i-am-acro

Contents

1 User Guide

1.1 Quick start

First import the package and define your acronyms in a dictionary like this:

```typ
#import "@preview/i-am-acro:0.1.3": * // import everything
#let acronyms = (
  LED: (
    en: (
      short: [LED],
      short-pl: [LEDs],
      long: [Light Emitting Diode],
    ),
    de: (
      short: "LED",
      long: "Leuchtdiode",
      long-pl: "Leuchtdioden",
    ),
  ),
)
```

Then pass the definitions to the package, set the default language and choose whether you want labels to be created between the acronyms and where they are printed (using `print-acronyms`):

```typ
#init-acronyms(acronyms, "en", always-link: false)
```

Now you can use your acronyms!

- `#ac("LED")` -> Light Emitting Diode (LED)
- `#ac("LED")` -> LED
- `#acl("LED")` -> Light Emitting Diode
- `#acl("LED", lang: "de")` -> Leuchtdiode

For all possible acronym functions see Section 2.

1.2 Full example using everything

This will show a configuration showing every possible feature used.

First we create langauge “en” (english) and “de” (german) using the acronyms LED and PLC. For PLC all possible variations are defined.

```typ
#let acronyms = (
  LED: (
    en: (
      short: [LED],
      short-pl: [LEDs],
      long: [Light Emitting Diode],
    ),
    de: (
      short: "LED",
      long: "Leuchtdiode",
```

```
      long-pl: "Leuchtdioden",
    ),
  ),
  PLC: (
    en: (
      short: [PLC],
      short-pl: [PLCs],
      long: [Programmable Logic Controller],
      long-pl: [Programmable Logic Controllers],
    ),
    de: (
      short: [SPS],
      short-pl: [SPSen],
      long: [Speicherprogrammierbare Steuerung],
      long-pl: [Speicherprogrammierbare Steuerungen],
    ),
  ),
  AC: (
    en: (
      short: [AC],
      long: [Acronym],
    ),
    de: (
      short: [AC],
      long: [Acronym],
    ),
  ),
)
```

After that we should define how other languages are written out, in case the second language gets used. See Section 1.3 for more details.

```typ
#let language-display = (
  en: [english],
  de: [german]
)
```

After that we pass this to the package and set the configuration.

```typ
#init-acronyms(
  acronyms,
  "en",
  language-display: language-display,
  default-second-lang: "de",
  always-link: true
)
```

Now we are ready to use out arconyms! The order of operation here is important, since showing the acronym will change its state. So the long form only gets shown the first time or when specifically requested.

Command	Output
`#ac("LED")`	Light Emitting Diode (LED, german: Leuchtdiode)
`#ac("LED")`	LED
`#acl("LED")`	Light Emitting Diode
`#acl("LED", lang: "de")`	Leuchtdiode
`#acs("PLC")`	PLC
`#ac("PLC")`	Programmable Logic Controller (PLC, german: Speicherprogrammierbare Steuerung)

We can also mark acronyms as used or unsed, even if we did not use them. Also we can make the package believe the long version was shown (or not) before. Marking an ancronym as used will make it appear in the acronym table.

```typ
#update-acronym-used("AC", true)
#update-acronym-long-shown("AC", true)
```

Lastly (or firstly, if you prefer it in your document) we print all used acronyms.

```typ
#print-acronyms()
```

Acronym	Definition
AC	Acronym
LED	Light Emitting Diode
PLC	Programmable Logic Controller

1.3 How does the second language work?

As seen before you can set a default second language using the `init-acronyms` function. This is implemented so you can have those two languages shown for an acronym if they work in both languages. There are some examples in german and english like “IR” which is “infrared” in english and “Infrarot” in german. Both is commom to be used in german.

This package will automatically show the second langaue of such words, when the `default-second-lang` parameter is given. The long form of the second language will be shown with the first long form of the default language.
Important here is the `language-display` parameter. This will define how the second language is displayed when needed.

Let’s see an example:

Light Emitting Diode (LED, german: Leuchtdiode)

The acronym “LED” was displayed using `#ac("LED")`, since the default second langaue is given from before (see Listing 2) the german version is also shown. The text `german:` was given by setting the `language-display` parameter accordinly (see Listing 1).
If the default second language is set, but the language was not defined in for the acronym key, the second language will be ignored.

1.4 Print custom acronym table

You can simply import the internal variables `_acroyms`, `_always-link` and `LABEL_KEY` to the stored acroynms with their states. With them you can create your implementation of an acronym table. The example below shows how it could be done. It is important to provide the correct labels. If `always-link` is set to true the labels must be of this scheme "`LABEL_KEY + KEY`" where `LABEL_KEY` is a contant of this package and `KEY` is the key of the printed acronym.

