



Terminal Editors For Ruby Core Toolchain

ITOYANAGI Sakura

RubyKaigi 2019 Fukuoka

Greeting



Hello, everyone!

Greeting



Welcome to Fukuoka, Japan!

Let me introduce myself



I'm

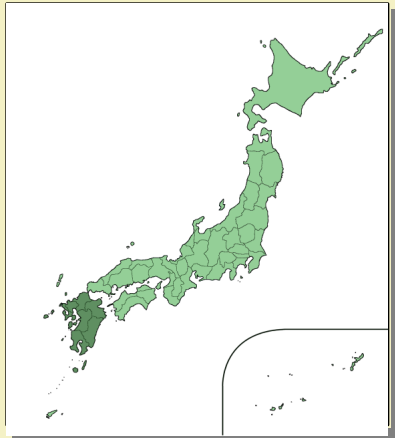
- a RDoc maintainer
- a Ruby committer
- a member of Ruby core team

Let me introduce myself



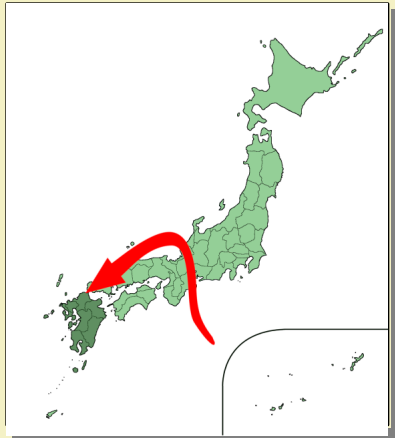
I'm so tired because this is
first session just after keynote.

Let me introduce myself



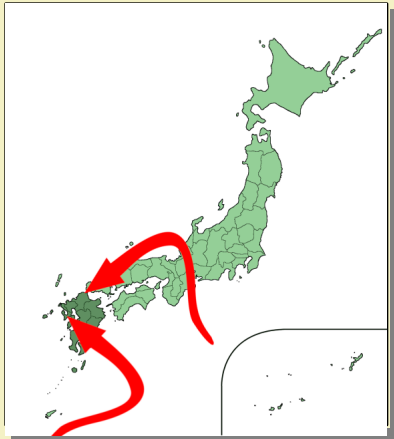
Dark green area is Kyushu

Let me introduce myself



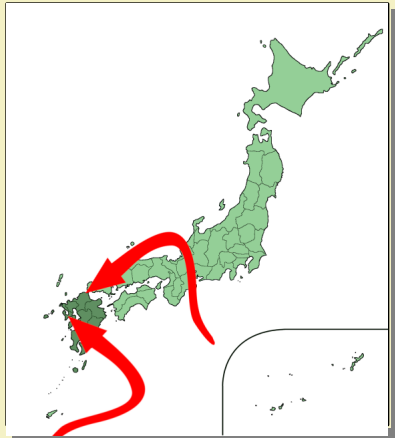
RubyKaigi venue is Fukuoka

Let me introduce myself



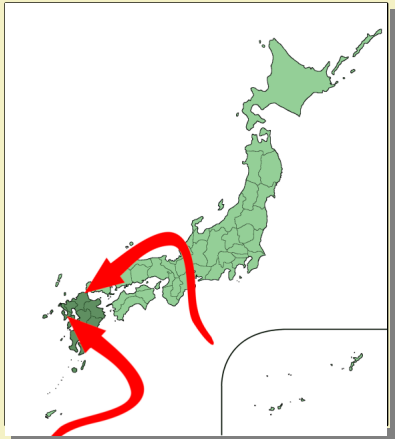
I was born at Nagasaki

Let me introduce myself



How strange shape the arrows are

Let me introduce myself



Both are almost the same culture
area

Let me introduce myself



Both eat **tonkotsu** ramen

Let me introduce myself



Black Mont Blanc

Let me introduce myself



This is the greatest soul food of
Kyushu people

Let me introduce myself



White ice cream is covered black
chocolate and crunchy chips in
perfect balance

Let me introduce myself



All children in Kyushu are raised
on Black Mont Blanc and
everyone love it, Black Mont
Blanc is nice so Kyushu is nice

