Notebook extensions

# Jupyter Notebook Extensions[¶](#Jupyter-Notebook-Extensions)

A wide variety of extensions are available that can be used to customise the Jupyter notebook environment. Customisations include the ability to add notebook toolbar or menu options and customise the behaviour of notebook cells.

In [5]:

!ls

buildtest IPython Cell Magic.ipynb source  
images nb\_extension\_wordexport tm351test  
installationTests Notebook extensions.ipynb Untitled.ipynb

In [1]:

!sudo pip install --upgrade ./nb\_extension\_wordexport  
!sudo jupyter bundlerextension enable --py wordexport.wordexport --sys-prefix

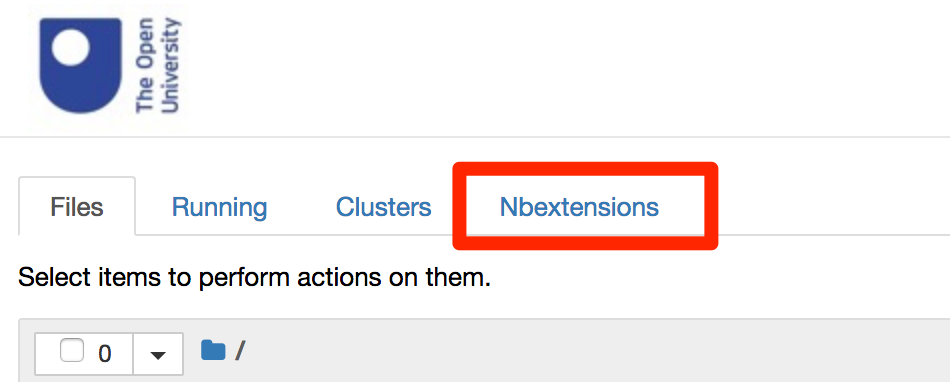
The directory '/home/oustudent/.cache/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.  
The directory '/home/oustudent/.cache/pip' or its parent directory is not owned by the current user and caching wheels has been disabled. check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.  
Processing ./nb\_extension\_wordexport  
Installing collected packages: wordexport  
 Found existing installation: wordexport 0.0.1  
 Uninstalling wordexport-0.0.1:  
 Successfully uninstalled wordexport-0.0.1  
 Running setup.py install for wordexport ... done  
Successfully installed wordexport-0.0.1  
Enabling wordexport\_wordexport bundler wordexport.wordexport...

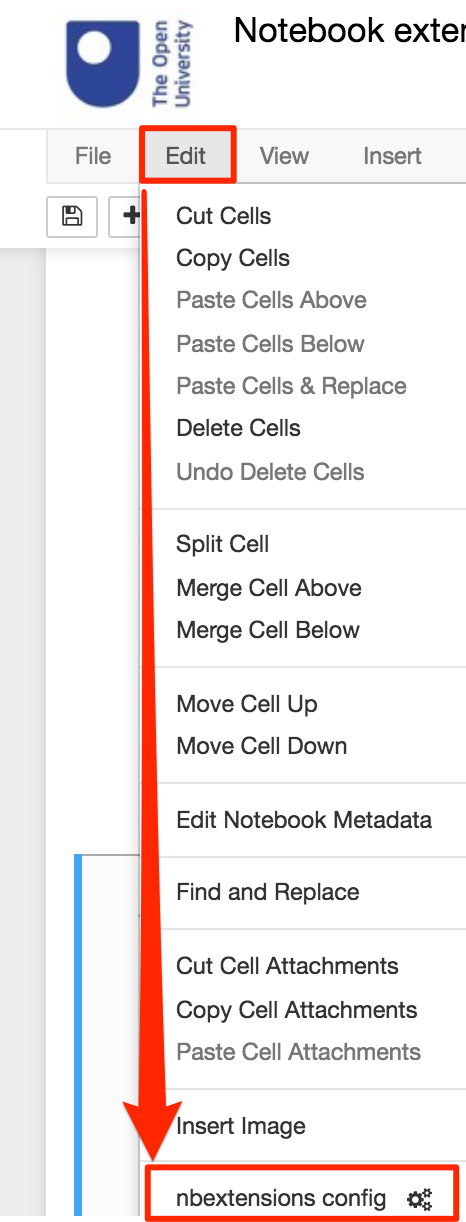
In [1]:

!pandoc -s "Notebook extensions.html" -o "Notebook extensions.docx"

Several extensions are pre-enabled in the TM351 virtual machine, as described below.