NOTE: It is not possible to sort by content. If you want to sort the arconyms by their short definition, they all need to be strings.

```typ
#import "@preview/i-am-acro:0.1.3": * // import all or only needed

#context {
  // Get the final states of all acronyms
  let final-acronyms = _acronyms.final()
  // get the default language to display the long forms
  let default-lang-final = _default-lang.get()
  // prepare dictionary where all acroyns to be displayed are stored
  let printable-acronyms = (:)

  // Itterate over all acroynms and filter them
  for (key, (value, used, long-shown)) in final-acronyms {
    if used {
      // extract only used acronyms with their default-lang short and long form
      let short-long = (value.at(default-lang-final).short, value.at(default-lang-final).long)
      printable-acronyms.insert(str(key), short-long)
    }
  }

  // Sort by key, it is not possible to sort by content (the short definition).
  printable-acronyms = printable-acronyms.pairs().sorted(key: it => it.at(0))

  // Display acronyms
  grid(
    columns: (auto, 1fr),
    row-gutter: 1em,
    column-gutter: 2em,
    [*Acronym*], [*Definition*],
    ..{
      for (key, value) in printable-acronyms {
        (
          [#value.at(0) #if _always-link.final() { label(LABEL_KEY + key) }],
          [#value.at(1)],
        )
      }
    },
```

```
37     )
38 }
```

The result looks like this:

Acronym	Definition
AC	Acronym
LED	Light Emitting Diode
PLC	Programmable Logic Controller

This is very similar to using `print-acronyms` (well it is what i came up with). I expect you to implement your own display of the acronyms to fit the design of your document. This is simply an example of how it could be done.

2 Full package documentation

2.1 i-am-acro

- ac()
- ac-custom()
- ac-suffix()
- acl()
- aclp()
- acp()
- acs()
- acsp()
- display-text()
- init-acronyms()
- print-acronyms()
- update-acro-lang()
- update-acro-second-lang()
- update-acronym-long-shown()
- update-acronym-used()
- verfiy-acronym-exists()

Variables

- _acronyms
- _language-display
- _default-second-lang
- _default-lang
- _always-link
- LABEL_KEY

ac

Show the acronym. If it is first shown, the long version with the short will be displayed.
This will mark the acronym as used.

Parameters

```
ac(
  key: string,
  lang: string none,
  second-lang: string none
) -> content
```

key `string`

Key of the desired acronym.

lang `string` or `none`

Language to be displayed. none will use the default language

Default: `none`

second-lang `string` or `none`

Second langauge to be displayed. If “auto” is passed, `_default-second-lang` will be used.

Default: `auto`

ac-custom
Display an acronym with custom short and long form. This is useful in case a custom ending or similar is needed once. The acronym will be treated the same as using `ac()`.
Note: Since the short and long form are custom, the language and suffix are omitted as parameters.

Parameters

```
ac-custom(
  key: string,
  short: string content,
  long: string content,
  suffix: string content none
) -> content
```

key `string`

Key of the desired acronym.

short `string` or `content`

Custom short form of the acronym.

long `string` or `content`

Custom long form of the acronym.

suffix `string` or `content` or `none`

Suffix which whill be displayed after the acronym. When none, suffix will be ignored

Default: `none`

ac-suffix
Display an acronym with a suffix, hyphenated to it (e.g., acronym-suffix).

Parameters

```
ac-suffix(
  key: string,
  suffix: string content,
  lang: string none,
  plural: bool
) -> content
```

key `string`

Key of the desired acronym.

suffix `string` or `content`

Suffix which whill be displayed after the acronym.

lang `string` or `none`

Language to be displayed. none will use the default language

Default: `none`

plural `bool`

Show the plural form of the acrony. An “s” will be appended if the plural form ist not defined.

Default: `false`

acl

Display the long form of an acronym. This will not set “long-shown” or “used” to true (since the short version is not displayed with it)

Parameters

```
acl(
  key: string,
  lang: string none
) -> content
```

key `string`

Key of the desired acronym.

lang `string` or `none`

Language to be displayed. none will use the default language

Default: `none`

aclp
Display the long plural form of an acronym. If no plural form was defined, “s” will be appended. This will not set “long-shown” to true (since the short version is not displayed with it)

Parameters

```
aclp(
  key: string,
  lang: string none
) -> content
```

key `string`

Key of the desired acronym.

lang `string` or `none`

Language to be displayed. none will use the default language

Default: `none`

acp
Display the plural form an acronym. If it is first shown, the long version with the short will be displayed. Also if no plural form was defined, “s” will be appended.
This will mark the acronym as used.

Parameters

```
acp(
  key: string,
  lang: string none
) -> content
```

key `string`

Key of the desired acronym.

lang `string` or `none`

Language to be displayed. none will use the default language

Default: `none`

acs
Show the acronym in the short form. This will mark the acronym as used.

Parameters

```
acs(
  key: string,
  lang: string none
) -> content
```

key `string`

Key of the desired acronym.

lang `string` or `none`

Language to be displayed. none will use the default language

Default: `none`

acsp
Show the acronym in the short plural form. If no short plural form was defined, “s” will be appended.
This will mark the acronym as used.

Parameters

```
acsp(
  key: string,
  lang: string none
) -> content
```

key `string`

Key of the desired acronym.

lang `string` or `none`

Language to be displayed. none will use the default language

Default: `none`

display-text
Display text with a link, when desired. The link will use `LABEL_KEY` with the key paramter to generate the label. The label will all point to the acronyms in `print-acronyms()`.