Community: Asakusa.rb



Asakusa.rb is holding every
Ruby Tuesday

Community: Asakusa.rb



Many speakers of RubyKaigi
2019 are from Asakusa.rb

Company: Space Pirates, LLC.



```
# image
# src = space-pirates-logo.svg
# relative-height = 70
# caption = Space Pirates, LLC.
# relative-padding-top = 0
# relative-padding-bottom = 0
# relative-padding-right = 0
# relative-padding-left = 0
```

Company: Space Pirates, LLC.



```
# image
# src = space-pirates-logo.svg
# relative-height = 70
# caption = I'm an Elasticsearch specialist in this company
# relative-padding-top = 0
# relative-padding-bottom = 0
# relative-padding-right = 0
# relative-padding-left = 0
```

Company: Space Pirates, LLC.



```
# image
# src = space-pirates-logo.svg
# relative-height = 70
# caption = I don't know Elasticsearch but I'm using kind of go with the mood
# relative-padding-top = 0
# relative-padding-bottom = 0
# relative-padding-right = 0
# relative-padding-left = 0
```


Hobby: climbing



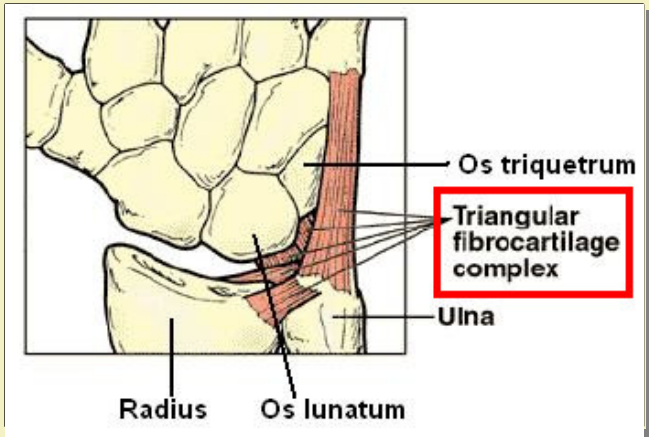
And, my hobby is climbing.

Hobby: climbing



I always go climbing when
Ruby conferences.

Hobby: climbing



But I got an injured TFCC by hard training

Hobby: climbing



Camp in a gorge at last weekend

Hobby: climbing



It's a prosthetic finger for climbing

Hobby: climbing



Climbing with prosthetic finger

Hobby: climbing



Camp in a gorge at last weekend

Hobby: climbing



Camp in a gorge at last weekend

Hobby: climbing



This type of injury has so hard problem, it's "I can't turn my humerus".

Hobby: climbing



It's natural position of humerus

Hobby: climbing



It's typing position of humerus

Hobby: climbing



"I can't turn my humerus"

VS

Typing position with turning
my humerus

Hobby: climbing



I'm still getting over the injury and I've gotten a lot better now.

Hobby: climbing



But I couldn't turn my humerus in the early days. It is effectively sentencing a programmer to death.

Hobby: climbing



In the severe situation, I felt I had to choose.

Hobby: climbing



I should:

- make a keyboard
- make an editor

Hobby: climbing



I should:

- make a keyboard
- **make an editor**

Terminal Editors For Ruby Core Toolchain

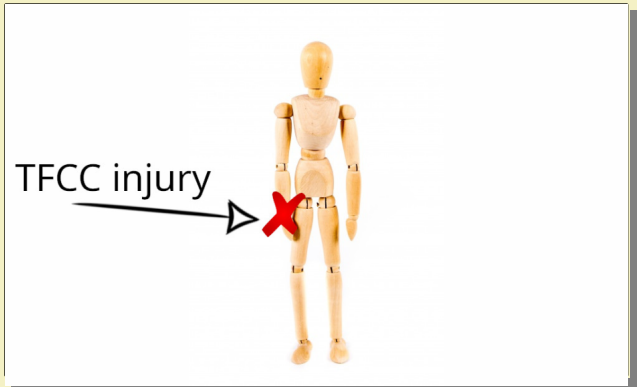


Terminal Editors For Ruby Core Toolchain



First, I got...

Terminal Editors For Ruby Core Toolchain



TFCC injury

GNU Readline



Ruby has one big problem
when installing, it's about GNU
Readline.

GNU Readline



GNU Readline is a line editor software. For example, you always use it on shell.

GNU Readline



Ruby has readline standard library, and it's based on GNU Readline as a native library.