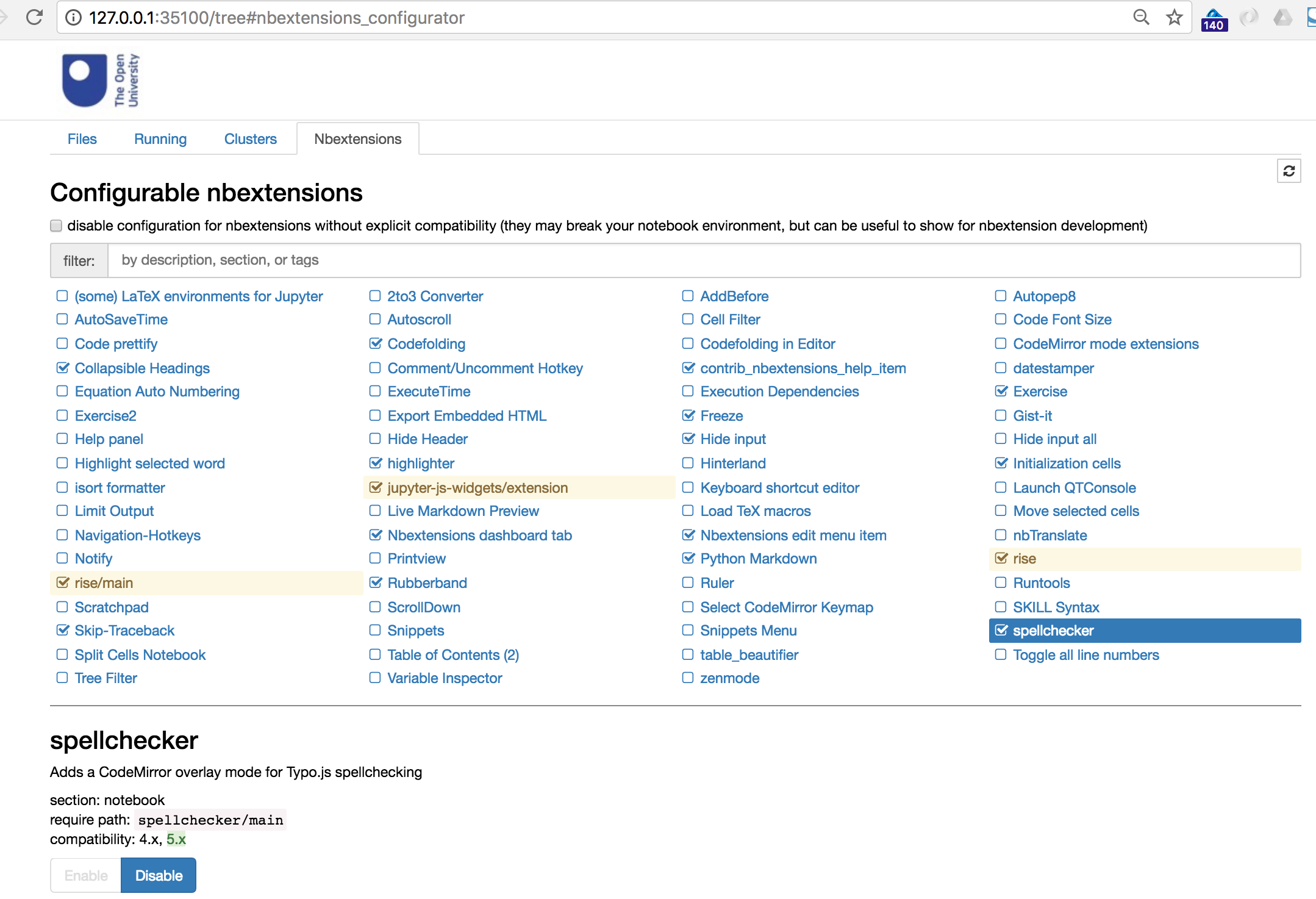
Notebook extensions can be managed via the [nbextensions\_configurator](/nbextensions), which can be found via the tab on the notebook homepage or from within a notebook via the Edit->nbextensions config menu option.