Parameters

```
display-text(
  text: content string,
  key: string,
  do-link: bool
) -> content
```

text `content` or `string`

Text to be printed

key `string`

Key used for label generation with `LABEL_KEY`. Only required when “do-link” is true.

Default: `none`

do-link `bool`

Generate a link to the printed aconyms from `print-acronyms()`.

Default: `false`

init-acronyms
Initialize the acronyms and the default settings.

Parameters

```
init-acronyms(
  acronyms: dictionary,
  default-lang: string,
  default-second-lang: string,
  language-display: dictionary,
  always-link
) -> none
```

acronyms `dictionary`

Dictionary containing all the defined acronyms.

default-lang `string`

Set the default language. For exmaple “en”, “de”, “fr”. You can change this later using `update-acro-lang()`

default-second-lang `string`

Set the default second language. For exmaple “en”, “de”, “fr”. You can change this later using `update-acro-second-lang()`

Default: `none`

language-display `dictionary`

Languages and their written form being used in `ac()` when two languages are displayed. Use “none” if this is not requiered.

Default: `none`

always-link

Controls if labels and links will be generated. The label will point to `print-acronyms()`, the link will be created on the displayed acronym.

Default: `true`

print-acronyms

Print all used acronyms in a grid. This will create labels, if `_always-link` is set to true. The list will be sorted by the acronym key. It is not possible to sort by content.

Parameters

```
print-acronyms() -> content
```

update-acro-lang

Update the current default language used for acronyms. Useful when writing a bilingual document.

Parameters

```
update-acro-lang(lang: string none) -> none
```

lang `string` or `none`

Language to be displayed. Examples: “en”, “de”, “fr”

update-acro-second-lang

Update the current default second language used for acronyms.

Parameters

```
update-acro-second-lang(lang: string none) -> none
```

lang `string` or `none`

Language to be displayed. Examples: “en”, “de”, “fr”

update-acronym-long-shown
Update the status of “long-shown” for a acronym from `_acronyms`.

Parameters

```
update-acronym-long-shown(
  key: string,
  long-shown: bool
) -> none
```

key `string`

Key of the acronym, which will be updated.

long-shown `bool`

New value for the “long-shown” key of the selected acronym.

update-acronym-used
Update the status of “used” for a acronym from `_acronyms`. This may cause the error “label does not exist in the document” if you set used to false and do not call `ac()` to reset the state to true.

Parameters

```
update-acronym-used(
  key: string,
  used: bool
) -> none
```

key `string`

Key of the acronym, which will be updated.

used `bool`

New value for the “used” key of the selected acronym.

verfiy-acronym-exists
Verfiy that an acronym, the requiered language, short and long term exist.

Parameters

```
verfiy-acronym-exists(
  key: string,
  lang: string
) -> error
```

key `string`

Key of the desired acronym.

lang `string`

Language to be tested.

_acronyms `dictionary`

State variable conataining all acronyms. The keys `short` and `long` must be provided for each language. Their plural forms are optional.
Use the structure as shown in the example here:

```typ
#let my-acronyms = (
  key: (
    language1: (
      short: [short version],
      long: [long version],
      short-pl: "short plural",
      long-pl: [long plural]
    ),
    language2: (
      short: [second language short],
      long: [second language long],
      short-pl: [2nd lang short plural],
      long-pl: [2nd lang long plural]
    )
  ),
  LED: (
   en: (
     short: [LED],
     short-pl: [LEDs],
     long: [Light Emitting Diode],
   ),
   de: (
     short: "LED",
```

```
      long: "Leuchtdiode",
      long-pl: "Leuchtdioden",
    ),
  ),
)
```

```
(
  key: (
    language1: (
      short: [short version],
      long: [long version],
      short-pl: "short plural",
      long-pl: [long plural],
    ),
    language2: (
      short: [second language short],
      long: [second language long],
      short-pl: [2nd lang short plural],
      long-pl: [2nd lang long plural],
    ),
  ),
  LED: (
    en: (
      short: [LED],
      short-pl: [LEDs],
      long: [Light Emitting Diode],
    ),
    de: (
      short: "LED",
      long: "Leuchtdiode",
      long-pl: "Leuchtdioden",
    ),
  ),
)
```

_language-display

State variable containing languages and their written form being used in `ac()` when two languages are displayed.

```
#let language-mapping = (
  en: [english],
  de: [german],
  fr: "french",
)
#language-mapping
```

```
(en: [english], de: [german], fr: "french")
```

_default-second-lang `string`

State variable, what the default second language is to be used on acronyms. Use “none” to disable this feature. This variable is context aware. Exmaple: “en”, “de”, “fr”

_default-lang `string`

State variable, what the default language is to be used with acronyms. This is context aware. Exmaple: “en”, “de”, “fr”

_always-link `bool`

State variable, if links and labels should be generated. If this is true `print-acronyms()` must be used. Alternatively you can create the labels yourself. See Section 1.4.
Only the final value will be used.

LABEL_KEY `string`

Prefix of label keys. Used to link acronyms to the printed acronym list.