GNU Readline



If you build and install Ruby
without installing GNU
Readline as a linkable files:

GNU Readline



- readline stdlib is nothing
- Only "input" and "backspace" are available on IRB
- Pry fails to launch

GNU Readline



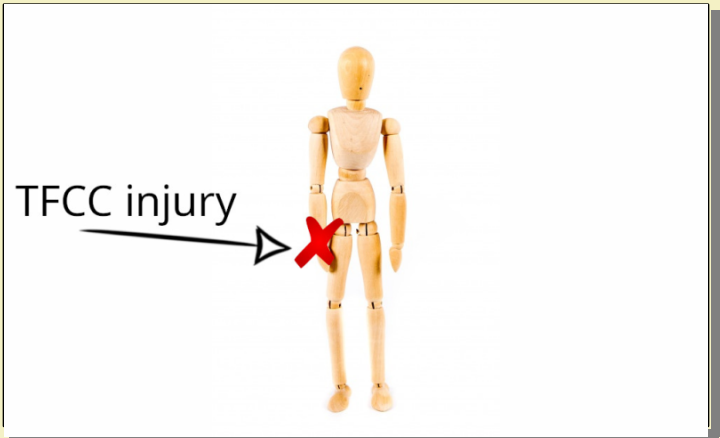
It's a very sad situation and a hard trap for beginners.

GNU Readline



And user including me needs to re-build Ruby, it means I need to type keyboard more.

GNU Readline



It's worst case for my wrist

GNU Readline



So I started to develop a GNU Readline compatible library by pure Ruby for Ruby core.

GNU Readline



I had a plan to explain
terminal technologies in this
session but now...

GNU Readline



Shugo Maeda

@shugomaeda



The Creator of Textbringer, a Ruby
committer, Director of Network
Applied Communication Laboratory,
and Secretary general of the Ruby
Association

JA

Terminal curses

Terminal programming with curses is useful and fun, but it sometimes brings terminal curses.

This talk shows basics of terminals (e.g., real text terminals, terminal emulators, the controlling terminal, `/dev/tty` and `con`, `pty`, control characters, escape sequences, `termcap/terminfo`, terminal mode), pros and cons of text-based user interfaces, an introduction to `curses.gem`, its applications, and issues you'll face when programming with curses.

Now, an unprecedented boom in
text editor.

GNU Readline



Shugo Maeda

@shugomaeda



The Creator of Textbringer, a Ruby
committer, Director of Network
Applied Communication Laboratory,
and Secretary general of the Ruby
Association

JA

Terminal curses

Terminal programming with curses is useful and fun, but it sometimes brings terminal curses.

This talk shows basics of terminals (e.g., real text terminals, terminal emulators, the controlling terminal, `/dev/tty` and `con`, `pty`, control characters, escape sequences, `termcap/terminfo`, terminal mode), pros and cons of text-based user interfaces, an introduction to `curses.gem`, its applications, and issues you'll face when programming with curses.

You can listen to terminal
techniques in this session
tomorrow.

GNU Readline



I developed **Reline** what is as
a GNU Readline
(almost)compatible library by
pure Ruby.

GNU Readline



Reline's development policies are:

- complete developing as soon as possible
 - because GNU Readline has insanely many features

GNU Readline



Reline's development policies are:

- Windows support by Win32 API
 - because it's best way to support Windows

GNU Readline



Therefore Reline uses:

- ANSI escape code on Unix like OSs
- Win32 API on Windows

GNU Readline



Therefore Reline uses:

- **ANSI escape code on Unix like OSs**
- Win32 API on Windows

ANSI escape code



ANSI escape code is a standard specification to control the cursor location, color, and other options on terminal.

ANSI escape code



If a terminal supports ANSI escape code, softwares control unified escape code.

ANSI escape code



Example:

```
puts "\e[31mred"  
puts "\e[32mgreen"  
puts "\e[34mbblue"
```


ANSI escape code



Example:

```
puts "\e[31m"      + "red"  
puts "\e[32m"      + "green"  
puts "\e[34m"      + "blue"
```

ANSI escape code



Example:

```
puts "\e[31m"      + "red"  
puts "\e[32m"      + "green"  
puts "\e[34m"      + "blue"  
#      ^color spec ^text
```

ANSI escape code



```
red  
green  
blue
```

Output should be like this

ANSI escape code



Example:

```
print "\e[#{num}A"  
# Cursor Up  
print "\e[#{num}B"  
# Cursor Down
```

ANSI escape code



Example:

```
print "\e[2K"  
# Erase in Line
```

ANSI escape code



But ANSI escape code has some problems:

- there aren't many things to be able to do
- some escape sequences are very slow grievously

ANSI escape code



For example,

```
print "\e[6n"  
# Device Status Report
```

is very slow.