[](/nbextensions)

[](/nbextensions)

## Notebook Extension Configurator[¶](#Notebook-Extension-Configurator)

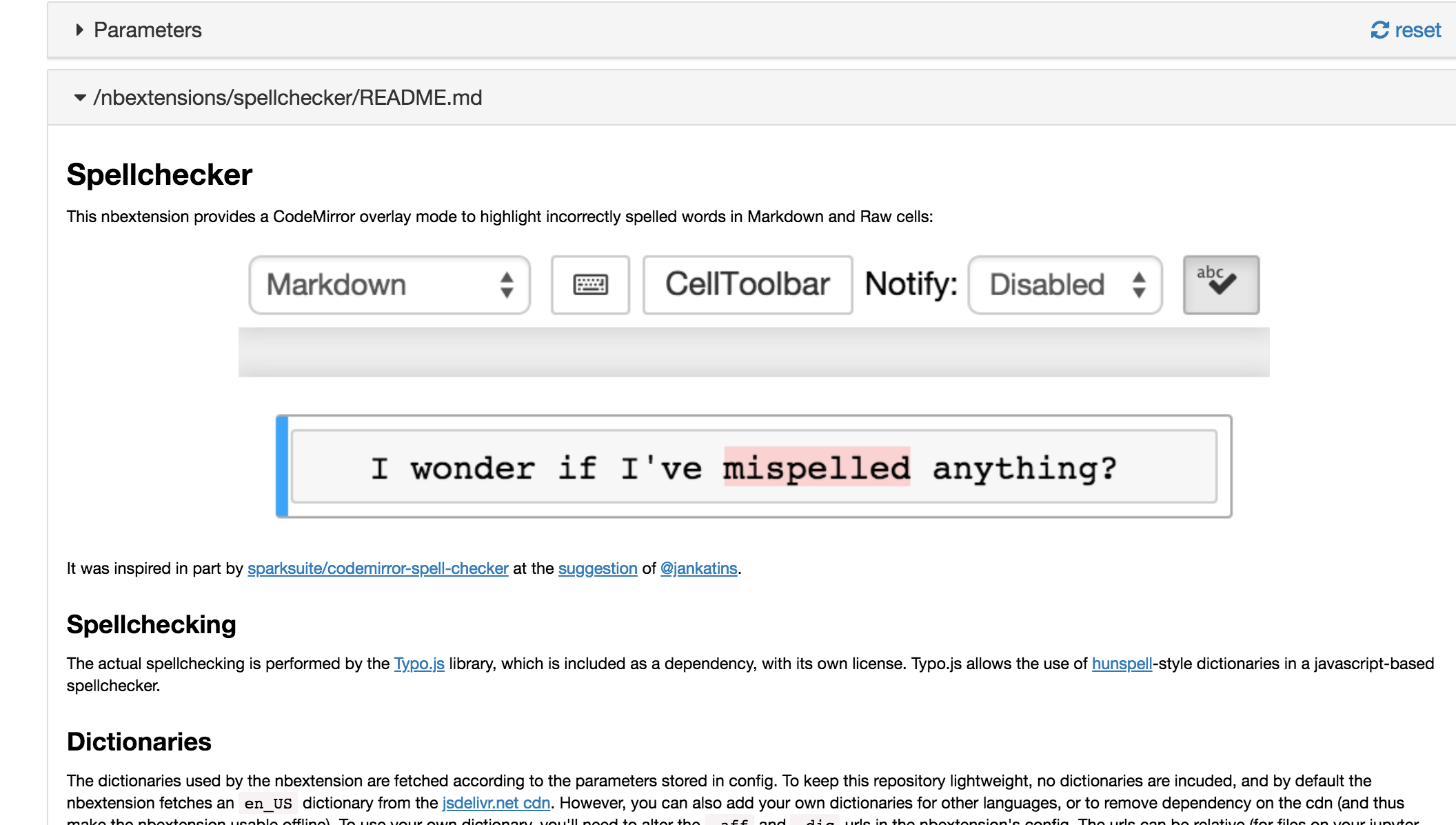
The *notebook extension configurator* allows you to enable or disable extensions by checking the corresponding select box. Extensions are versioned and an option exists to disable activation of an extension from the configurator if it is not explicilty declared as compatible with the current notebook. In many cases, these extensions will still work, so I tend to allow all extensions to be enabled and then just ignore any that cause me problems with a particular notebook.

[](/nbextensions)

