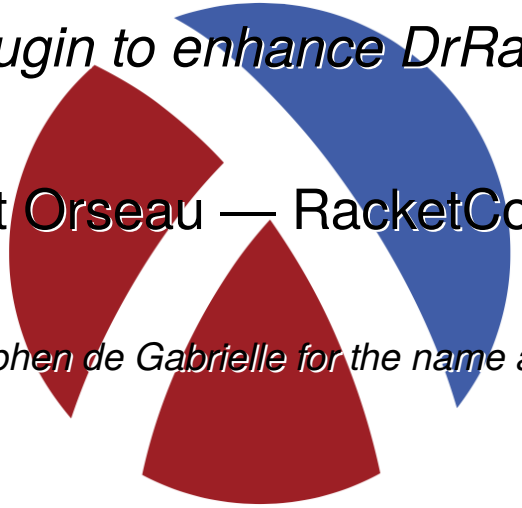
Quickscript

A plugin to enhance DrRacket



Laurent Orseau — RacketCon 2018

Thanks to Stephen de Gabrielle for the name and some help

A simple use case

Suppose for your current project

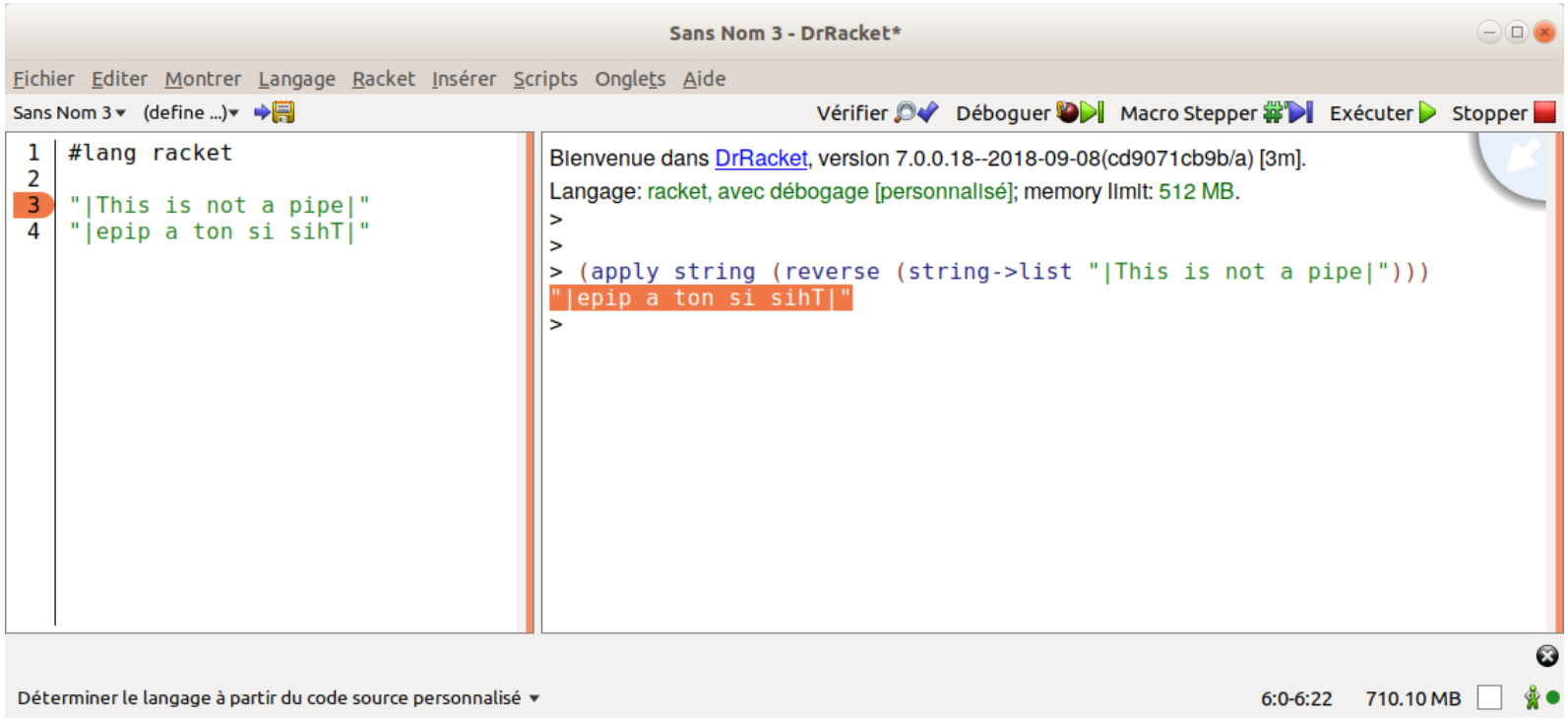
you often need to reverse text strings

A simple use case

Suppose for your current project

you often need to reverse text strings

- First idea:



