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.
])
```

`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 texual 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,
  el: content
)
```

f `function`

Unary function to pass text to.

el `content`

Content element to traverse.

`set-word-count-state()`

Get word count statistics of some content and store the results is a global state.

The results are accessible anywhere in the document with `#total-words` and `#total-characters`.

Parameters

```
set-word-count-state(el: content)
```

`word-count()`

Perform a word count.

Accepts content (in which case results are accessible with `#total-words` and `#total-characters`) or a callback function (see `word-count-callback()`).

Passing content only works if you do it once in your document, because the results are stored in a global state. For multiple word counts, use the callback style.

Parameters

```
word-count(arg: content fn)
```

`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)
```

fn `function`

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

`word-count-of()`

Get word count statistics of a content element.

Returns a results dictionary, not the content passed to it.

Parameters

```
word-count-of(el: content) -> dictionary
```