wordometer

A small Typst package for quick and easy in-document word counts.

github.com/Jollywatt/typst-wordometer

Version 0.1.0

Basic usage

```
#import "@preview/wordometer:0.1.0": word-count, total-words

#show: word-count

In this document, there are #total-words words all up.

#word-count(total => [
  The number of words in this block is #total.words
  and there are #total.characters letters.
])
```

Excluding elements by type or label

```
#show: word-count.with(exclude: ("heading", "strike"))

= This Heading Doesn't Count

In this document #strike[(excluding me)], there are #total-words words all up.

#word-count(total => [
  One, two, three, four.
  #[That was #total.words, excluding this sentence!] <no-wc>
], exclude: <no-wc>)
```

`concat-adjacent-text()`

Simplify an array of content by concatenating adjacent text elements.

Doesn't preserve content exactly; `smartquotes` are replaced with `'` or `"`. This is used on `sequence` elements because it improves word counts for cases like "Digby's", which should count as one word.

For example, the content

Qu'est-ce **que** c'est !?

is structured as:

```
(
  [Qu],
  smartquote(double: false),
  [est-ce],
  [ ],
  strong(body: [que]),
  [ ],
  [c],
  smartquote(double: false),
  [est],
  [ ],
  [!?],
)
```

This function simplifies this to:

```
([Qu'est-ce ], strong(body: [que]), [ c'est !?])
```

Parameters

```
concat-adjacent-text(children: array)
```

children `array`

Array of content to simplify.

`map-tree()`

Traverse a content tree and apply a function to textual leaf nodes.

Descends into elements until reaching a textual element (`text` or `raw`) and calls `f` on the contained text, returning a (nested) array of all the return values.

Parameters

```
map-tree(
  f: function,
  content: content,
  exclude: array
)
```

f `function`

Unary function to pass text to.

content `content`

Content element to traverse.

exclude `array`

List of labels or element names to skip while traversing the tree. Default value includes equations and elements without child content or text: `"display"`, `"equation"`, `"h"`, `"hide"`, `"image"`, `"line"`, `"linebreak"`, `"locate"`, `"metadata"`, `"pagebreak"`, `"parbreak"`, `"path"`, `"polygon"`, `"repeat"`, `"smartquote"`, `"space"`, `"update"`, and `"v"`.

To exclude figures, but include figure captions, pass the name `"figure-body"` (which is not a real element). To include figure bodies, but exclude their captions, pass the name `"caption"`.

Default: `IGNORED_ELEMENTS`

`string-word-count()`

Get a basic word count from a string.

Returns a dictionary with keys:

- `characters`: Number of non-whitespace characters.
- `words`: Number of words, defined by `regex("\b[\w'’]+\b")`.
- `sentences`: Number of sentences, defined by `regex("\w+\s*[.?!]")`.

Parameters

```
string-word-count(string: string) -> dictionary
```

`word-count()`

Perform a word count on content.

Master function which accepts content (calling `word-count-global()`) or a callback function (calling `word-count-callback()`).

Parameters

```
word-count(
  arg: content fn,
  ..options:
) -> dictionary
```

arg `content` or `fn`

Can be:

- `content`: A word count is performed for the content and the results are accessible through `#total-words` and `#total-characters`. This uses a global state, so should only be used once in a document (e.g., via a document show rule: `#show: word-count`).
- `function`: A callback function accepting a dictionary of word count results and returning content to be word counted. For example:

```
#word-count(total => [This sentence contains #total.characters letters.])
```

..options

Additional named arguments:

- `exclude`: Content to exclude from word count (see `map-tree()`).

---

word-count-callback()

Simultaneously take a word count of some content and insert it into that content.

It works by first passing in some dummy results to `fn`, performing a word count on the content returned, and finally returning the result of passing the word count retults to `fn`. This happens once — it doesn't keep looping until convergence or anything!

For example:

```
#word-count-callback(stats => [There are #stats.words words])
```

Parameters

```
word-count-callback(
  fn: function,
  ..options:
) -> content
```

fn `function`

A function accepting a dictionary and returning content to perform the word count on.

..options

Additional named arguments:

- `exclude`: Content to exclude from word count (see `map-tree()`).

`word-count-global()`

Get word count statistics of the given content and store the results in global state. Should only be used once in the document.

The results are accessible anywhere in the document with `#total-words` and `#total-characters`, which are shortcuts for the final values of states of the same name (e.g., `#locate(loc => state("total-words").final(loc))`)

Parameters

```
word-count-global(
  content: content,
  ..options:
) -> content
```

content `content`

Content to word count.

..options

Additional named arguments:

- `exclude`: Content to exclude from word count (see `map-tree()`).

`word-count-of()`

Get word count statistics of a content element.

Returns a results dictionary, not the content passed to it. (See `string-word-count()`).

Parameters

```
word-count-of(
  content: content,
  exclude: array,
  counter: fn
) -> dictionary
```

exclude `array`

Content elements to exclude from word count (see `map-tree()`).

Default: `(:)`

counter `fn`

A function that accepts a string and returns a dictionary of counts.

For example, to count vowels, you might do:

```
#word-count-of([ABCDEFG], counter: s => (
    vowels: lower(s).matches(regex("[aeiou]")).len(),
))
```

Default: `string-word-count`