ANSI escape code



If software uses Device Status Report so many times, it should slower by the by.

ANSI escape code



So terminal software with ANSI escape code should be devised within limited operations.

GNU Readline



Reline uses:

- ANSI escape code on Unix like OSs
- **Win32 API on Windows**

Win32 API



Fiddle what is for calling functions inside .dll or .so dynamic libraries is only one solution for supporting Win32 API.

Win32 API



This is "Console Functions"
page URL of Win32 API.



<https://docs.microsoft.com/en-us/windows/console/console-functions>

Win32 API



"Console Functions" is enough to do that the same of ANSI escape code.

Win32 API



- `GetStdHandle()` for console handle
- `SetConsoleCursorPosition()` is for cursor up and down
- `GetConsoleScreenBufferInfo()` is for cursor position
- blah blah blah

Hard work



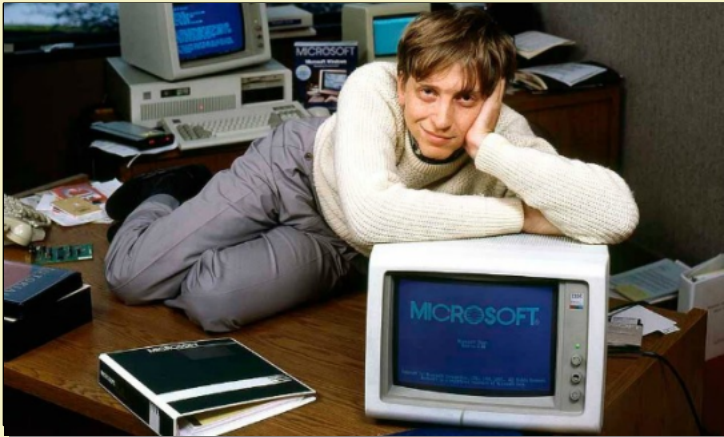
It was so hard work to complete.

Hard work



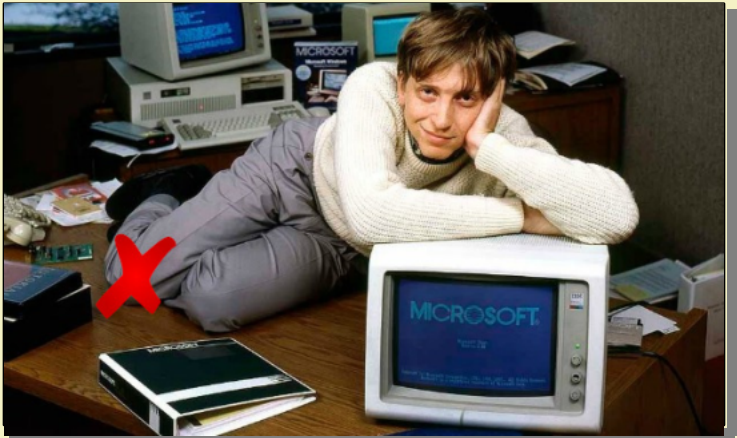
So I always do programming,
on a desk, on a bed, in trains,
in a bathroom, in a toilet...

Hard work



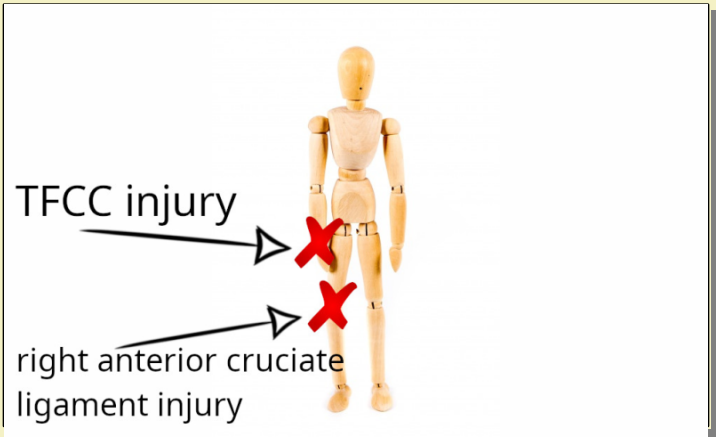
Sometimes I use computer on the floor

Hard work



I got right anterior cruciate ligament injury

Hard work



I got right anterior cruciate ligament injury

Unicode support



Multibyte characters of
Unicode has so complex
specifications:

Unicode support



- combination plural bytes to one character
- combination plural characters to one grapheme cluster
- character width depending on the situation
- blah blah blah

Unicode support



- **combination plural bytes to one character**
- **combination plural characters to one grapheme cluster**
- character width depending on the situation
- blah blah blah

I'll omit to explain these 2 specs because of too complex.

Unicode support



- combination plural bytes to one character
- combination plural characters to one grapheme cluster
- **character width depending on the situation**
- blah blah blah

Unicode support



Some Unicode characters' width are changed by the situation, for example Cyrillic alphabet.

Unicode support



I'll show 2 gnome-terminal
screenshot.

Unicode support



echo ' あ '

echo ' Д '

"Д" as a single width character

Unicode support



```
echo ' あ '
```

```
echo ' Д '
```

"Д" as a double width character

Unicode support



echo ' あ '

echo ' あ '

echo ' Д '

echo ' Д '

Both are the same strings, but
changed width of "Д"

Unicode support



echo ' あ '

echo ' あ '

echo ' д '

echo ' д '

The behaviour is based on
terminal software settings

Unicode support



How to resolve:

- show in actuality and check width and...
- **delete it before user awakes, in an eye's blink**

**Terminal
Ninja!**



Unicode support



Vim has a function,
`may_req_ambiguous_char_width()`,
the comment of it says...

Unicode support

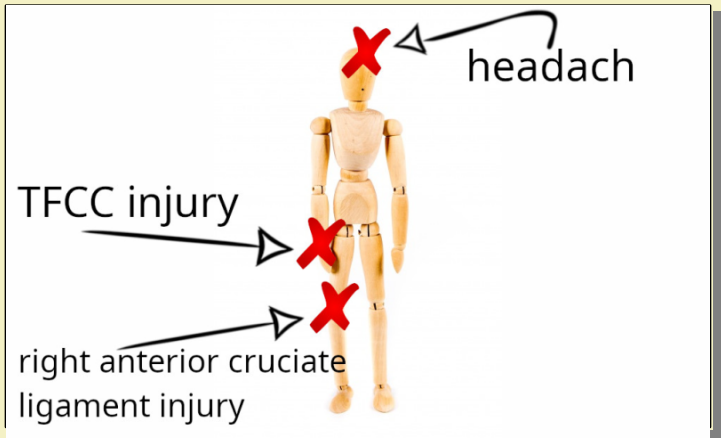


First, we move the cursor to (1, 0)
and print a test ambiguous character
`\u25bd` (WHITE DOWN-POINTING TRIANGLE)
and query current cursor position.

**COOLEST
TECH
IN
2019**



Unicode support



New headache comes, it's killing me

Unicode support



I ported it to Reline. Coolest software.

Line editing implementation

Next, let me implement line editing features to Reline.

Line editing implementation

These are key assigned operations...

Line editing implementation













- operation method list
 - `ed_insert(key)`
 - `ed_quoted_insert(str, arg: 1)`
 - `ed_next_char(key, arg: 1)`
 - `ed_prev_char(key, arg: 1)`
 - `ed_move_to_beg(key)`
 - `ed_move_to_end(key)`
 - `ed_prev_history(key, arg: 1)`
 - `ed_next_history(key, arg: 1)`

Line editing implementation

Oh, list is cut off in the middle...

Line editing implementation

operation method list(smaller)

-  `ed_insert(key)`
-  `ed_quoted_insert(str, arg: 1)`
-  `ed_next_char(key, arg: 1)`
-  `ed_prev_char(key, arg: 1)`
-  `ed_move_to_beg(key)`
-  `ed_move_to_end(key)`
-  `ed_prev_history(key, arg: 1)`
-  `ed_next_history(key, arg: 1)`
-  `ed_newline(key)`
-  `em_delete_prev_char(key)`
-  `ed_kill_line(key)`
-  `em_kill_line(key)`

Line editing implementation

Ah...