DrRacket keybindings

Purpose: Often reverse strings in the definitions window

- Second idea: Use DrRacket keybindings

DrRacket keybindings

Purpose: Often reverse strings in the definitions window

- Second idea: Use DrRacket keybindings
- Then just a matter of select and perform keyboard shortcut

DrRacket keybindings

Purpose: Often reverse strings in the definitions window

- Second idea: Use DrRacket keybindings
- Then just a matter of select and perform keyboard shortcut
- Need to deal with keymaps

DrRacket keybindings

Purpose: Often reverse strings in the definitions window

- Second idea: Use DrRacket keybindings
- Then just a matter of select and perform keyboard shortcut
- Need to deal with keymaps
- Need to remember keybinding
  - No menu item

DrRacket keybindings

Purpose: Often reverse strings in the definitions window

- Second idea: Use DrRacket keybindings
- Then just a matter of select and perform keyboard shortcut
- Need to deal with keymaps
- Need to remember keybinding
  - No menu item
- Limited number of keybindings
  - Need to resort to unintuitive combinations
  - Hard to have many variants of the script

DrRacket keybindings

Purpose: Often reverse strings in the definitions window

- Second idea: Use DrRacket keybindings
- Then just a matter of select and perform keyboard shortcut
- Need to deal with keymaps
- Need to remember keybinding
  - No menu item
- Limited number of keybindings
  - Need to resort to unintuitive combinations
  - Hard to have many variants of the script

→ Not very friendly to short-lived scripts

DrRacket plugins

- Third idea: DrRacket plugin
- Can do anything
  - Menu items and more

DrRacket plugins

- Third idea: DrRacket plugin
- Can do anything
  - Menu items and more
- Need to deal with the plugin system

DrRacket plugins

- Third idea: DrRacket plugin
- Can do anything
  - Menu items and more
- Need to deal with the plugin system
- Need to restart DrRacket
  - Long development/debugging process

DrRacket plugins

- Third idea: DrRacket plugin
- Can do anything
  - Menu items and more
- Need to deal with the plugin system
- Need to restart DrRacket
  - Long development/debugging process

→ Not very friendly to short-lived scripts

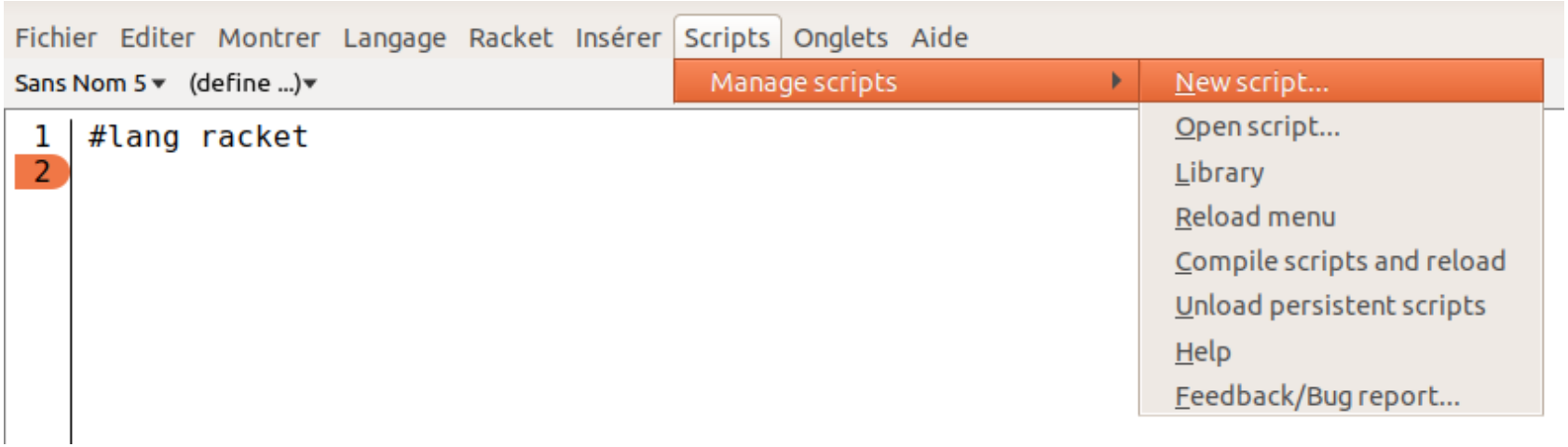
Quickscript

- A plugin for DrRacket
- For small to medium scripts
- No need to restart DrRacket
- Menu items + keybindings
  - Easy to organize

