole or in part, be copied, photocopied or reproduced without written consent from Ergonis Software, although you may make copies of the software for backup purposes only. You may not loan, rent or license the software or the manual.

Limited warranty

Ergonis Software makes no warranties, either express or implied, that the software will meet your requirements, that operation of the software will be uninterrupted or error free, or that all software errors will be corrected. In no event will Ergonis Software or the author be liable for direct, indirect, special, incidental, or consequential damages resulting from any defect in the software or documentation, even if they have been advised of the possibility of such damages.

Contact

If you have any questions about this product, please contact us at:

Ergonis Software GmbH
Herrenstraße 20
4320 Perg / Austria

Fax: +43 720 348424

e-mail: typinator-support@ergonis.com for technical support
typinator@ergonis.com for orders / license keys

WWW: https://www.ergonis.com about the company

About Typinator

Typinator is a simple yet powerful application that helps you to quickly type phrases or insert pictures in documents. Basic configuration is easy: You just define abbreviations and their expansion text or picture. Whenever you type one of these abbreviations in an arbitrary application, Typinator inserts the corresponding expansion.

Typinator can be helpful for a variety of tasks. You can use it to increase your …

Personal Productivity:

- Set up a list of often used e-mail phrases, addresses and other boilerplates
- Insert the current date and/or time in a variety of formats with a few keystrokes
- Auto-correct commonly misspelled words across all the application you use.
- Insert pictures, such as signatures and smilies

Business Productivity:

- Create standard customer service responses
- Acknowledge customer orders and send shipping advisories (with the current date and time automatically inserted into your standard text)
- Insert logo, product schematics, and maps
- Auto-correct your most frequent typing errors
- Insert Unicode symbols by typing a few special regular characters (e.g., type "->" to insert an arrow symbol)

Development Productivity:

- Create templates for code fragments, code blocks, and templates
- Implement Documentation Standards
- Define shorthands for frequently used Unix commands

System requirements

Typinator requires macOS 10.6 or newer. macOS 10.11 or newer is recommended.

To use regular expressions, macOS 10.7 or newer is required.

Installation

Installation of Typinator is easy. Download the disk image from our web server:

https://www.ergonis.com/downloads/

Mount the disk image and double-click the Typinator icon in the Finder window that opens. Typinator helps you with the installation by copying itself to the Applications folder and launching the installed copy.

You will see that Typinator adds a new icon to your menu bar:

This icon tells you that Typinator is now active and watching your keystrokes to expand any abbreviation you type. To configure Typinator, click the icon. To access a menu with common commands, click the small triangle next to the icon.

When you start Typinator for the first time, the abbreviations window appears. We recommend adding Typinator to your list of login items by clicking the Preferences icon in the toolbar and enabling the checkbox “Automatically start Typinator at Login”.

To monitor keystrokes and insert expansions in documents, Typinator needs permission to do so. This is controlled by a setting in System Preferences. When Typinator does not have the required permission, it will tell you how you can enable the corresponding option in System Preferences. Please note that you need administrator privileges to enable this checkbox. If you do not have the necessary privileges, ask your system administrator to enable this option for your Macintosh.

First steps

When you install Typinator for the first time, it comes with four predefined abbreviations:

Abbreviation	Expansion
dt	{YYYY}-{MM}-{DD}
typicon	*Picture*
typurl	http://www.ergonis.com/products/typinator/
wbr	With best regards,¶John Smith¶(Smith & Smith, Ltd.)

Typinator also opens a tutorial document that demonstrates in a few steps how Typinator works. At the end of the tutorial, you should have an impression what Typinator can do for you to simplify your daily typing tasks.

Defining abbreviations

Before you can use your own abbreviations, you need to define them. Click the “+” button below the abbreviation list to add a new item, then enter the abbreviation and the expansion text. You don’t need to confirm or save your settings; all changes take effect immediately.

Once you are done adding abbreviations, you can close the Typinator window. Typinator will then work in the background, watch your keystrokes and expand abbreviations as you type them. If you wish to add more abbreviations or edit existing abbreviations, just click the Typinator icon in the menu bar.

Recommended abbreviations

Abbreviations must not conflict with each other. If there is any problem with an abbreviation, Typinator displays an error symbol in the list and a message in red below the abbreviation:

Abbreviations marked with the error symbol are inactive and will not get expanded when you type them. To activate them, you need to fix the problem by making the abbreviation unique.

Note that Typinator does not expand abbreviations in the middle of a word. This means that you could, for example, safely use "mm" as an abbreviation because it would get expanded only when you start a new word with "mm", but not in "hammer".

To create unique abbreviations that are unlikely to occur elsewhere, you may want to use a special character as prefix or suffix. If you use a keyboard with the US layout, you could add the symbol "/" to an abbreviation (such as "/cal" or "cal/") to make sure it does not expand by accident. Since abbreviations are often written with a period (such as "etc."), you may find it convenient to use a period as a suffix. In this way, you just need to avoid whole words, as they would expand at the end of sentences.

Hint: You can define common prefixes and suffixes for many abbreviations at once easily in the "Set Info" window. For more information, see the section *Sets* further below.

Using a special character as a prefix is also helpful for abbreviations that you want to use within words or immediately after words. For example, you might want to use "TM" to generate the trademark symbol "™". Typing "TypinatorTM" would not work, but when you begin the abbreviation with a slash ("/TM"), you can type "Typinator/TM" to write "Typinator™".

Another possibility to define replacements that should be made within words is by means of regular expressions. This is a powerful feature that lets you do much more than just replacing one text pattern with another. For more information, please see the separate section on *Regular Expressions* further below.

Whole word

Abbreviations are most useful when they are very short, so you can remember and type them easily. On the other hand, short abbreviations will more likely expand by accident if a regular word happens to begin with the same letters. For example, you might want to use "ty" as an abbreviation for "Typinator", but you don't want it to expand when you actually want to write "typically" or "tyre".

When you enable the "Whole word" option for an abbreviation, Typinator expands it only when the next typed character is neither a letter nor a digit. This means that the abbreviation will not expand immediately after you type the letter "y" because Typinator must wait to

see what you type next. When you then type a space, period, or any other separator character, Typinator recognizes the abbreviation “ty” as a whole word and expands it.

Whereas the “Whole word” option is useful for short abbreviations that you type deliberately, you may want to disable it for word stems when you want Typinator to automatically correct typing errors. For example, when you often mistype “receive” as “recieve”, you could define “reciev” as an abbreviation with the expansion “receiv”, *without* the “Whole word” option. Typinator will then correct misspelled words like “recieve”, “reciever”, or “recieving”.

You will notice that the “Whole word” option is available only if the abbreviation ends with a letter or digit. This is deliberate because it makes sense only when the abbreviation can be considered a “word” in the first place.

If you just typed a “whole word” abbreviation and do *not* want to expand it when you type a separator character, quickly press and release the command key (⌘) before the separator. You can use this trick to write about abbreviations, such as

I use the abbreviation “ty” for “Typinator”.

In this example, just hit the ⌘ key after *ty* and before the quote to prevent the expansion.

Case variations

If an abbreviation consists of letters, you can use the pop-up menu next to the abbreviation to specify how Typinator should handle case variations:

If **Case must match** is selected, you need to type the abbreviation exactly as you have defined it. When you select **Case does not matter**, Typinator also expands “btw” when you type “BTW” or even “bTw”.

When you select **Case affects expansion**, mixed case in a typed abbreviation affects the expansion in the following way (assuming that “btw” stands for the expansion “by the way”):

- If all typed letters are capital letters (“BTW”), the expansion will be inserted in all caps as well (“BY THE WAY”).
- If you start the abbreviation with a capital letter and the remainder contains at least one lower case letter (e.g., “Btw”), the first letter of the expansion appears in upper case (“By the way”).
- If the first typed letter is a lowercase letter, the expansion starts with a lower case letter (“by the way”).

In short: You can use this setting to generate correctly capitalized phrases at the beginning and in the middle of a sentence.

Formatted text

To create an expansion with multiple fonts and/or embedded pictures, select “Formatted Text” as the expansion type. You can now apply various formats to the text using the formatting menus and insert pictures via Copy/Paste or by dragging image files into the text field.

When you type an abbreviation with a formatted expansion, the current application must be able to handle text with multiple fonts. This works for most word processors. If an application uses a fixed font for all text, the expansion will assume the font used by the application; additional formats applied in Typinator will not appear.

Pictures

To create a picture expansion, select “Picture” as the expansion type, then insert the picture of your choice using one of the following techniques:

- Copy a picture into the clipboard in another application, then click the Paste button in Typinator.
- Drag an image file from the Finder into the picture area.

HTML expansions

The fourth type of expansion is “HTML”. This is not just text that happens to contain HTML code, but rather a distinct expansion format that can be described in HTML.

HTML expansions are most useful for mail applications that support rich text messages, such as Apple Mail, Microsoft Outlook, Airmail, Thunderbird, as well as many web-based mail services like iCloud Mail and Gmail. These applications use the HTML format internally for describing and transmitting rich text messages. With Typinator’s HTML expansions, you can create snippets for mail messages with sophisticated formatting, such as:

- text styles, including CSS formatting
- separator lines
- tables
- links
- references to pictures

Some mail applications may not fully support all types of HTML content. For example, the <hr> tag and certain <img> tags may not work in some applications. We therefore recom-

mend that you test your HTML expansions thoroughly: Create a mail message in your preferred mail application, use Typinator to insert HTML fragments, then send the message to yourself and also try whether other people with different mail programs on other platforms receive your messages in the intended format.

