

# in-dexter

## An index package for Typst

Version 0.4.2 (9.6.2024)

Rolf Bremer, Jutta Klebe

Contributors: @epsilonhalbe, @jewelpit, @sbatial, @lukasjuhrich

## Content

1 Sample Document to Demonstrate the in-dexter package .....	1
1.1 Importing the Package .....	1
1.2 Marking of Entries .....	2
1.2.1 Nested entries .....	2
1.2.1.1 LaTeX Style index grouping .....	2
1.2.2 Entries with display .....	2
1.2.3 Advanced entries .....	2
1.2.3.1 Symbols .....	3
1.2.3.2 Formatting Entries .....	3
1.3 The Index Page .....	3
1.3.1 Skipping physical pages .....	4
2 Why Having an Index in Times of Search Functionality? .....	4
3 Index pages .....	5
3.1 The Default Index page .....	5
3.2 Secondary Index .....	6
3.3 Tertiary Index .....	6
3.4 Combined Index .....	6
3.5 Combined Index - all lower case .....	6
3.6 Math Index .....	7

## 1 Sample Document to Demonstrate the in-dexter package

This document explains how to use the `in-dexter` package in `typst`. It contains several samples of how to use `in-dexter` to effectively index a document. Make sure to look up the `typst` code of this document to explore, what the package can do.

Using the `in-dexter` package in a `typst` document consists of some simple steps:

1. Importing the package `in-dexter`.
2. Marking the words or phrases to include in an index.
3. Generating the index page(s) by calling the `make-index()` function.

### 1.1 Importing the Package

The `in-dexter` package is currently available on GitHub in its home repository (<https://github.com/RolfBremer/in-dexter>). It is still in development and may have breaking changes in its next iteration.

```
#import "./in-dexter.typ": *
```

The package is also available via Typst's build-in Package Manager:

```
#import "@preview/in-dexter:0.4.2": *
```

Note, that the version number of the typst package has to be adapted to get the wanted version.

## 1.2 Marking of Entries

We have marked several words to be included in an index page. The markup for the entry stays invisible. Its location in the text gets recorded, and later it is shown as a page reference in the index page.

```
#index[The Entry Phrase]
```

or

```
#index([The Entry Phrase])
```

or

```
#index("The Entry Phrase")
```

Entries marked this way are going to the “Default” Index. If only one index is needed, this is the only way needed to mark entries. In-dexter can support multiple Indexes. To specify the target index for a marking, the index must be addressed.

```
#index(index: "Secondary")[The Entry Phrase]
```

This is the explicit addressing of the secondary index. It may be useful to define a function for the alternate index, to avoid the explicitness:

```
#let index2 = index.with(index: "Secondary")
```

```
#index2[One Entry Phrase]
```

```
#index2[Another Entry Phrase]
```

### 1.2.1 Nested entries

Entries can be nested. The `index` function takes multiple arguments - one for each nesting level.

```
#index("Sample", "medical", "blood")
#index("Sample", "medical", "tissue")
#index("Sample", "musical", "piano")
#index("Sample")
```

#### 1.2.1.1 LaTeX Style index grouping

Alternatively or complementing to the grouping syntax above, the “bang style” syntax known from LaTeX can be used:

```
#index("Sample!medical!X-Ray")
```

They can even be combined:

```
#index("CombiGroup", "Sample!musical!Chess!")
```

Note that the last bang is not handled as a separator, but is part of the entry. To use the bang grouping syntax, the `make-index()` function must be called with the parameter `use-bang-grouping: true`:

```
#make-index(use-bang-grouping: true)
```

### 1.2.2 Entries with display

These entries use an explicit display parameter. It is used to display the entry on the index page. It can contain rich content, like math expressions:

```
#index(display: "Level3", "Aaa-set3!l2!l3")
#indexMath(display: [$\cal(T)_n$-set], "Aa-set")
#indexMath(display: [$\cal(T)^n$-set], "Aa-set4")
```

### 1.2.3 Advanced entries

Simple math expressions can be used as entry key, like the following sample, where we also provide an initial parameter to put sort the entry under “t” in the index:

```
#indexMath(initial: "t")[$t$-tuple]
```

but note, that more complex ones may not be convertible to a string by in-dexter. In such cases it is recommended to use the display parameter instead:

```
#indexMath(initial: "t", display: [$cal(T)_n$c], "Tnc")
```

this will put the entry in the “t” section, and uses the key (“Tnc”) as sort key within that ‘t’ section. The entry is displayed as  $cal(T)_n$ .

The following entry will place the entry in the “D” section, because we have not provided an explicit initial parameter, so the section is derived from the keys first letter (“DTN”).

```
#indexMath(display: [d-$cal(T)_n$], "DTN")
```

The index-function to mark entries also accepts a tuple value:

```
#indexMath([d-$rho_x$], "RTX")
```

The first value of the tuple is interpreted as the display, the second as the key parameter.

### 1.2.3.1 Symbols

Symbols can be indexed to be sorted under "Symbols", and be sorted at the top of the index like this

```
#indexMath(initial: (letter: "Symbols", sort-by: "#"), [$sigma$])
```

### 1.2.3.2 Formatting Entries

Entries can be formatted with arbitrary functions that map content to content

```
#index(fmt: it => strong(it), [The Entry Phrase])
```

or shorter

```
#index(fmt: strong, [The Entry Phrase])
```

For convenience in-dexter exposes `index-main` which formats the entry bold. It is semantically named to decouple the markup from the actual style. One can decide to have the main entries slanted or color formatted, which makes it clear that the style should not be part of the function name in markup. Naming markup functions according to their purpose (semantically) also eases the burden on the author, because she must not remember the currently valid styles for her intent.