Install Quickscript

```
raco pkg install quickscript
```

New 'Script' menu



A quick Quickscript script

Scripts > Manage Scripts > New script (meta-S M N)

A quick Quickscript script

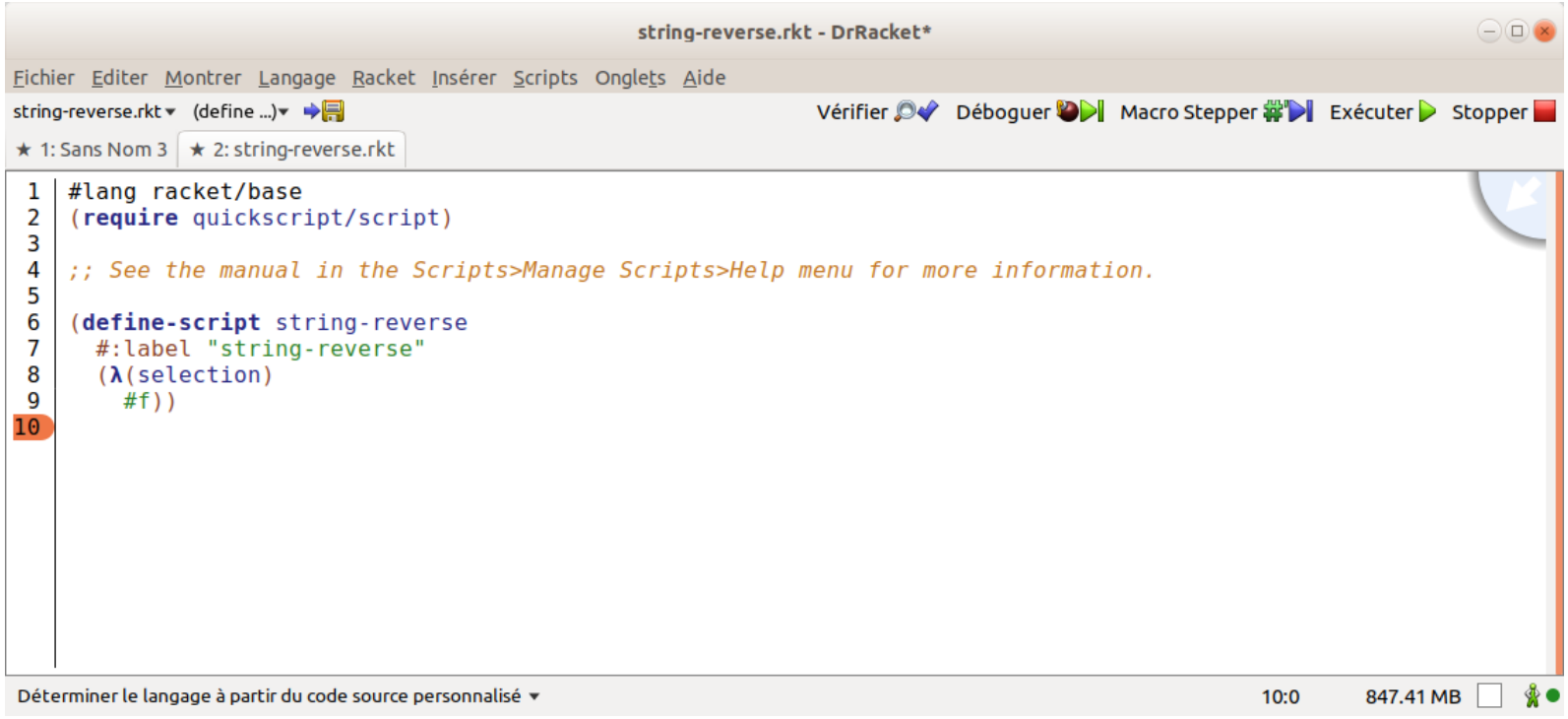
Scripts > Manage Scripts > New script (meta-S M N)

Enter: *string-reverse*

A quick Quickscript script

Scripts > Manage Scripts > New script (meta-S M N)

Enter: *string-reverse*



New menu item: **`Scripts > string-reverse`**

A quick Quickscript script

```
(define-script string-reverse
  #:label "String Reverse"
  (λ (selection)
    (apply string (reverse (string->list selection)))))
```