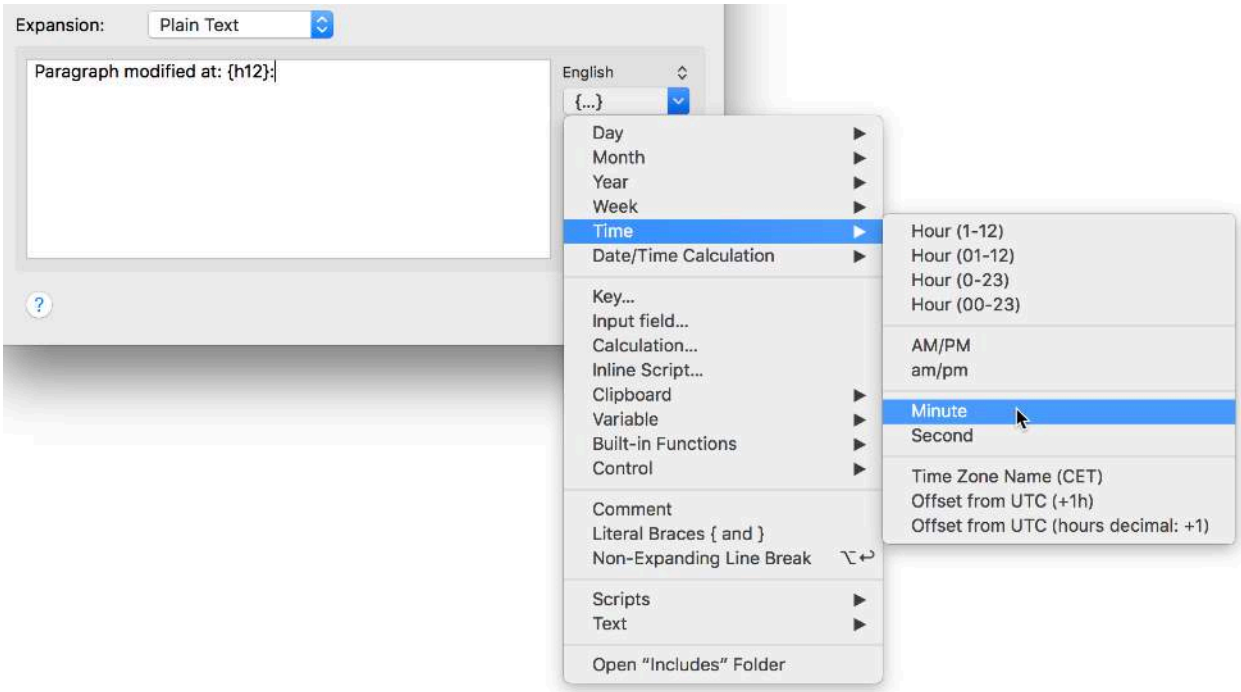
Descriptions

You can add a description to an item by clicking the button with the blue speech bubble below the list of abbreviations. When you enter a description, it appears in the list in place of the expansion. Descriptions are especially useful for long expansions (where only the first few words are visible in the list) and for pictures expansions (which otherwise appear just as "Picture" in the list).

Markers in the {…} menu

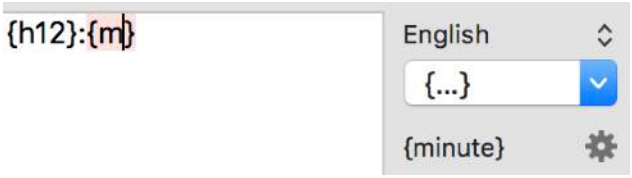
When you create a plain or formatted text expansion, you will notice the {…} pop-up menu next to the expansion text field. The items in this menu create special markers at the insertion point in the expansion. These markers typically are enclosed in curly braces and turn simple text fragments into powerful dynamic snippets that contain the current or calculated date and time, clipboard contents, cursor positions, variables, interactive input fields, contents of separately stored text files and results of script invocations.

To insert any of these markers in your expansion text, set the insertion position where you want the marker to appear and select the desired element from the {…} menu:



Assistants

When you place the insertion point inside a marker, Typinator highlights the entire marker and displays a short description below the {…} pop-up:



Typinator simplifies creation and editing markers by means of "assistants". If an assistant is available for the currently selected marker, a gearwheel icon appears next to the description. Click that icon to edit the marker with the help of the assistant. If you are familiar with the notation of markers, you can also edit their definition directly in the expansion text.

To quickly open an assistant, you can also double-click the enclosing braces of the marker. If no assistant is available for a certain marker, double-clicking selects the whole marker.

Date and time

To include the current date or time in an expansion text, select the corresponding item(s) from the Day, Month, Year, Week, or Time submenus.

For example, if you select "Minute", Typinator inserts {m} in the expansion text. When you type the corresponding abbreviation, Typinator replaces this item with the minutes of the current hour. Since you can compose date and time templates using the individual parts, you have some flexibility in how to format the time and date.

For the day of the week or the name of the current month, you can choose in which languages Typinator should insert it. The language is specific for each abbreviation, so you can, for example, create separate English and French expansions if you are working in multiple languages. The language menu contains all languages as defined in the "Language & Text" section of System Preferences.

There are three markers for time zone offsets ("Offset from UTC"). The first one {z} creates a standard format with "h" for hours and "m" for minutes. For example, the UTC offset for New Delhi is shown as "+5h30m". The second marker {zn} creates the offset in the notation "+hh:mm" (e.g., +05:30), and the third marker {zh} creates the hours in decimal notation ("+5.5" for New Delhi), which is suitable for date and time calculations.

Date and time calculations

The submenu "Date/Time Calculation" contains items like "+/– Day" and "+/– Hour". These items perform simple date and time calculations and let you temporarily adjust the date and time used by other markers. For example, the "+/– Day" item creates a marker {{dayDelta=+1}}.

The default value "+1" switches the date to tomorrow, but you can replace it with any positive or negative value. For example, {{dayDelta=+14}} moves the date forward by two weeks. All date and time markers that follow will now use the new date. To switch back to the current date, use a marker with the value 0.

As an example, try the following phrase:

> You can expect the shipment in a one week {{dayDelta=+7}}({NN} {D}).

Note that the calculation markers by themselves do not produce any output but only modify the date or time. You need to add the actual date and time markers to insert the parts of the date and time in the desired format, such as {NN} {D} in the example above.

The variables dayDelta, hourDelta, minuteDelta and secondDelta may have fractional parts. For example, {{dayDelta=+1.5}} advances by one day and 12 hours, and {{hourDelta=+0.25}} moves the clock forward by 15 minutes.

Date and time calculations are always relative to a reference date and time. Normally, this is the moment when the expansion takes place. For example, the marker {{dayDelta=+3}} means "three days in the future *from now*". Instead of using the current date and time, you can explicitly specify an arbitrary date and time with the "Set Reference Date" and "Set Reference Time" markers. For example, "Set Reference Date" creates a template in the form

> {{refDate=yyyy-mm-dd}}

Just fill in the desired year, month and day, such as {{refDate=2014-06-17}}. All subsequent date markers will then refer to this reference date.

You can also omit parts of the date and time format by entering anything that is not a number. For example, you could write {{refDate=*-12-31}} to refer to the last day of the current year. To switch back to "now", just enter anything that does not conform to the format "yyyy-mm-dd". For example, you can use {{refDate=today}} to return to the default value of the reference date (i.e., the date when the expansion takes place).

Keystrokes

The **Key...** command opens an assistant for inserting a keystroke inside an expansion. This can be any key with an arbitrary combination of ⇧, ⌘, ⌥, and ctrl. For example, you can insert a {tab} marker in the middle of an expansion to simulate a tab keystroke at this point. Typinator then inserts the text up to the marker at the current insertion point, sends a tab keystroke to the target application and then inserts the rest of the expansion. You can use this technique to jump from field to field in web forms and other dialog windows.

Clipboard functions

Text from Clipboard creates a marker {clip} that tells Typinator to insert a plain text representation of the current clipboard contents.

The following example shows a letter template that combines a clipboard marker with a cursor placement:

> Dear {clip},
> {^}
> With best regards,
> John Smith

Assuming that you have previously copied the name “Mary Jones” to the clipboard, this expansion will result in:

> Dear Mary Jones,
> ¦
> With best regards,
> John Smith

The cursor position will be at “¦” in the second line, so you can immediately continue typing your letter.

Paste Clipboard inserts a {paste} marker, which performs a “Paste” operation in the middle of an expansion. Note that this is different from the clipboard marker {clip}, which inserts the plain text representation of the clipboard’s contents in the expansion. In contrast, {paste} temporarily stops the expansion process and then simulates a selection of the Edit/Paste menu command. This means that {paste} can be used to insert pictures, formatted text and other types of data in the target document.

Variables, interactive input fields and calculations

Similar to the date and time markers, you can define your own **variables** that can be inserted in expansions. For example, the notation {{releaseDate}} refers to a variable named “releaseDate”. When you type the corresponding abbreviation, Typinator replaces the marker with the current value of the variable.

To assign a value to a variable, use the notation {{releaseDate=April 25}}. The entire text to the right of the equals sign up to the closing double braces is taken as the new value of the variable. Typinator remembers variables globally, even when you quit Typinator or restart your computer. This means that you can set a variable in one expansion and then use it in multiple places in other expansions.

You can also use date and time markers in variable assignments. For example, the assignment {{releaseDate={NN} {D}}} replaces the variable with the current month and day. To change the releaseDate variable, you just need to trigger the corresponding expansion once, and then you can refer to the variable’s value from other expansions, such as

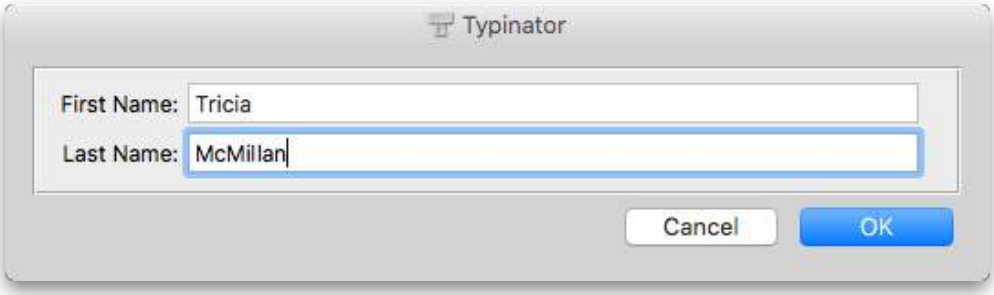
> We released the current version on {{releaseDate}}.

You can even use the {clip} marker to assign the content of the clipboard to a variable, like this:

> {{specialOffer={clip}}}

Interactive input fields are another possibility to create flexible expansions. Like variables, they are represented by special markers inside an expansion. You can use them to interactively enter parts of an expansion at the time when you type the corresponding abbreviation.