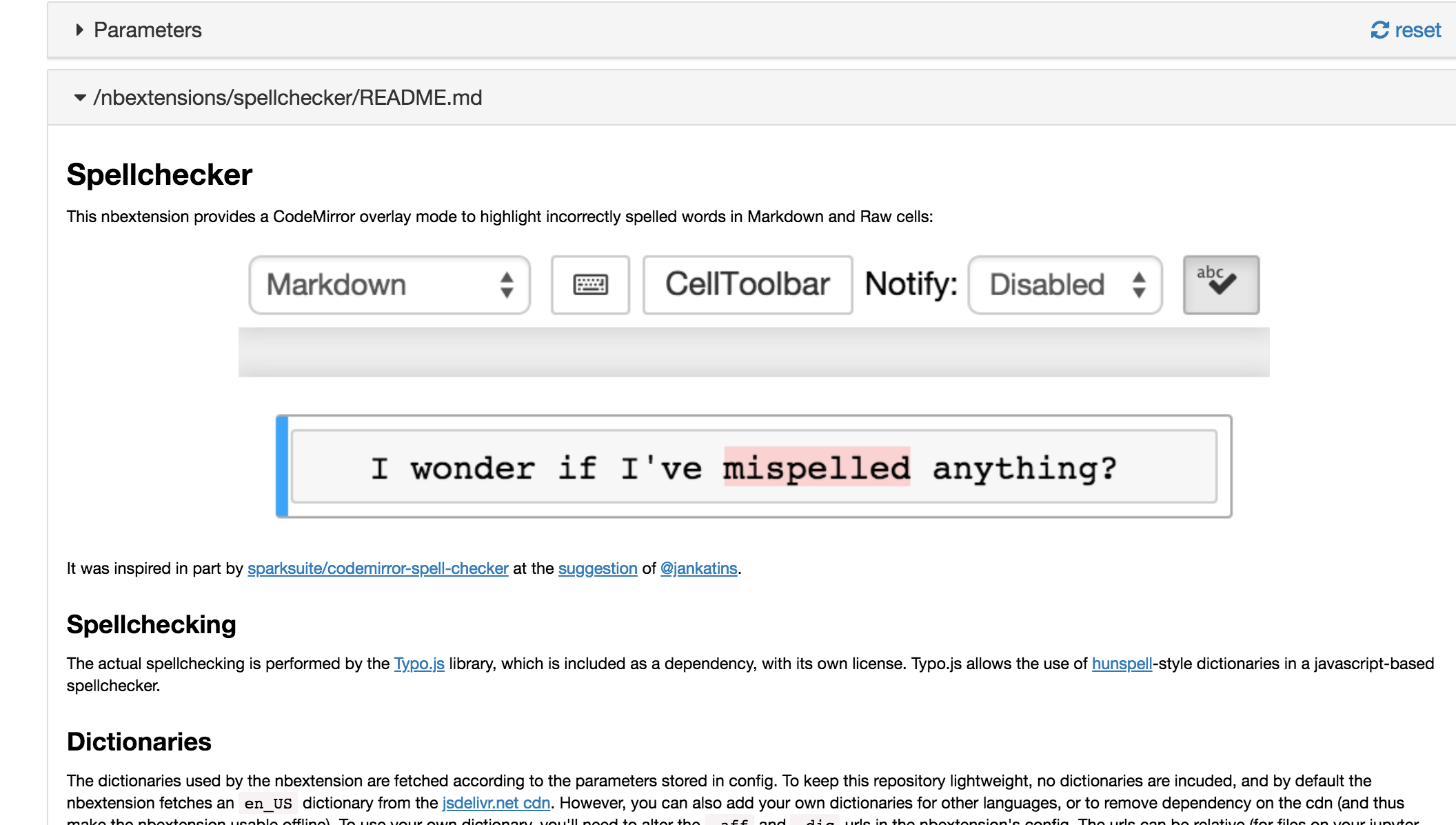
If you click on the name of an extension, a descriptive panel appears that describes that extension. This panel contains two sections:

* a parameter section
* a README section

The parameter section allows you to customise the settings of the extension itself:



The README section describes the operation of the extension:



## Pre-Enabled Extensions[¶](#Pre-Enabled-Extensions)

The TM351 VM is bundled with the "official" [ipython-contrib/jupyter\_contrib\_nbextensions](https://github.com/ipython-contrib/jupyter_contrib_nbextensions) as well as several other custom extensions including extensions developed to support OU related activities.

These extensions may be disabled from the notebook extension configurator, although the notebooks may not displayed as intended if you do disable them.

### General Notebook Presentation[¶](#General-Notebook-Presentation)

* [collapsible\_headings](/nbextensions/?nbextension=collapsible_headings/main): provides upport for collapsible heading so that hierarchically organised sections of the notebook appearing in separate markdown cells can be collapsed or expanded.

The [Table of Contents (2)](/nbextensions/?nbextension=toc2/main) (*sic*) extension (*not enabled by default*) can also use headings to create a table of contents that appears at the top of the notebook (check the appropriate setting).

The [Code Font Size](/nbextensions/?nbextension=code_font_size/code_font_size) extension (*not enabled by default*) allows you to increase or decrease the font size of code cells or the contents of markdown cells in edit mode.