Line editing implementation

operation method list(smallest)

- `ed_insert(key)`
- `ed_quoted_insert(str, arg: 1)`
- `ed_next_char(key, arg: 1)`
- `ed_prev_char(key, arg: 1)`
- `ed_move_to_beg(key)`
- `ed_move_to_end(key)`
- `ed_prev_history(key, arg: 1)`
- `ed_next_history(key, arg: 1)`
- `ed_newline(key)`
- `em_delete_prev_char(key)`
- `ed_kill_line(key)`
- `em_kill_line(key)`
- `em_delete_or_list(key)`
- `em_yank(key)`
- `em_yank_pop(key)`

Line editing implementation

Too many

Line editing implementation

operation method list (smallest)

```
ed_insert(key)
ed_quoted_insert(str, arg: 1)
ed_next_char(key, arg: 1)
ed_prev_char(key, arg: 1)
ed_move_to_beg(key)
ed_move_to_end(key)
ed_prev_history(key, arg: 1)
ed_next_history(key, arg: 1)
ed_endline(key)
ed_delete_prev_char(key)
ed_kill_line(key)
ed_kill_line(key)
ed_delete_or_list(key)
ed_yank(key)
ed_yank_pop(key)
ed_clear_screen(key)
ed_next_word(key)
ed_prev_word(key)
ed_delete_next_word(key)
ed_delete_prev_word(key)
ed_transpose_chars(key)
ed_capitol_case(key)
ed_lower_case(key)
ed_upper_case(key)
ed_all_folding(key)
copy_for_v(text)
vi_insert(key)
vi_add(key)
vi_command_mode(key)
vi_next_word(key, arg: 1)
vi_prev_word(key, arg: 1)
vi_end_word(key, arg: 1)
vi_next_big_word(key, arg: 1)
vi_prev_big_word(key, arg: 1)
vi_end_big_word(key, arg: 1)
vi_delete_prev_char(key, arg: 1)
vi_delete_prev_char(key, arg: 1)
vi_yank(key)
vi_change_meta(key)
vi_delete_meta(key)
vi_yank(key)
vi_list_or_not(key)
ed_delete_next_char(key, arg: 1)
vi_to_history_line(key)
vi_hlitedit(key)
vi_paste_prev(key, arg: 1)
vi_paste_next(key, arg: 1)
ed_argument_digit(key)
vi_to_column(key, arg: 0)
vi_next_char(key, arg: 1)
search_next_char(key, arg)
```

Line editing implementation

GNU Readline features are:

Line editing implementation

- emacs mode
 - kill-ring
 - yank, yank-pop
- vi mode
 - argumented operations
 - combination of operation and motion
 - undo
- setting files
 - key binding
 - macro
- blah blah blah

**It means
that I made
almost full
2 editors**



Line editing implementation

Demonstration

Multiline editor



Multiline editor



Today's description of this session explains about "Reidline".

Multiline editor



"Reidline" is authored for new IRB by keiju-san who is Ruby's grandfather, it behaves as a multiline editor like JavaScript console on browsers.

It had many technical problems but I've already solved that when I implemented Reline.

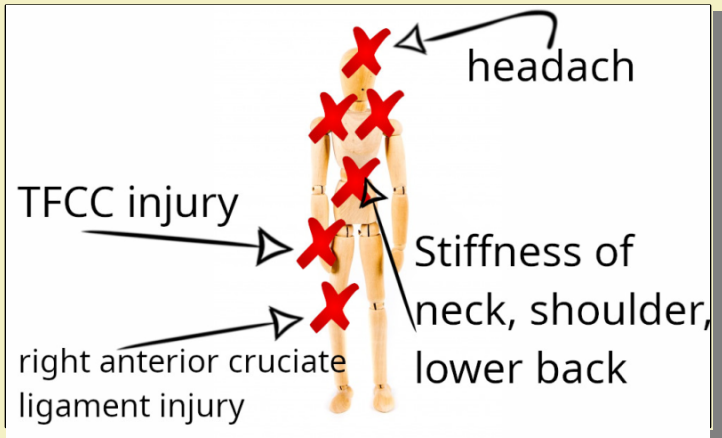
So I helped to complete Reidline.