The basic syntax of an input field marker looks like this: {{?First name}}. You can replace “First name” with any descriptive text. This will become the label that appears in front of the input field when Typinator asks you to enter it. When an expansion contains two such fields “First Name” and “Last Name”, Typinator shows a dialog window in which you can enter the actual values:



After confirming the dialog window by clicking the OK button, Typinator replaces the input markers within the expansion text with the values that you enter. When an expansion contains multiple such input markers, Typinator displays a single window with all the input fields, in the same order as in the expansion text. Multiple input fields with the same label appear in the dialog window only once, so you can, for example, insert the first name and last name fields in multiple places inside a letter template.

Typinator remembers the values that you enter. When the same input markers are used again later in the same or another expansion, Typinator presets the input field with the values from the previous entry. You can override this behavior by explicitly specifying an initial value in angle brackets, such as:

{{?Operating System<macOS>}}

If an initial value is present, Typinator always presets the input field with the specified string, instead of what you entered the last time. You can also use other markers in the specification of the initial string. For example, you can preset a “Month” input field with the current month, using the {M} marker:

{{?Month<{M}>}}

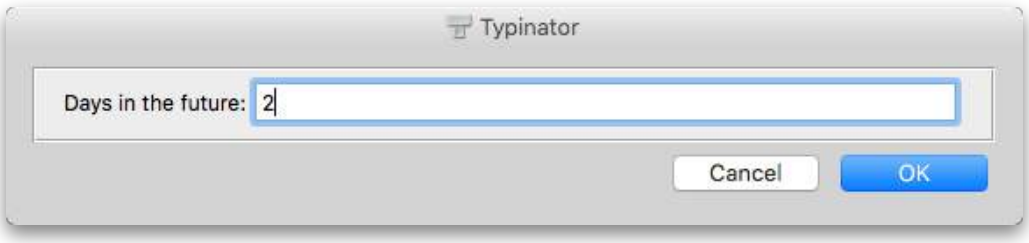
You can also combine interactive input fields with assignments. For example, Typinator interactively asks for a release date when you use the following marker inside an expansion: {{releaseDate=?Release date}}

In this case, the entered value is assigned to the releaseDate variable, but it does not appear in the final expansion, unless you insert the value of the variable with {{releaseDate}}.

Note that the date and time calculations are actually also based on variables with predefined names. For example, you can use an input field in the definition of "dayDelta" like this:

{{dayDelta=?Days in the future}}{YYYY}-{MM}-{DD}

Typinator will then ask for the “Days in the future”:



And when you enter the value 2, Typinator assigns that to the predefined dayDelta variable, so that the date markers {YYYY}-{MM}-{DD} yield the day after tomorrow.

Note that the predefined "…Delta" variables are not remembered across expansions. After processing an expansion, Typinator automatically resets all these variables to 0, so the date and time markers correctly refer to the current date and time when the next expansion takes place.

You don't need to remember the notation for input fields, since Typinator provides an assistant for creating and editing input fields. Using the assistant, you can create not only text fields, but also pop-up menus for selecting one of multiple choices as well as checkboxes for selecting one of two text fragments (for "yes" and "no"). Just select the desired field type from the pop-up menu in the assistant:

Alternatives lets you select one of multiple predefined choices. Each of the choices has a title that appears in a pop-up menu, and an optional "extended text result". If you attach an extended text result to a choice, Typinator inserts that text when this alternative is chosen. If an item does not have an extended text result, the item's title is inserted.

You can also check "Allow free text", which turns the pop-up menu into a "combo box", where you can choose one of the predefined alternatives and enter other text as well.

Option can be used for yes/no choices, which are represented as checkboxes. You can define which text fragment is inserted for both cases. To create an optional text fragment that can be included or excluded, enter the text in the "Yes" part and leave the "No" part empty.

Heading, **Description**, and **Separator Line** can be used to add structure to the form. Form fields are displayed in the order in which they occur in the expansion. If you prepare sophisticated expansions, you can use headings and separator lines to group form fields, and you can add descriptions to explain the purpose of input fields.

Calculations are another way to create even more powerful expansions. To calculate a value that depends on variables, use the number sign (#) in a marker like this:

```
{{#count*price}}
```

The number sign introduces a calculation, which can contain any variables, numbers and the same operators that are supported in the Quick Search field (see the section about Typinator's built-in *pocket calculator*). In the example above, count and price would be variables that have been computed earlier or entered via an input field.

The results of computations can be assigned to variables like this:

```
{{total=#count*price}}
```

Like other assignments, this does not produce a visible result but simply assigns the computed value to the variable *total*, which can then be used in other places in the same expansion or subsequent expansions: {{total}}.

When a calculation results in a floating point number, you can add a colon with the desired number of digits after the decimal point, as shown in this example:

```
The total costs are ${{#count*price:2}}.
```

When, for example, count is 10 and price is 4.99, the result of the calculation would be "49.9". With the suffix ":2", Typinator formats the result with two digits after the decimal point, so that the expansion finally results in

```
The total costs are $49.90.
```


You can use assignments with calculations for a number of cool expansions. For example, you can count certain expansions like this:

```
{{donations=#donations+1}}Including your donation, we have already
received {{donations}} contributions. Thank you!
```

You can use the same technique for date and time calculations. For example, you can increment the dayDelta variable to advance to the next day from a previous calculation, like this:

```
{{dayDelta=#dayDelta+1}}
```

Calculations are a rather simple concept, but the possibilities are virtually endless. In combination with input fields, you can use Typinator to write a complete bill with just a few keystrokes, with discounts, taxes, etc. Combined with some scripts (such as the Convert script from our Download Extras page), you can even include the amount in other currencies, based on the current exchange rate.

For more inspiring examples, visit our "Download Extras" web page and check out the packages in the "Typinator Scripting Extras" section. These packages contain useful scripts (as described further below) and come with sample abbreviations that show how you can use these scripts in elegant ways with input fields and variables.

Comments

The **Comment** item in the {…} menu creates a marker of the form {{--comment}}. Typinator removes such comments from the expansion text. You can use them for hints and remarks, such as:

```
{{--IMPORTANT: Trigger this expansion once on release days.}}
```

Literal braces {. and .}

Since the special markers are enclosed with curly braces { and }, you may occasionally run into a situation where you want to insert such braces literally in an expansion, but Typinator creates an unwanted result because it treats the braces as markers. To prevent this from happening, you can mark braces with periods to tell Typinator that you want them to appear literally in the expansion. Just add a period after an opening brace or before a closing brace, as shown in the following example:

```
Typinator replaces the marker {.clip.} with the content of the clipboard.
```

In this case, Typinator will simply remove the periods, but it will not insert the clipboard.

In most cases, Typinator is smart enough to figure out where braces are meant as delimiters of markers, so you do not need to mark every literal brace with a period. For example, braces in source code snippets for C or Java will typically appear as intended. You will need to "escape" incorrectly treated braces only in rare circumstances, for example, when you wish to insert statements from the clipboard inside a block that is surrounded by braces, like this:

```
if (…) {.{clip}.}
```

In this example, the outer braces need to be marked with periods because they are meant literally, whereas the inner pair of braces has special meaning for Typinator. Without the periods, Typinator would treat {{clip}} as a variable with the name clip.

To insert a template with braces, select the item “Literal Braces { and }” from the {…} pop-up menu.

Non-expanding line break

The item “Non-Expanding Line Break” inserts a line break and adds the symbol “↩” at the end of the previous line. As a shorthand, you can also press the option key and hit the return key. For example, when you insert such a line break in the middle of “abcdef”, you get:

abc↩
def

This special type of line break helps to improve readability of complex expansions, but it is finally removed from the result. In the example above, the resulting expansion is still “abcdef”, without the line break and the “↩” symbol.

Non-expanding line breaks are especially helpful with interactive input fields, assignments and calculations. Here is an example:

{{product=?Product}}{{dayDelta=?Days in the future}}We will release {{product}} on {NN} {D}, {YYYY}.

You might be tempted to break this expansions into multiple lines, like this:

{{product=?Product}}
{{dayDelta=?Days in the future}}
We will release {{product}} on {NN} {D}, {YYYY}.

But this form would create two empty lines at the beginning, because the assignments are removed from the result, but the line breaks after them are retained.

Non-expanding line breaks suppress these extraneous line breaks in the created result:

{{product=?Product}}↩
{{dayDelta=?Days in the future}}↩
We will release {{product}} on {NN} {D}, {YYYY}.

Built-in functions

The “Built-in Functions” submenu contains a few powerful functions for transforming text fragments within expansions:

{/**Lowercase** *any text*} converts *any text* to lower case.

{/Lowercase Here is some TEXT} → here is some text

{/**Uppercase** *any text*} converts *any text* to upper case.

{/Uppercase Here is some TEXT} → HERE IS SOME TEXT

{/**Capital** *any text*} capitalizes *any text* by converting the first letter of each word to upper case.

{/Capital Here is some TEXT} → Here Is Some Text

{/**DecodeHTML** *text in HTML format*} converts HTML entities into their readable form.

{/DecodeHTML Österreich} → Österreich

{/**EncodeHTML** *text*} converts reserved HTML characters in *text* for use in web pages.

{/EncodeHTML Miller & Sons} → Miller & Sons

{/**Length** *any text*} returns the number of characters in *any text*

{/Length Hello} → 5

{/**Unicode** *character number(s)*} inserts the Unicode character with the given *character number*. Multiple characters can be inserted at once by separating the numbers with commas or spaces.

{/Unicode 8984} → ⌘

{/**Left** *n/any text/*} returns the *n* leftmost characters of *any text*

{/Left 3/Typinator/} → Typ

{/**Right** *n/any text/*} returns the *n* rightmost characters of *any text*

{/Right 3/Typinator/} → tor

{/**Mid** *start/n/any text/*} returns *n* characters in the middle of *any text*, starting at *start*

{/Mid 3/3/Typinator/} → pin

{/**Repeat** *n/any text*} repeats *any text n* times

{/Repeat 3/Ho, } → Ho, Ho, Ho,

{/**Case** */value/case1/result1/case2/result2/…/…/default result/}* compares the given *value* with "*case* values" and returns the corresponding *result* of the first match. If there is a single last value after the list of *case/result* pairs, it is used as the *default result* when the given input *value* does not match any *case* value.

{/Case /2/0/no/1/one/2/two/3/three/more/} → two

{/**Replace** */original/replacement//any text/*} replaces all occurrences of *original* in *any text* by *replacement*

{/Replace /s/abra//scads/} → abracadabra

This function can contain multiple *original/replacement* pairs, delimited by //, as in:

{/Replace /1/one/2/two/3/three//1234} → onetwothree4

{/**Regex** */original/replacement//any text/*} similar to /Replace, but with regular expressions (requires macOS 10.7 or newer)

{/Regex /([A-Z])([0-9])/$2$1//This is A1 quality./} → This is 1A quality.