Another reason to use semantically markup functions is to have them defined in a central place. Changing the style becomes very easy this way.

```
#index-main[The Entry Phrase]
```

It is predefined in in-dexter like this:

```
#let index-main = index.with(fmt: strong)
```

Here we define another semantical index marker, which adds an “ff.” to the page number.

```
#let index-ff = index.with(fmt: it => [#it _ff._])
```

## 1.3 The Index Page

To actually create the index page, the `make-index()` function has to be called. Of course, it can be embedded into an appropriately formatted environment, like this:

```
#columns(3)[
  #make-index()
]
```

The `make-index()` function accepts an optional array of indexes to include into the index page. The default value, `auto`, takes all entries from all indexes. The following sample only uses the entries of the secondary and tertiary index. See sample output in Section 3.4.

```
#columns(3)[
  #make-index(indexes: ("Secondary", "Tertiary"))
]
```

### 1.3.1 Skipping physical pages

If page number 1 is not the first physical page of the document, the parameter `use-page-counter` of the `make-index()` function can be set to `true`. Default is `false`. In-dexter uses the page counter instead of the physical page number then.

## 2 Why Having an Index in Times of Search Functionality?

A *hand-picked* or *handcrafted* Index in times of search functionality seems a bit old-fashioned at the first glance. But such an index allows the author to direct the reader, who is looking for a specific topic (using `index-main`), to exactly the right places.

Especially in larger documents and books this becomes very useful, since search engines may provide too many locations of specific words. The index is much more comprehensive, assuming that the author has its content selected well. Authors know best where a certain topic is explained thoroughly or merely noteworthy mentioned (using the `index` function).

Note, that this document is not necessarily a good example of the index. Here we just need to have as many index entries as possible to demonstrate (using a custom made `index-ff` function) the functionality and have a properly filled index at the end.

Even for symbols like ( $\rho$ ). Indexing should work for for any Unicode string like Cyrillic (Скороспелка ) or German (Ölrückstoßabdämpfung). - though we need to add initials:

```
#index(initial: (letter: "C", sort-by: "Ss"), "Скороспелка")
```

or

```
#index(initial: (letter: "Ö", sort-by: "Oo"), "Ölrückstoßabdämpfung").
```

---

### 3 Index pages

In this chapter we emit several index pages for this document.

3.1 The Default Index page .....	5
3.2 Secondary Index .....	6
3.3 Tertiary Index .....	6
3.4 Combined Index .....	6
3.5 Combined Index - all lower case .....	6
3.6 Math Index .....	7

#### 3.1 The Default Index page

Here we generate the Index page in three columns. The default behavior (auto) is to use all indexes together.

<b>SYMBOLS</b>		Formatting Entries	<b>3</b>	Search Engines	4
(ρ)	4	Functionality	4	Search Functionality	4
(σ)	3	<b>H</b>		Searching vs. Index Specific	4
<b>A</b>		Hand Picked	4	<b>C</b>	
-set	2	Handcrafted	4	Скороспелка	4
-set	2	<b>I</b>		<b>T</b>	
Aaa-set3		Index	4	T-tuple	3
L2		Index Page	2, 3	Thoroughly	4
Level3	2	Invisible	2	-c	3
Authors responsibility	4	Iteration	1	Topic	
<b>B</b>		<b>L</b>		Certain	4
Books	4	Large Documents	4	Specific	4
Breaking Changes	1	<b>N</b>			
<b>C</b>		Noteworthy	4		
CombiGroup		<b>O</b>			
Sample		Old-fashioned	4		
Musical		<b>Ö</b>			
Chess!	2	Ölrückstoßabdämpfung	4		
Comprehensive	4	<b>P</b>			
Content	4	Physical page	4		
<b>D</b>		Properly	4		
Demonstrate	4 ff.	Provide	4		
Development	1	<b>R</b>			
D- *Σ(d)	3	D-	3		
D-	3	<b>S</b>			
<b>E</b>		Sample	2		
Engines	4	Medical			
Entries	4	Blood	2		
Environment	3	Tissue	2		
Example	4	X-Ray	2		
Explained	4	Musical			
<b>F</b>		Piano	2		
Filled	4				
Formatting	3				

### 3.2 Secondary Index

Here we select explicitly only the secondary index.

#### E

Example	4
---------	---

#### S

Specific	4
----------	---

---

### 3.3 Tertiary Index

Here we select explicitly only the tertiary index.

#### E

Engines	4
---------	---

#### F

Filled	4
--------	---

---

### 3.4 Combined Index

Here we select explicitly secondary and tertiary indexes.

#### E

Engines	4
---------	---

Example	4
---------	---

#### F

Filled	4
--------	---

#### S

Specific	4
----------	---

---

### 3.5 Combined Index - all lower case

Here we select explicitly secondary and tertiary indexes and format them all lower case.

#### E

engines	4
---------	---

example	4
---------	---

#### F

filled	4
--------	---

#### S

specific	4
----------	---

---

3.6 Math Index

Here we select explicitly only the Math index.

SYMBOLS

( $\sigma$ ) 3

A

-set 2

-set 2

D

D-  $\ast \sum(d)$  3

R

D- 3

T

T-tuple 3

-c 3