### Workflow Support[¶](#Workflow-Support)

* [highlighter](/nbextensions/?nbextension=highlighter/highlighter): highlight selected text in a markdown cell
* [spellchecker](/nbextensions/?nbextension=spellchecker): performs a live spell check as you edit the contents of a markdown cell
* [export\_embedded](/nbextensions/?nbextension=export_embedded/main): adds an export option to the File menu that allows you to export the current notebook as an HTML file with images embedded in it (File -> Download as -> HTML Embedded (.html))
* [freeze](/nbextensions/?nbextension=freeze/main): allows you to make a cell read-only / non-editable or frozen (read-only and non-executable if it's a code cell)

In [22]:

!sudo jupyter bundlerextension enable --py jupyter\_docx\_bundler --sys-prefix

Enabling docx\_bundler bundler jupyter\_docx\_bundler...

In [ ]:

print('Read only code cell')

In [ ]:

print('Frozen code cell')

### Python / Code Cell Related Presentation and Styling[¶](#Python-/-Code-Cell-Related-Presentation-and-Styling)

* [init\_cell](/nbextensions/?nbextension=init_cell/main): allows one or more cells to set as initialisation cells that can be executed from the toolbar or autmomatically whenever the notebook kernel is (re)started.
* [skip-traceback](/nbextensions/?nbextension=skip-traceback/main): collapse Python error messages under a succinct statement of the error.

In [14]:

pront("What happens if you run this cell, as is...?")

---------------------------------------------------------------------------  
NameError Traceback (most recent call last)  
<ipython-input-14-d130e3700c65> in <module>()  
----> 1 pront("Did I type that wrong???")  
  
NameError: name 'pront' is not defined

* [codefolding](/nbextensions/?nbextension=codefolding/main): allow code folding on function names, first line comments and cell magics

In [5]:

def collapsed\_function1():  
 pass  
  
def uncollapsed\_function():  
 pass  
  
def collapsed\_function2():  
 pass

In [ ]:

# First line comment lets you collapse a code cell in its entirety  
pass

* [hide\_input](/nbextensions/?nbextension=hide_input/main): allows the output of a specific code cell to be hidden. For example, select the (hidden) code cell directly beneath cell and see if you can reveal it.

If you run the cell while it is hidden, any output *will* be displayed. You can also select a hidden code cell and run it to see if it renders any output.

In [13]:

# Found it!  
print('Hello... The code that produced me is (or at least, was) hidden...')

Hello... The code that produced me is (or at least, was) hidden...

* [python-markdown](/nbextensions/?nbextension=python-markdown/main): allows the value of python variables to be returned inline into a markdown cell

Some markdown with a sum in it: 1 + 1 = 2. Put this markdown cell into cell edit mode to see the code...

It works with *pandas* too...

In [10]:

import pandas as pd  
df = pd.DataFrame({'col1':['a','b','c'],'col2':[1,2,3] })

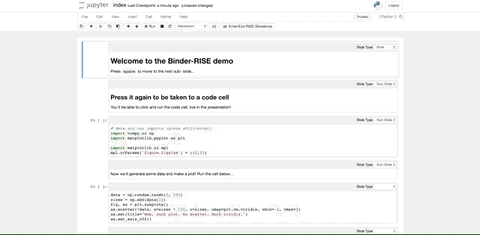
Here's my table...

|  |  |  |
| --- | --- | --- |
|  | col1 | col2 |
| 0 | a | 1 |
| 1 | b | 2 |
| 2 | c | 3 |

Although not directly relevant to notebook extensions, it's also worth noting that *pandas* dataframes can be styled as per [*pandas* docs - Styling](https://pandas.pydata.org/pandas-docs/stable/style.html).

### Module Team / AL Support[¶](#Module-Team-/-AL-Support)

* rubberband:
* exercise:
* [RISE](/nbextensions/?nbextension=rise): this extension turns a notebook with appropriately set cell metadata into a "live slideshow", with cells appearing as slides or slide fragments. Code cells are executable within the slideshow. Any output returned from an executed code cell will also be rendered within the slide. For more detail, see the [official docs](https://damianavila.github.io/RISE/).



### OU custom menu extensions[¶](#OU-custom-menu-extensions)

As well as the notebook toolbar extensions, two custom OU extensions have been added to the File menu.

* [MS Word export](https://github.com/innovationOUtside/nb_extension_wordexport): download the current notebook as a Microsoft Word document (File -> Download as -> MS Word (.docx))
* [ODS](https://github.com/innovationOUtside/nb_extension_odszip) : download the current notebook as an HTML file and compress it with the original (saved) .ipynb document as a zip file using an .nbk suffix so that it can be uploae to *Open Design Studio* (File -> ODSzip (.nbk))