{/**Choose** *x/text1/text2/text3/…/*} inserts one of the text items text1, text2, etc. The first parameter (x) determines which item is chosen. x can be:

a number, which is taken as the index of the desired item:

{/Choose 2/one/two/three/} → two

a regular expression pattern, which chooses the first matching item:

{/Choose r/one/two/three/} → two (the first item containing "r")

an asterisks, which chooses an arbitrary item:

{/Choose */one/two/three/} → a random choice of "one", "two", or "three"

{/**Count** /pattern/text/} determines how often the *pattern* (described as a regular expression) occurs in *text*.

{/Count /[0-9]+/first:1 second:23 third:456/} → 3

{/**Index** n/pattern/text/} returns the *n*-th occurrence of *pattern* (described as a regular expression) in *text*.

{/Index 2/[0-9]+/first:1 second:23 third:456/} → 23

{/**Any** /pattern/text/} returns an arbitrary occurrence of *pattern* (described as a regular expression) in *text*.

{/Any /[0-9]+/first:1 second:23 third:456/} → 1 or 23 or 456
(Typinator randomly picks one of the matches at each invocation)

{/**List** /pattern/separator/text/} lists all occurrences of *pattern* (described as a regular expression) in *text*, and inserts *separator* between them.

{/List /[0-9]+/, /first:1 second:23 third:456/} → 1, 23, 456

{/**Extract** */before/after//any text/*} Extracts text between the first occurrence of *before* and the first occurrence of *after* that follows.

{/Extract /V/ //Typinator V6.0 is…/} → 6.0

There can be multiple occurrences of *before*, which are then skipped left to right. The last item before // is still taken as the "*after*" delimiter.

{/**Sort** *any text*} When *any text* consists of multiple lines, the /Sort functions sorts the lines in ascending order.

{/Sort {clip}} sorts the text found in the clipboard.

{/**URL** *http://url*} fetches the web page with the given address and returns the contents.

The names of the built-in functions always begin with a slash. The slashes in the parameter part can be replaced with any other special character. For example, you could use a colon if the text or search and replace patterns contain slashes.

Built-in functions are most powerful when combined with other markers. The possibilities are virtually endless. Here are just a few examples that illustrate what you can do with these functions:

{/Uppercase {clip}}
text from the clipboard, converted to upper case.

{/Regex / +/ //{clip}/}
text from the clipboard, with multiple spaces removed.

{/Replace /{{?Original}}/{{?Replacement}}//{clip}/}
asks for "Original" and "Replacement", replaces all occurrences in the text found in the clipboard, and returns the resulting text.

{/Count /[:upper:]{2,}+/{clip}/}
analyzes the contents of the clipboard and returns the number of words that contain only uppercase letters and are at least two letters long.

{/Case /{{dayDelta}}/0/today/1/tomorrow/in {{dayDelta}} days/}
: uses the {{dayDelta}} variable (see the example for "Days in the future" in the section on *Interactive input fields* above) to generate a relative date ("today" for 0, "tomorrow" for 1, and "in x days" for all other values).

Control functions

The "Control" submenu contains a few items for controlling whether and how Typinator should process expansions:

Cursor Position creates a marker {^} that defines the cursor position immediately after the expansion. When you combine this with markers from the **Keys** submenu, the {^} marker must be in the last segment of the expansion, after the last special keystroke. For technical reasons, the {^} marker does not work in HTML expansions, because Typinator cannot know how exactly the current mail application translates an HTML expansion into formatted text.

Expand other Abbreviations inserts a marker of the form {"abbreviation"}. Replace the placeholder between the quotes with another abbreviation whose expansion you want to insert a this point. This substitution takes place before the processing of all other markers. As a result, the inserted expansion may contain further markers that are processed later. For example, you can insert a completely assembled date, which in turn contains day, month and year markers. The inserted expansion may even contain invocations of further abbreviations, as long as this does not create a circle in which an abbreviation invokes itself.

The text between the quotes can be any abbreviation or regular expression in any other set, as long as the replacement rule is "active" in the current application. Typinator does not consider abbreviations in sets that are disabled for the current application or deactivated by turning off the checkbox in the set list.

When the embedded abbreviation is defined with the "Case affects expansion" option, the text between the quotes determines the case of the result. For example, if you have an abbreviation "tmso" whose expansion begins with "this month's special offer…", the invocation {"Tmso"} capitalizes the first letter and produces "This month's special offer…".

When you mix plain text, formatted text and picture expansions, the following rules apply:

- A formatted expansion within another formatted expansion retains its format.
- A formatted expansion within a plain text expansion loses all formats and embedded pictures.
- A plain text expansion within a formatted expansion assumes the format of the {"abbreviation"} marker in the container expansion.
- Picture expansions can be embedded only inside formatted expansions.

Suppress this Replacement inserts a marker of the form {X}. When this marker is present in an expansion, Typinator does not modify the typed abbreviation in any way. This is helpful for specifying exceptions that would otherwise be processed by other rules. For example, the predefined "Auto-Cap Exceptions" set contains common abbreviations ending with periods. The expansion of these items contains {X}, which overrides the replacement

rule in the “Auto-Capitalize Sentences” set. The result is that you can, for example, write “etc.”, and Typinator will not convert the next lowercase letter into uppercase.

Suppress next Replacement inserts a marker of the form {X>}. When this marker is present in an expansion, Typinator performs the current replacement and then temporarily enters pause mode (with two vertical bars in Typinator’s menu bar icon). When you continue typing, the next replacement is suppressed, and Typinator leaves pause mode. You can use this marker to define a character sequence that avoids certain expansions. For example, you could define an abbreviation “##” with the expansion “{X>}”. When you wish to type an abbreviation without expanding it, type two number signs, followed by the abbreviation. Typinator briefly enters pause mode and automatically leaves pause mode again as soon as you finish typing the abbreviation.

Delay creates a marker of the form {delay:0.5}, which inserts a delay in the middle of an expansion. The number specifies the duration in seconds. This is sometimes helpful when you insert a keystroke, but the target application needs some time to process the keystroke.

Includable text files and scripts

Typinator can include arbitrary text files in expansions. In a default installation, Typinator shows a “Text” submenu in the {…} pop-up menu. This submenu contains two items, “Lorem Ipsum” and “made with Typinator”. When you select “Lorem Ipsum”, Typinator inserts a marker {Text/Lorem Ipsum.txt}, which refers to a text file that contains the famous pseudo-Latin placeholder text. When an expansion contains this marker, it will be replaced with the actual contents of the text file. You can use this technique to maintain phrases that you want to use in multiple abbreviations. For example, you could set up greetings and signatures, store and maintain them in a single location, yet use them in as many places as desired. Text files can be plain text (extension “.txt”) or rich text (extensions “.rtf” and “.rtfd”). Rich text files can contain formatted text and pictures. Obviously, they make sense only in “formatted text” expansions. As an example, try the “made with Typinator” item.

The last item in the {…} menu (Open “Includes” Folder) opens the special folder where Typinator expects these includable text files. You can add your own text files here to include them in the {…} menu. You can also create your own subfolders to organize your text files by topics. For example, you could create subfolders “Signatures” and “Price Lists”, which will then appear as submenus inside the {…} menu.

In addition to static text files, the Includes folder can also contain executable scripts. Typically, these are AppleScript or JavaScript[1] files, but Typinator actually supports all scripting languages that you can use for shell scripts (such as Bash, Perl, PHP, Python, and Ruby). Typinator comes pre-installed with a few AppleScript samples that you can use right out of the box.

Using scripts does not require any programming knowledge. Just select an item from the “Scripts” submenu to insert a script marker in an expansion. You can include multiple scripts in a single expansion and use the same script in multiple expansions. As an example, try the *FinderSelection* script, which results in the marker

{Scripts/FinderSelection.applescript}.

[1] JavaScript support is available on macOS 10.10 (Yosemite) or newer.

When you use this expansion, Typinator runs the FinderSelection script, which replaces the marker with the paths of all items that are currently selected in the Finder.

You will also notice that the Includes folder contains a subfolder named "(About Includes)". This subfolder contains short descriptions of the include mechanism and the default scripts.

If you wish to create your own scripts, please visit our "Download Extras" web page and download a copy of the "Creating Typinator Scripts" package, which contains extensive documentation (English only) and script samples that demonstrate the power of Typinator's scripting features.

Templates for expansions

Plain and rich text files in the Includes folder are treated as "templates" if their file names start with a $ sign or they are located in a subfolder whose name starts with $. Selecting such an item from the {…} menu inserts the *contents* of the file in the expansion instead of a marker that inserts the file when the expansion takes place.

You can use such templates for text or marker combinations that you often need in expansions. For example, you can create a file

YYYY-MM-DD.txt

with the contents

{YYYY}-{MM}-{DD}

and put that into a folder

$Date and Time

When you select this item from the {…} menu, it inserts the marker combination for a date in the expansion. You can use similar templates for creating frequently used input fields, date and time calculations, script invocations, etc.

Inline scripts

Separately stored scripts in the Includes folder are convenient for complex tasks, as they hide the complexity in the external script files. To use such scripts, you do not need to know how to write script; you just need to invoke them in expansions.

However, there are also situations in which it is simpler and easier to directly include the complete script in the expansion. This is what the "Inline Script…" command in the {…} pop-up does: It inserts a script directly in your expansion text, and when the expansion takes place, the script's result gets inserted.

To define the contents of the script, we recommend that you use the assistant, in which you can choose among a number of scripting languages (AppleScript, JavaScript, Shell script, Perl, PHP, Python, Ruby, and Swift). When you create a new inline script with the assistant, it starts with a simple sample script for the chosen language, which illustrates how scripts should be written.

The definition of the script's result depends on the language. For example, to create the text "amount: " followed by a number (value of a variable x, e.g., 123), you would use the following techniques:

AppleScript	return statement	return "amount: " & x
JavaScript	last expression	"amount: " + x

Shell script	echo or printf	printf "amount: $x"
Perl	print	print "amount: $x";
PHP	echo	echo "amount: $x";
Python	sys.stdout.write	sys.stdout.write("amount: "+str(x))
Ruby	print	print "amount: #{x}"
Swift	print	print("amount: (x)", terminator: "")

The specification of the empty “terminator” in Swift tells the print statement that you do not want a line break after the result.