Save

A quick Quickscript script

```
(define-script string-reverse
  #:label "String Reverse"
  (λ (selection)
    (apply string (reverse (string->list selection)))))
```

Save

Scripts > Manage Scripts > Reload menu (meta-S M R)

A quick Quickscript script

```
(define-script string-reverse
  #:label "String Reverse"
  (λ (selection)
    (apply string (reverse (string->list selection)))))
```

Save

Scripts > Manage Scripts > Reload menu (meta-S M R)

Select a string, and click on **Scripts > String Reverse**

String Reverse quick script: Menu hotkey

```
(define-script string-reverse
  #:label "String &Reverse"
  (λ (selection)
    (apply string (reverse (string->list selection)))))
```

Select a string, then meta-S R

String Reverse quick script: Reorganize menu

```
(define-script string-reverse
  #:label "String &Reverse"
  #:menu-path ("Sele&ction")
  (λ (selection)
    (apply string (reverse (string->list selection)))))
```

Select a string, then meta-S C R

String Reverse quick script: Keyboard shortcut

```
(define-script string-reverse
  #:label "String &Reverse"
  #:menu-path ("Sele&ction")
  #:shortcut #\r
  #:shortcut-prefix (ctl alt)
  (λ (selection)
    (apply string (reverse (string->list selection)))))
```

Select a string, then ctl-alt-R

Procedure arguments

- Unusual: Callee can request arguments

```
(require quickscript/script
         racket/class)

(define-script copy-to-interactions
  #:label "Copy selection to interactions"
  (λ (selection #:interactions ints)
    (send* ints
      (begin-edit-sequence)
      (insert selection)
      (end-edit-sequence))
    #f))
```

Procedure arguments

Callee can request arguments

- `#:interactions ints`
  - The current interactions editor

Procedure arguments

Callee can request arguments

- `#:interactions ints`
  - The current interactions editor
- `#:definitions defs`
  - The current definitions editor

Procedure arguments

Callee can request arguments

- `#:interactions ints`
  - The current interactions editor
- `#:definitions defs`
  - The current definitions editor
- `#:editor ed`
  - The current editor, either definitions or interactions
  - Manipulate contents (text, snips, etc.) of the editor

Procedure arguments

Callee can request arguments

- `#:interactions ints`
  - The current interactions editor
- `#:definitions defs`
  - The current definitions editor
- `#:editor ed`
  - The current editor, either definitions or interactions
  - Manipulate contents (text, snips, etc.) of the editor
- `#:frame fr`
  - Manipulate DrRacket's frame (menus, tabs, ...)

Procedure arguments

Callee can request arguments

- `#:interactions ints`
  - The current interactions editor
- `#:definitions defs`
  - The current definitions editor
- `#:editor ed`
  - The current editor, either definitions or interactions
  - Manipulate contents (text, snips, etc.) of the editor
- `#:frame fr`
  - Manipulate DrRacket's frame (menus, tabs, ...)
- `#:file f`
  - File path of the current tab

Script properties

```
(script-help-string "A description of the script")

(define-script a-complete-script
  ; Properties:
  #:label "Full script"
  #:menu-path ("Submenu" "Subsubmenu")
  #:shortcut #\a
  #:shortcut-prefix (ctl shift)
  #:output-to selection ; (one-of/c selection new-tab message-box clipboard #f)
  #:persistent
  #:os-types (unix macosx windows)
  ; Procedure with its arguments:
  (λ (selection #:frame fr
                #:editor ed
                #:definitions defs
                #:interactions ints
                #:file f)
    "Hello world!"))
```

Persistent scripts

- By default, scripts are non-persistent
  - Namespace lives only the time of the execution
- Persistent script
  - Namespace lives longer
  - **`Script > Unload persistent scripts`**

Quickscript extra

A bundle of third party scripts

Sample scripts:

- Regex Replace

Quickscript extra

A bundle of third party scripts

Sample scripts:

- Regex Replace
- Table formatter

Quickscript extra

A bundle of third party scripts

Sample scripts:

- Regex Replace
- Table formatter
- Provided by

Quickscript extra

A bundle of third party scripts

Sample scripts:

- Regex Replace
- Table formatter
- Provided by
- Dynamic completion

Quickscript extra

A bundle of third party scripts

Sample scripts:

- Regex Replace
- Table formatter
- Provided by
- Dynamic completion
- Lots of snippets

Quickscript extra

A bundle of third party scripts

Sample scripts:

- Regex Replace
- Table formatter
- Provided by
- Dynamic completion
- Lots of snippets
- Reorganize tabs

Quickscript extra

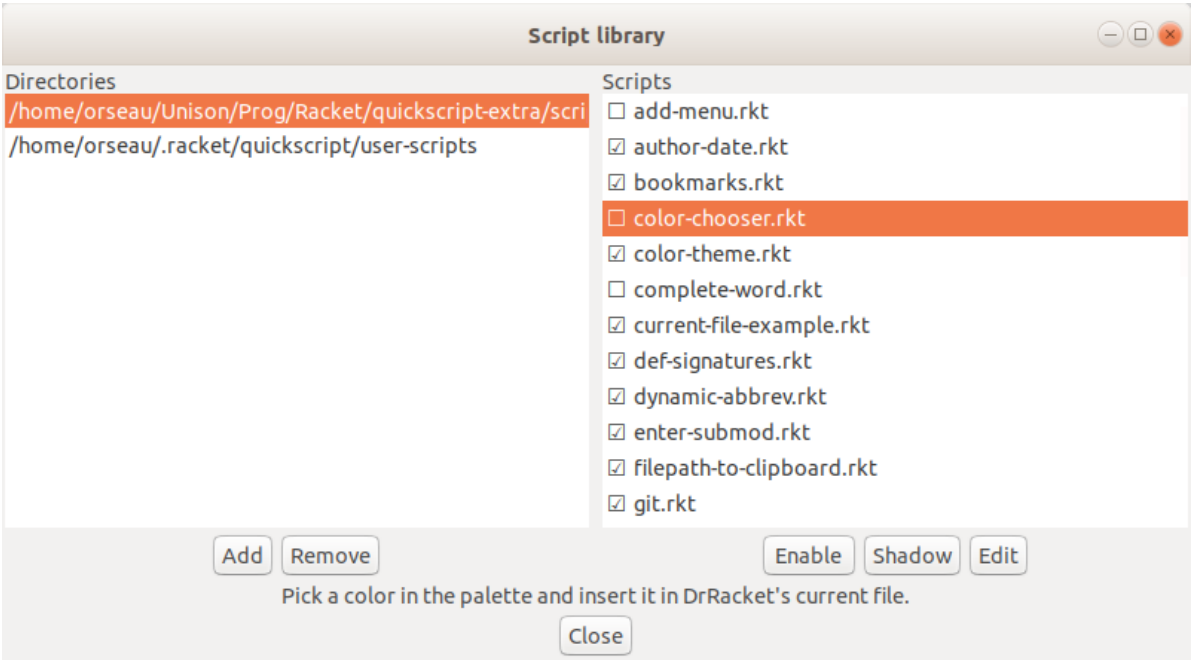
A bundle of third party scripts

Sample scripts:

- Regex Replace
- Table formatter
- Provided by
- Dynamic completion
- Lots of snippets
- Reorganize tabs
- Open terminal here
- ...

The library

Scripts > Manage scripts > Library



- Add/remove user or third-party script directories
- Enable/disable/edit scripts
- Shadow third-party scripts

Shadow scripts

Problem:

- You want to install a third-party script

Shadow scripts

Problem:

- You want to install a third-party script
- But you want to change a few things
  - Name, menu path, keyboard shortcuts

Shadow scripts

Problem:

- You want to install a third-party script
- But you want to change a few things
  - Name, menu path, keyboard shortcuts
- Obvious solution: Copy and edit script

Shadow scripts

Problem:

- You want to install a third-party script
- But you want to change a few things
  - Name, menu path, keyboard shortcuts
- Obvious solution: Copy and edit script
- But how can third-party provider maintain script?

Shadow scripts

Problem:

- You want to install a third-party script
- But you want to change a few things
  - Name, menu path, keyboard shortcuts
- Obvious solution: Copy and edit script
- But how can third-party provider maintain script?
- Solution: Shadow scripts
  - Make a local script wrapper that calls the original script

Shadow script example

In library, select target script and click on 'Shadow script

```
(require quickscript/script
         (file "path-to-script-dir/abstract-variable.rkt"))
(define-script shadow:abstract-variable
  #:label "&Abstract variable"
  #:menu-path ("Sele&ction")
  #:shortcut #f
  #:shortcut-prefix #f
  #:output-to selection
  abstract-variable)
```

Code

https://github.com/Metaxal/quickscript

https://github.com/Metaxal/quickscript-extra

```
raco pkg install quickscript
raco pkg install quickscript-extra
racket -l quickscript-extra/register
```