Inline scripts are compiled on demand, just before they are executed. This means that you can use Typinator variables or input fields in scripts. For example, you can use an input form to ask for a number “hours” and assign that value to a variable in AppleScript like this:

```
set nHours to {{?hours}}
```

Inline scripts are ideal for cases where you need a simple script in only one expansion. If it gets more complex or you need a script in multiple places, consider creating an external script instead.

Pictures

In addition to text files and scripts, you can also put picture files in the “Includes” folder. Although this is not necessary, you can avoid clutter by putting pictures in a separate subfolder, such as “Pictures”. You will then find the available pictures in the “Pictures” submenu of the {…} pop-up.

When you insert a picture in an expansion, you will not see the picture itself but rather a picture marker like this:

```
{Pictures/CompanyLogo.png}
```

When the expansion takes place, Typinator loads the picture and pastes it in the correct location.

You could also use expansions of the type “Formatted Text” to mix text with pictures. Inserting pictures via the the Includes mechanism differs from this approach in the following aspects:

- Included pictures save space because you can reuse the same picture in many expansions.
- You can easily exchange pictures by replacing the picture files with new versions. For example, you may want to include banner ads in mail messages. When you start a new campaign, simply put the new banner into the “Pictures” folder, and it will get inserted in all expansions that use this picture marker.
- You can use picture markers in plain text expansions. This may be convenient when you do not want to explicitly specify the font and style of the text but rather want the inserted text to match the font of the target document.

Sets

Typinator lets you arrange replacement rules in multiple sets. With sets, you can …

- group your abbreviations in categories, such as letter templates, typing corrections, mail signatures, code snippets, Unix commands, etc.
- import and export sets for sharing them with others or moving them to another Mac.
- define which abbreviations shall be active in which applications.
- temporarily disable groups of abbreviations.

Types of sets

Typinator distinguishes between two types of sets, which appear under separate headings in the set list:

- **Abbreviation sets** contain simple rules that translate short sequences of characters (the "abbreviations") into a longer text fragments or pictures (the "expansions").
- **Regular expression sets** contain rules that use a pattern description language to define which text input should trigger expansions. Regular expressions are a powerful but more complex concept. They are described in detail in a separate section further below.

Creating sets

To create a new set, click the "+" button below the left list and select the type of set (abbreviations or regular expressions). To rename a set, double-click its name in the list.

You can also choose "New Predefined Set" to add one of Typinator's built-in sets that you can use out of the box. For more information on *predefined sets*, see the corresponding section further below.

Assigning abbreviations to sets

To move an abbreviation to another set, select it in the abbreviation list and drag it onto the desired set in the set list. To move multiple abbreviations, command-click to extend the selection by single abbreviations or shift-click to select a range of abbreviations. Then click in any selected abbreviation and drag the abbreviations to the destination set.

You can also create copies of abbreviations in another set by pressing the option (alt) key when you drop them onto the destination set.

Activating sets for specific applications

Once you have a few sets, you can tell Typinator which sets should be enabled or disabled in which applications. For example, you could enable a set with mail signatures only in Mail, or define that your code snippets expand only when you work with Xcode.

To define the assignment of sets to applications, click the button with the application icon below the set list.

If you open the application settings window for the first time, the list at the left will contain only a single item "All Applications". You can use this entry to globally enable or disable sets.

To add applications with special requirements to the list, drag them from the Finder into the list, or click the "+" button to add one of the currently running applications. The item "All Applications" will then change to "All Other Applications". It represents all applications that are not explicitly listed.

To define which sets should be active in a specific application, select the application in the list and check the sets that should be considered in this application.

Notes:

- To quickly turn all sets on or off for an application, select the application in the list and click the *All* or *None* buttons below the set list.
- Disabling **all** sets for a specific application tells Typinator that you do not want it to expand anything in this application. You can use this setting to disable Typinator for individual applications.
- If you want to disable Typinator temporarily for all applications, you can do so by selecting "Pause Expansions" from Typinator's menu. This will deactivate Typinator entirely and a pause symbol appears in the menu bar until you select "Resume Expansions".
- You can use the same feature to control in which applications the automatic "DOuble CAps" correction should take place. For example, you may want to turn this set off in Xcode to avoid unwanted changes when you write source code.

Disabling sets

You can disable sets by turning the checkbox in the set list off. This can be quite handy when you need different abbreviations in different projects, but use the same applications in all projects. Just define separate sets for your projects, and toggle the checkboxes as needed to quickly switch from one project to another one.

Note that the checkboxes in the set list act as "master switches". They take precedence over the application-specific set assignments described in the previous section.

The checkboxes control whether the items in a set expand while you type, but they do not affect the Quick Search function described further below.

Conflicts between sets

Abbreviations in one set may overlap with abbreviations in other sets. For example, you could have the same abbreviation "bzz" with different expansions in two sets A and B. When sets A and B are active in different applications, the expansion of "bzz" depends on the current application.

When both A and B are active in a certain application, Typinator uses a simple rule to resolve the conflict: Sets at the top of the list take precedence. When A appears above B in the set list, "bzz" within A gets expanded, and the corresponding entry in B will be ignored. To change the order of sets, simply drag them up or down in the set list.

If an abbreviation may be disabled by another one in an earlier set, Typinator displays a warning symbol in the abbreviation list:

When you select the abbreviation, a message in red tells you about the potential conflict:

Note: Typinator does not report conflicts with regular expressions. Regular expressions are always checked in the listed order, and only if the typed text does not match any of the defined abbreviations. When a regular expression matches the typed text, subsequent regular expressions are not checked.

Exporting and importing sets

To move sets from one Mac to another, use the Import and Export commands in Typinator's menu. You can also export sets by dragging them from the set list to the Finder. In the opposite direction, drag a set file from the Finder into the set list.

You can also import snippet files from Textpander, TextExpander and TypeIt4Me, "autocorrection lists" (ACL files) of Microsoft Office, as well as the built-in text substitution rules of macOS:

- TextExpander: Import the file Settings.textexpandersettings (in older versions, the file was called Settings.textexpander) in
  ~/Library/Application Support/TextExpander
  within your home folder[1].
- TypeIt4Me (version 3.0 or newer): Import your clippings file (with the extension "typeit4me").
- aText: Export your snippet collection with aText's File/Save command and select "aText" as the file format. Then import the created "atext" file in Typinator.
- Microsoft ACL files: The "ACL" files that contain autocorrections and other text replacements can be found in the "~/Library/Preferences/Microsoft" folder[1] inside your home folder. The "Microsoft" folder contains a subfolder with the name of the current version (such as "Office 2011"), which in turn contains the ACL files.
- macOS text substitution rules: Open System Preferences / Keyboard / Text, click any item in the Replace/With list, type ⌘A to select all items, then drag the selection to the desktop. This creates a file named "Text Substitutions.plist", which you can import in Typinator.

Exporting and importing text files

When you use the Export command in Typinator's menu to export a set, you can choose the file format that should be created:

- **Typinator Set** creates a set file in Typinator's native format (with the extension .tyset), which you can later import again by double-clicking the file in the Finder.

---

[1] On macOS 10.7 (Lion) and newer, the Library folder may be invisible. To quickly access it, open the Finder's "Go" menu. Then press the alt key (⌥), and the command "Library" will appear.

- **Tab-delimited Text** creates a text file (extension .txt) with a separate line for each item. Each line starts with the abbreviation, followed by a tab character and the expansion:

  abbr<tab>expansion

  If an expansion contains tab, return (CR, carriage return) or newline (LF, line feed) characters, Typinator replaces them with special character sequences, such as \t, \r and \n. Typinator actually uses representations that are not present in any expansion. The first three lines in the text file indicate which sequences were used for these special characters, such as:

  tab=\t
  return=[cr]
  newline=\n

- **Comma-separated Values** (extension .csv) creates a text file in which the abbreviations and expansions are separated with commas. CSV is a popular exchange format that is used by many spreadsheet and database applications. For example, you can import such CSV files into Numbers, FileMaker, OpenOffice, and Excel. Typinator uses the informal specification of the CSV format, as described in the RFC 4180 memo.

Typinator exports text files in the UTF-8 Unicode encoding. Many applications import UTF-8 files correctly, but some applications require that you explicitly specify the encoding when you import CSV files. Unfortunately, Excel does not handle CSV files in UTF-8 format correctly, so some characters (such as accented letters) may get lost.

When you check the option “Include case and whole word options”, Typinator adds two additional columns to both tab-delimited text and CSV files. The third column contains a single word that describes the case options that are available in the Case pop-up menu:

- *match:* Case must match
- *ignore:* Case does not matter
- *apply:* Case affects expansion

The fourth column contains the word *whole* when the “Whole word” option is turned on for an item, and *part* if the option is off.

Typinator can import both tab-delimited text and CSV files. When an imported text file contains the *tab=x, return=x* and *newline=x* entries in the first few lines, Typinator translates these sequences into the respective special characters. When text or CSV files contain the keywords *match, ignore, apply, word* or *part* in the third and fourth columns, Typinator uses this information to set the corresponding attributes of the imported items.

Replacing existing sets during import

If you regularly import sets from other sources (such as a list of articles from a spreadsheet or database), you may wish to replace the previous version of the set with the new version. You can do this by adding "-rep" (for "replace") to the file name extension before importing a file in Typinator. Typinator recognizes the following special extensions:

tyset-rep
txt-rep
csv-rep

When you import such a file and a set with the same name already exists, Typinator replaces the original set with the new version. If no set with the same name exists, Typinator just adds the new set.

To prevent loss of data, Typinator does not just delete or overwrite the old version, but rather moves the old version to the trash. If you have inadvertently replaced a set, you can restore the original by recovering it from the trash.

Predefined sets

Typinator comes with a couple of useful sets that you can install by clicking the "Predefined Sets" icon in the toolbar or by selecting "New Predefined Set" after clicking the "+" button below the set list

The list of predefined sets contains a few sets that help you to automatically correct frequent typing errors in English, German and French. These sets have been set up such that they can be used together. If you need to switch between English, German and French, you can therefore add all three languages to your set list.

Typinator also contains an autocorrection set created by the TidBITS publisher Adam Engst and Micah Alpern. The "TidBITS AutoCorrection" set with more than 2300 entries is an excellent addition when you write primarily in English, but it may create a few unwanted replacements when you write in other languages.

You will also find a few other sample sets in this list, such as the "DOuble CAps" set or a set with a single rule that removes all formatting from text in the clipboard. Some of these are usable "out of the box", others may require customization. You can find even more examples our "Download Extras" web page.

Set-specific settings

Click the button with the blue "i" symbol below the set list to access a few set-specific settings. In the info window, you can define a few characters that should be used as common **prefixes and suffixes** for all characters within the same set, without editing hundreds of abbreviations individually. For example, you will notice that all abbreviations in the predefined HTML set start with "<<". If you prefer a different prefix (such as "h-") or a suffix or both, you can make the change for all HTML snippets in a few seconds.

In the **Count replacements as...** section, you can specify the type of the entries in a set. This setting affects the feedback sound and the usage statistics that you see in Typinator's About window.

The **Feedback sound** option lets you override or disable the sound played by Typinator. Select "Default" to use the sounds specified in the *Preferences* window (see below).

The section **Include in Quick Search** allows you to define under which circumstances which parts of a set should be considered when you type search terms in the Quick Search field. For more information, see the section on *Quick Search*.

In the **Default Options** section, you can specify the initial case and whole-word settings for new abbreviations that you add to this set.

Finally, there is a **Notes** field where you can describe the purpose and usage of a set. This is particularly convenient when you create a set that you want to share with other Typinator users.

Publications and subscriptions

Subscriptions

Starting with Typinator 7.0, you can subscribe to sets that were created by someone else. A regular import of a set creates a local copy of your computer that you can now modify independently from the original. In contrast, subscribed sets are linked to the original set, and they are automatically updated whenever the original changes.

Typically, subscriptions come from a remote source, where someone has made a set publicly available on the Internet. If you know the URL of such a set, click the "+" button below the set list, select "Subscribe to Set via URL…", enter the URL, and Typinator copies the set from the external source to your local collection. Such a subscribed set is basically read-only (with some exceptions, described below). When the external source changes, Typinator automatically updates your local copy. To see the status of a subscription, right-click or ctrl-click the set in the list and select "Subscription Info…". To manually trigger an update, select "Update Subscription". The Sets submenu of the Action toolbar item also contains an item that lets you update all your subscriptions at once.

Subscriptions are great for auto-corrections. We therefore provide all our predefined auto-correction sets as subscriptions. Click the "+" button below the set list and select "New Predefined Set…". All *AutoCorrection* sets as well as the "Product Names" set are provided as subscriptions, as indicated by the icons in the list. This means that you will immediately benefit from improvements that we make to these sets.

You cannot edit the contents of subscribed sets, as this could create conflicts when a new version becomes available. You can, however, change the name of a subscription set, enable or disable it, change the expansion sound, and you can disable individual items. For example, one of the misspelled words in an AutoCorrection set may actually be correct in a certain context. To prevent unwanted correction when you type that word, just select it in the list and hit the delete key. The item will appear in strike-through style to indicate that this rule is now ineffective. It will also remain ineffective when the subscription is updated. To re-enable an ineffective item, select it and hit the delete key again.

Publications

To create a set to which other users can subscribe, you must export it in a special format (file extension “typubset”). You can do this from the “Export…” command in the “Sets” submenu of the Action toolbar item (or via the context menu with a ctrl-click or right-click on a set in the list). Select the export format “Publication” and specify the suggested update interval. Then put the file in a location from which other users can subscribe to the set. If you have your own web site, you can publish your Typinator sets there. If you use Dropbox, you can put your publication file into your Dropbox folder. If you use Dropbox also as the storage location for your Typinator sets (see the chapter about the *Sets folder* below), do not save your publication inside your sets folder. We recommend that you use either the “Public” folder or create a dedicated publication folder on the top level of your Dropbox. When Typinator has exported the set, ctrl-click or right-click the “typubset” file and select “Copy Dropbox Link” (or “Copy Public Link”, if your file is located in the “Public” folder). Send that URL to your friends and tell them to use Typinator's “Subscribe to Set via URL…” command to subscribe to your set. When you wish to update your set, export it again and replace the previous “typubset” file with the new version.

You can also use Typinator's “automatic publishing” function to export a set immediately whenever you make changes. To use this feature, ctrl-click or right-click a set and select “Automatic Publishing…”. You can then select the target location (for example, in your Dropbox). If you already know the final URL (from which your subscribers will load the set), we recommend that you enter that under “Download updates from a specific URL”.

You can also create shell scripts that are automatically executed when a file has been published. Such scripts are invoked with the path of the just created “typubset” file as the only parameter. For example, the following simple script copies a published set to a different directory:

```
#!/bin/bash
cp "$1" ~/publications/
```

You can use such scripts to automate further steps that may be required to put the exported file in a publicly reachable location. For example, you can use *scp* for copying it to a remote web server, or you can use *git* to automatically upload the file to GitHub.

Status icons

When you look at the left column of the set list, you will notice that every set has an icon that displays its status:

 local set, neither subscribed nor published automatically

 set that is automatically published after every change

 subscribed set with active updating

 subscribed set with paused updating

 subscribed set for which the most recent update attempt failed

To see more detailed information about the subscription or publication status of a set, right-click or ctrl-click a set and select “Subscription Info…” or “Automatic Publishing…”.

Regular Expressions

Regular expressions are a flexible and powerful notation for describing text patterns. Typinator's implementation is based on the ICU package (see the ICU User's Guide for more information). With regular expressions, you can describe patterns like "a digit, followed by the letter X" or "a date in the form YYYY-MM-DD" or "any text enclosed in quotes". Writing regular expressions is not trivial, but Typinator 6.0 comes with a built-in cheat sheet (select Help > Regular Expressions… from Typinator's Action menu).

Regular expressions are most powerful when you combine them with scripts or Typinator's built-in functions. For example, you can create a rule with the regular expression

```
([0-9]+)x"(.+)"
```

This means "a decimal number, followed by the letter 'x', then some text in quotes". For example, the text

```
50x"+"
```

matches this pattern. The number and the quoted text are enclosed in parentheses, which makes these parts available as placeholders $1 and $2 inside the expansion. Now you can use the expansion

```
{/Repeat $1/$2/}
```

to invoke the built-in Repeat function, which repeats $2 (the quoted text) as many times as specified by $1 (the number before the "x"). As a result, the above pattern creates 50 plus signs in a row:

```
++++++++++++++++++++++++++++++++++++++++++++++++++
```

When a regular expression contains a repetition (expressed by * or +), it normally matches the longest possible sequence. This can sometimes lead to unexpected results because of the way how Typinator works. Whenever you type a character, Typinator checks whether the text you have recently typed matches one of the defined patterns. As soon as this is the case, Typinator replaces the matched text with the corresponding expansion. When a regular expression ends with a repetition, the first typed occurrence will already trigger the expansion. Example: Although the regular expression [0-9]+ normally tries to match the longest possible sequence of digits, typing a single digit will already trigger the expansion, since Typinator cannot predict what you will type next. There are two possible solutions:

- When the repeated pattern ends with letters or digits, you can turn on the "Whole word" option. Typinator will then try a match only when the next typed character is neither a letter nor a digit.
- You can add a terminator character to the regular expression, such as:
  ```
  ([0-9]+)([^0-9])
  ```
  This expression matches the longest possible number as $1, but it also requires a non-digit as a terminator character, which is captured as $2. In the expansion, you can then process the number and add $2 at the end of the expansion to make sure that the terminating character does not get lost.

Beyond the “whole word” setting, the case options (must match, does not matter, affects expansion) also apply to regular expression rules. In addition to these options, two further options are available with regular expressions:

- **Combine with previous expansion:** When this option is turned on, the result of a previous expansion (no matter if triggered by an abbreviation or a regular expression) may be part of a regular expression match. For example, a previous expansion my have produced an expansion that ends with a decimal number. A regular expression can then “pick up” that decimal number and include it as part of the recognized pattern.
- **Combine with following abbreviation:** When this option is turned on, the result of the expansion is allowed as part of a subsequent abbreviation (or other regular expression). For example, an expansion could end with characters that occur as prefixes of other abbreviations.

A cool application of these options is the subsequent modification of previous expansions: Assume that someone uses two mail addresses, peter@first.com and p.green@second.com. Peter could now define “@@” as the abbreviation for his primary address “peter@first.com”. In addition to that, he can create a regular expression “peter@first\.com@”[1] with the expansion “p.green@second.com” and the option “Combine with previous expansion”. The effect is that Peter can now type @@ to produce the primary address, and typing a third @ replaces it with the secondary address.

Rules based on regular expressions are treated specially by Typinator. You will notice that the set list is divided in two sections “ABBREVIATIONS” and “REGULAR EXPRESSIONS”. When you type text, Typinator first checks the contents of the first section whether you just completed an abbreviation. If not, it checks whether the typed text matches a regular expression in the second section. Regular expressions are always checked sequentially. To define the order in which they are tested, you can use drag&drop to change the order of the regular expression sets, and you can also rearrange regular expression rules inside sets.

Another special property of regular expressions is that they are checked at any text position, even in the middle of words. For example, you can define a regular expression “o/” that with the expansion “ø”. You could then type “so/ster” to write the Danish word “søster”. If you explicitly want a regular expression to match only letters at the beginning of a word, start the regular expression with the word boundary anchor “\b”. For example, the expression

```
\bn[0-9]
```

matches the letter “n” followed by a digit, but only when the preceding character was not a letter.

To make the formulation of regular expressions easier, you can define “macros” for certain complex patterns in regular expressions. To do so, open the “Set Info” window and define one or more short forms along with their meaning in the Notes field. Such definitions must be written on single lines each and have the form

```
shortform=pattern;
```

The definition must end with a semicolon. Do not enter spaces before and after the equal sign. You can then use “shortform” in a regular expression wherever you would otherwise

[1] The backslash is required to represent the period, which would otherwise have special meaning in the regular expression: “.” means “any character”, but “\.” stands for a literal period.

need to enter the complex pattern. An advantage of such a macro is that you can use it in multiple rules, and you can modify the pattern in one central location.

You can see an example in the predefined set “Inline Calculation”, which contains the following macro definition:

FORMULA=([-+*/^%!°()0-9,.a-zπ:]+);

“FORMULA” defines that any sequence of the enclosed characters is treated as a computation. The macro is then used in the rule

FORMULA=\?

You could also add another rule with the pattern

compute FORMULA!

With the same expansion {{#$1}}, you could then type

compute 123.4/567.8!

to get the result

0.2173300457908

Note that Typinator’s implementation of regular expressions is based on a system component that requires macOS 10.7 or newer. Typinator 6 still works with Mac macOS 10.6, but the regular expressions (and other features based on regular expressions, such as automatic correction of DOuble CApitals) do not work on systems older than 10.7.

Quick Search

The number of sets and snippets that you can use with Typinator is virtually unlimited. But the more items you add, the more difficult it is to invent and remember new abbreviations. Fortunately, you can use Typinator’s Quick Search feature to look up items and insert the expansions in your documents with a few keystrokes.

To start a search, select Quick Search from Typinator’s menu or type the corresponding keyboard shortcut (which you can change in the preferences, as described further below). A search field will appear at the top of the main screen, partly overlapping the menu bar:

Now type a few words or parts of words that you are looking for. For example, if you want to find a mail template for offering new articles, just type “off art”, and Typinator will display a list of all snippets that contain both search terms.

To search for picture expansions, you can also type “picture” as the search term.

To insert an item in the current document, double-click it or select it with the up/down cursor keys, then hit the return or enter keys. When you press the command or ctrl keys when you activate an item, Typinator opens its window and selects the item, so you can quickly make some changes.

Typinator remembers the items that you pick from the result list. When you look for similar terms again, these items will appear at the top of the list with a green triangle in the first

column. Typinator also performs some ranking; good matches always appear further to the top. When you enter multiple words, only those items that contain all these words will match. You can therefore easily reduce a large set of matches by adding more words. To search for an exact phrase (such as “please tell me”), enclose the phrase in double quote characters. Typinator will then search for the phrase exactly as written instead of the individual words “please”, “tell”, and “me”.

Hint: You can list the previously picked snippets by opening the Quick Search field and typing a single space character. Typinator will then list all the recently used snippets in reverse order, with the most recent item at the top. You can then enter additional terms to search within the list of recently used items.

You can control which sets will be included in a Quick Search. Select a set in the Typinator window and click the blue “i” button below the set list. You can then define whether the abbreviation or the expansion or both will be included in the Quick Search. Turn both checkboxes off if you do not want a set to be included in a Quick Search. For example, both checkboxes are off for the built-in autocorrection sets, as you will hardly ever want to search for misspelled words.

You can also specify a keyword that must be present for searching a set. For example, you could define “p” as the search keyword for a set containing a price list. To search the price list, you must type the keyword “p” in the search field, along with the words that you are looking for (e.g., “p book” for all items in the price list that contain “book”). You may want to use search keywords for long sets that would otherwise deliver thousands of matches when you are actually looking for something else. Keywords enable you to turn the search in specific sets on and off as needed. If you occasionally wish to ignore all set keywords, start your search with two asterisks: “** book” searches for the word “book” in *all* sets.

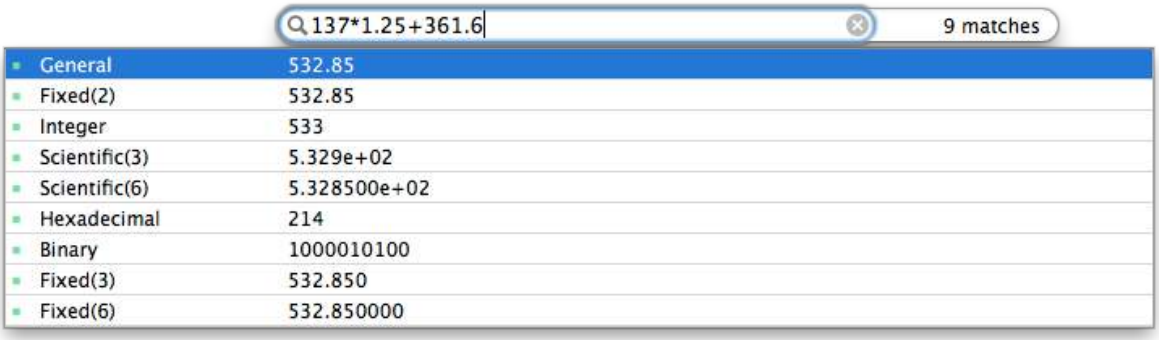
If you have a large set that you plan to use exclusively with the Quick Search function, you can turn off the checkbox in the set list to exclude the items in this set from expansion by typing. Once you have done this, you can use the Abbreviation field of the snippets for additional search terms that do not occur within the expansion. For example, you could include “xm” in the abbreviation field of all items that have to do with special offers for Christmas. Just make sure that the abbreviation is enabled for the Quick Search in the set’s settings, and then you can search for “xm box” to find all Christmas offers that also contain the word “box”.

As an alternative to invoking Quick Search and then entering a search string, you can use Typinator’s “Suggest Abbreviation” function. To use this feature, you first need to define a keyboard shortcut: Open Typinator’s preferences, select the “Suggest Abbreviation” shortcut field in the Activation tab and type the key combination of your choice. Once you have done this, type part of an abbreviation or a typical word that is contained in the desired expansion. Then type the “Suggest Abbreviation” shortcut, and Typinator shows a list of matching items. Select the desired item, and Typinator replaces the search word with the expansion. This mode of operation is particularly convenient when you have multiple abbreviations that start with the same letters: Just type a few letters and then let Typinator suggest potential continuations.

You can also use the Quick Search field to search for recently used abbreviations. This search function is most useful when Typinator made an unexpected expansion, and you want to find out which replacement rule was responsible for that. To find the most recent expansions, enter one of the phrases “last”, “recent”, “latest”, or “last 3” in the Quick Search field. This type of search also works in the search field in the Typinator window.

Pocket calculator

The Quick Search field also serves as a simple, yet powerful pocket calculator. Just enter a simple calculation, such as “137*1.25+341.6”, and the result of the calculation appears in the list below in multiple formats:



Just pick the desired item to insert the result in the corresponding format. And when you use one format, Typinator remembers your choice and moves this format to the top of the list, so you can access it faster the next time.

The calculator accepts the usual operators: +, –, *, /, % (for modulus), and ^ for raising a value to an exponent. You can also use the notation x! for computing the factorial of x, and |x| yields the absolute value of x.

The calculator supports relational operators (=, ≠, <, >, ≤, ≥) and logical operators (“|” for *or*, “&” for *and*, the unary prefix “!” for *not*). Alternatives for the relational operators are “!=” and “<>” for “≠” as well as “<=” and “>=” for “≤” and “≥”, resp. When you use numeric values as operands of logical operators, the value 0 counts as *false*; all other values are treated as *true*.

Calculations may contain the functions abs, exp, gamma, int, if, min, max, round, ln, log/log10, ld/log2, sign, sqrt, sin, cos, tan, cot, asin/arcsin, acos/arccos, atan/arctan, acot/arccot, sinh, cosh, tanh, coth, asinh/arcsinh, acosh/arccosh, atanh/arctanh, acoth/arccoth (names separated with slashes are alternative names for the same function).

The trigonometric functions expect the arguments in radians. To make calculations with degrees, you can use the ° symbol as an operator, as in *sin*(60°).

You can also include the constants *pi* or *e* in your calculations. For example, 4*pi yields the circumference of a circle with a diameter of 4 units. The logical constants *true* or *yes* stand for the numeric value 1; *false* and *no* represent the value 0.

Some functions take more than one argument. In this case, you must separate arguments with semicolons instead of commas (because the comma is used as a decimal separator in some countries, as in 1,23). The *min* and *max* functions accept an arbitrary number of arguments and return the smallest or largest value. For example, *max*(12; 42; 9) returns the value 42. The *round* function may have a second argument that determines the number of digits afer the decimal point. For example, *round*(*pi*) returns 3, but *round*(*pi*; 3) returns 3.142.

The *if* function expects three arguments. The first argument is a boolean value; if it is true, the second argument is taken as the result, otherwise the third argument. For example, if(x>=0; sqrt(x); 0) returns the square root of x for positive numbers and zero for negative numbers.

The calculator even supports variables. For example, you can write “price=12.9*1.2” to assign the result to the variable “price”. When a calculation contains an assignment, the format list will also contain an item “assign” that only performs the assignment without inserting the result in the current document. After an assignment, you can use the variable “price” in subsequent calculations. For example, “150*price” yields the total price of 150 items.

You can also convert decimal numbers to hexadecimal or vice versa. To enter a hexadecimal number, use a leading dollar sign or the prefix 0x: $2A or 0x2A.

Quickly defining new items

The Typinator menu contains two commands for quickly defining new items:

- **New Item from Selection...** uses the current selection as the basis for the new item.
- **New Item from Clipboard...** uses the current contents of the clipboard.

Actually, the first command first copies the current selection to the clipboard and then performs the same operation as “New Item from Clipboard…”.

Both commands display a window in which you can specify further information required for creating a new item. Depending on the contents of the selection or clipboard, Typinator may offer up to three choices:

- **Boilerplate:** The selection or clipboard is taken as the expansion, and you can tell Typinator which abbreviation you want to use. If the text contains formatting information, you can turn the “formatted” checkbox on or off, depending on whether you want to create a plain text or formatted text expansion.
- **AutoCorrection:** If the selection or clipboard contains only one or two words, Typinator uses that as mistyped original text and lets you enter the correction. Typinator also offers suggestions in the languages that you have defined in System Preferences / Language & Text.
- **Picture:** If the selection or clipboard contains a picture, Typinator offers to create a picture expansion, and you can specify the desired abbreviation.

In all cases, you can specify to which set the new item should be added. Typinator remembers your choice separately for the three types of items. If you specify that newly defined pictures should be added to a set named “Artwork”, Typinator will suggest the same set for

the next picture that you add. Likewise, you can specify different sets for boilerplate text and spelling corrections.

To complete the definition of an item, click the Add button. Typinator will then add the new item to the specified set, and the new abbreviation will take effect immediately. When you create a new AutoCorrection item, Typinator also copies the correction to the clipboard. If the original misspelled word is still selected in the current document, just type command-V to replace it with the correction.

The “whole word” and case attributes of the newly defined item are taken from the default settings of the destination set (see the section *Set-specific settings*). In most cases, this will be exactly what you want. If you want to check or modify the attributes of the new item, click the button “Add and Edit in Typinator”. This will open the Typinator window, so you can make further modifications to the newly created item.

Hint: You can assign keyboard shortcuts to the “New Item from…” commands in the Preferences window. See the following section for details.

Preferences

Click the Preferences icon in the toolbar to change the following settings:

Activation:

- **Open window when Typinator starts** opens the list of abbreviations whenever you launch Typinator. If you want Typinator to start silently (especially when it starts automatically at login), turn this checkbox off. To open the list of abbreviations, just click the Typinator icon in the menu bar.
- **Automatically start Typinator at login** adds Typinator to your list of login items, so Typinator will automatically launch whenever you start up your Mac.
- **Show Typinator in menu bar** controls whether the Typinator icon should be shown in the right-hand section of the menu bar. When you choose not to show the icon, you need to double-click Typinator in the Finder in order to open its window. As a quick alternative, we suggest that you drag Typinator’s application icon into the dock. You can then open Typinator’s window with a single click on the dock icon.
- The section **Keyboard shortcuts** lets you assign keyboard combinations for quickly opening Typinator’s window, for temporarily pausing expansions, for starting a Quick Search, and for quickly creating new items.

Expansion:

- The **Enable quick expansion** option allows Typinator to use a new expansion technique that is available on macOS 10.5.5 or newer. This technique enables Typinator to expand most plain text snippets much faster than before, while avoiding subtle compatibility issues with certain applications and third-party utilities. The quick expansion option is therefore the preferred choice and should be enabled when possible. If you encounter incorrect expansions in some applications, try turning the quick expansion option off. If

this helps, please also contact typinator-support@ergonis.com and tell us about the problem.

- In the **Default feedback sounds** section, you can define the preferred sounds for expansions and corrections separately. You can override this setting in individual sets. Feedback sounds are great as a confirmation that Typinator recognized typed abbreviations and for making sure that autocorrections produce the desired results.
- The **Sets folder** part allows you to specify where Typinator should store your abbreviations. See the following section for more information about this feature.

Updates:

- **Check for available updates** lets you define in which intervals Typinator shall check our servers for the availability of new versions, so you will always remain up-to-date.

Note that the section "Updates" will not be available when you use the site version of Typinator. This is not necessary because we send separate update notifications to site customers.

The Sets folder

Typinator stores your sets in a folder on your hard disk. Starting with Typinator 3.6, you can tell Typinator where it should store your sets. The default location is the "~/Library/Application Support/Typinator/Sets" folder[1] inside your home folder. To change the location, open Typinator's Preferences and click the Change… button in the "Sets folder" section of the "Expansion" tab.

Typinator will restrict the folder that you can select to empty folders and folders that have already been used for Typinator sets before. This restriction helps to avoid potential loss of data in folders that already contain other documents. When you select an empty folder, Typinator will offer to copy the contents of your current sets folder to the new location. Note that you can use the "New Folder" button in the folder selection window to quickly create an empty folder. Of course, the new folder can have any name, but we recommend that you call it "Typinator Sets" or something like that, so you can easily identify this special folder when you see it in the Finder.

Why should you care about where exactly Typinator stores your abbreviations? There are a couple of good reasons for storing the sets in certain places:

- If you regularly synchronize or backup your Documents folder, you can put your sets folder inside the Documents folder. Whenever you run your synchronization or backup software, it will take care of all your Typinator sets, too.
- If you sometimes need different sets for a certain project, you can create a second sets folder and switch back and forth between sets by selecting the desired sets folder in the Preferences window.

---

[1] On macOS 10.7 (Lion) and newer, the Library folder is invisible. To quickly access it, open the Finder's "Go" menu. Then press the alt key (⌥), and the command "Library" will appear.

- You can put your sets folder inside your Dropbox[1] folder to synchronize it among multiple Macs. To quickly move all your sets to your Dropbox, click the Change… button in the Expansion section of Typinator's Preferences, navigate to your Dropbox folder, create and choose a new folder "Typinator Sets", and finally let Typinator copy all your sets to the new location when it offers to do so. On all your other Macs, just select the "Typinator Sets" folder that you have created on the first Mac.

Note that the "Includes" folder (which contains your includable text files and scripts) is also located inside the Sets folder. If you use, for example, Dropbox to synchronize all your sets among multiple computers, this means that all text files and scripts that you use in your expansions will also get synchronized.

Controlling Typinator via scripts

Starting with version 6.7, you can control many aspects of Typinator via AppleScript or JavaScript.

If you are familiar with the scripting technologies of macOS, most of Typinator's scripting features should be rather straightforward to use. You can, for example, pause or resume Typinator, query or modify attributes of rules and sets, or trigger an expansion as if the corresponding abbreviation had been typed.

To find more about Typinator's scripting capabilities, take a look at Typinator's scripting suite in Script Editor's Library window.

Here is a simple example that shows some of Typinator's scripting functions in action:

```
tell application "Typinator"
	if exists rule set "Scripting" then
		if enabled of rule set "Scripting" then
			expand string "scx"
			pause expansions
		end if
	end if
end tell
```

This script checks whether the rule set "Scripting" exists and is enabled. If so, it triggers expansion of the abbreviation "scx" and finally puts Typinator in pause mode.

Exceptions and troubleshooting

If typing an abbreviation does not work, take a look at the Typinator icon in the menu bar. It may appear in one of the following variants:

The dark gray regular icon appears when Typinator is active. Abbreviations should work normally in this case.

The white variant appears while Typinator's window is open and in the front. Typinator does not expand typed abbreviations in this situation to avoid circular expansions while you are in the process of defining abbreviations.

[1] For more information, see www.dropbox.com.

The pause icon appears when you have manually paused Typinator (by selecting “Pause Typinator” from the menu or typing the corresponding shortcut. To enable expansions, type the shortcut again or select “Resume Expansions” from the menu.

The red “X” appears when Typinator does not have the required permissions to watch your keystrokes and insert expansions in your documents. For assistance, open the Typinator window and click the button at the bottom of the window.

A gray icon with a slashed circle appears for applications that are incompatible with Typinator. This is, for example, the case for applications that control a remote computer (such as Screen Sharing or Microsoft's Remote Desktop Connection) or runs a virtual computer (such as Parallels Desktop or VirtualBox). Since expansions carry the risk of interfering with the other (remote or virtual) computer, Typinator disables expansions in this situation.

An icon with two bullets appears when you type in a password field. Typinator uses a keyboard monitor function of macOS to watch what you type. macOS turns this monitor off in password fields for security reasons. Typinator then does not “see” what you type and therefore cannot expand abbreviations.

The keyboard monitor can also be turned off by other applications. For example, this happens when you activate the “Secure Keyboard Entry” option in the Terminal application. There are also some third-party utilities that disable the keyboard monitor when they should not. For up-to-date information about such cases, see the FAQ page on our web server at

https://www.ergonis.com/products/typinator/faq.html.

An icon with a blue underline means that the currently active keyboard input source (for example, for Chinese or Japanese) prevents Typinator from correctly detecting typed abbreviations. To avoid unwanted results, Typinator disables expansions in this situation.

Registering Typinator

If you use Typinator under the terms of a site license, the following information does not apply. When you click the Registration button, Typinator displays information about the site license.

Typinator is distributed on a “try before you buy” basis. Unless you have a license key, you can use only the first five abbreviations in the list.

You can order a license key directly from our online store:

https://www.ergonis.com/store/

You can also click the Registration icon in the toolbar and then use the “Order License Key” button to visit our online store.

Once you have received your license key, click the Registration icon in the toolbar and enter the license key. This will immediately remove the demo limitation and let you use all defined abbreviations.

Uninstalling Typinator

To uninstall Typinator permanently, perform these steps:

- Open the Typinator window.
- Click the Preferences icon in the toolbar and make sure that “Automatically start Typinator at login” is disabled.
- Quit Typinator.
- Move the Typinator application to the trash.

Reporting problems

If you have any further questions or wish to report a problem, please send an e-mail message to <typinator-support@ergonis.com> or simply use the Contact button in Typinator's toolbar. We also would like to hear from you if you just want to tell us that you like Typinator or you would like to suggest an improvement or additional feature. In any case, please include the following information in your reports:

- Macintosh model
- macOS version
- version number of Typinator

If you wish to report a problem where Typinator does not work correctly with a particular application, please also include

- the version number of the application,
- a detailed description of the situation, so we can reproduce the problem (in particular, tell us what expansion you try to insert with which abbreviation).

Thank you for your feedback. We are looking forward to hearing from you.

Known issues

Typinator uses a couple of sophisticated techniques to monitor your keystrokes and insert the expansion in the current application. Although these techniques work well for the vast majority of applications, you may come across one of the following deficiencies:

- Text-only applications (such as BBEdit or Terminal) will not be able to insert picture expansions, and formatted text will lose embedded pictures as well as specific font and style settings. This is not a bug but rather a limitation of these applications.
- If the expansion text contains Unicode characters, strange characters will appear instead in some older applications that do not fully support Unicode (such as AppleWorks or MS Word versions older than Word 2004).
- Formatted text expansions with embedded images will work only in those applications that support this clipboard format. If an application does not support this format, the result typically is just the text, and the embedded pictures will get lost. This happens, for examples, in all components of MS Office.

Compatibility with older versions

Typinator 6.0 introduced new formats that are not completely backwards-compatible with older versions of Typinator. When a set that was created or modified with Typinator 6 is used by an older version of Typinator, the following features will not work:

- Regular expression sets will be treated as classic abbreviation sets. Since regular expressions are typically complex patterns, it is very unlikely that you type such "abbreviations" by accident.
- Descriptions added to expansions (using the new blue speech bubble button) will not appear. Older versions of Typinator will not delete these descriptions when you just *use* the corresponding items. But when you use an older Typinator version to *modify* any item in such a set, the whole set will be written to disk, and the added descriptions will get lost.
- HTML expansions are treated as plain text items with the comment

  ```
  <!-- HTML Expansion-->
  ```

  in the first line. To ensure compatibility with Typinator 6, just leave this comment intact.

These differences are important only when you transfer sets between computers that run different versions of Typinator. When you use Dropbox to synchronize your sets, we therefore recommend that you upgrade to Typinator 6 on all computers that use the shared sets.