Multiline editor



Therefore, I improved Reline
for Reidline. It supports
multiline a few days ago!


Multiline editor



Stiffness of neck, shoulder, lower back

Multiline editor





JA


Terminal curses

Terminal programming with curses is useful and fun, but it sometimes brings terminal curses.

This talk shows basics of terminals (e.g., real text terminals, terminal emulators, the controlling terminal, /dev/tty and con, pty, control characters, escape sequences, termcap/termio, terminal mode), pros and cons of text-based user interfaces, an introduction to curses.gem, its applications, and issues you'll face when programming with curses.

Shugo Maeda
@shugomada

The Creator of Textbringer, a Ruby committer, Director of Network Applied Communication Laboratory, and Secretary general of the Ruby Association




EN

Terminal Editors For Ruby Core Toolchain

I implemented "Reline" that is a compatibility library with "readline" stdlib by pure Ruby and works correctly without GNU Readline and works on Windows too.

"Reline" is authored for new IRB by keiju-san who is Ruby's grandfather. It behaves as a multiline editor like JavaScript console on browsers. It had many technical problems but I've already solved that when I implemented Reline. So I helped to complete Reidline.

ITOYANAGI Sakura
@eyecats



EN

Terminal Editors For Ruby Core Toolchain

I implemented "Reline" that is a compatibility library with "readline" stdlib by pure Ruby and works correctly without GNU Readline and works on Windows too.

"Reline" is authored for new IRB by keiju-san who is Ruby's grandfather. It behaves as a multiline editor like JavaScript console on browsers. It had many technical problems but I've already solved that when I implemented Reline. So I helped to complete Reidline.

ITOYANAGI Sakura
@eyecats

There are 3 editors by Ruby

Multiline editor



And, I'm the current RDoc maintainer. So I added new feature that shows document after completion.

Multiline editor



Demonstration

Multiline editor



I started development of editor for my wrist, but crash of my body is continued.

Multiline editor



Incredible situation.

Multiline editor



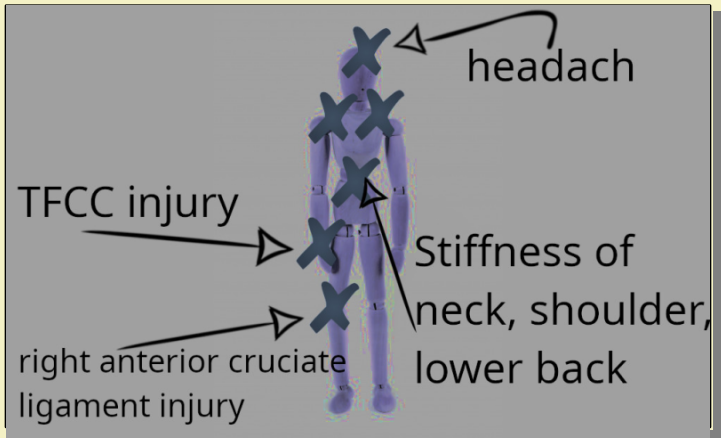
And, at 3rd day of RubyKaigi,
Ruby 2.7.0-preview1 will be
released, with Reline and
Reidline.

Multiline editor



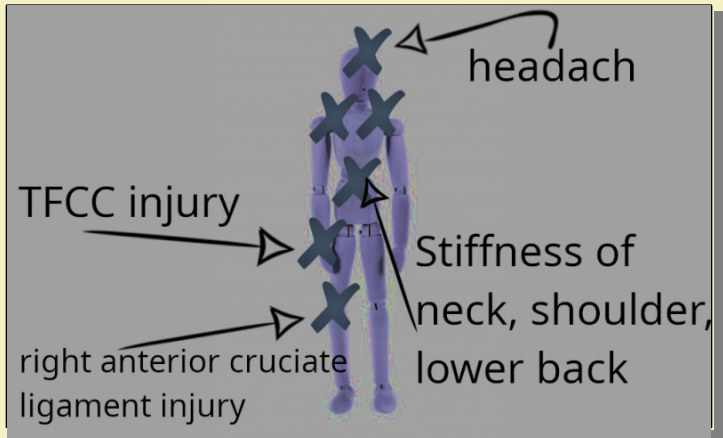
So I should fix all bugs for the day.

Multiline editor



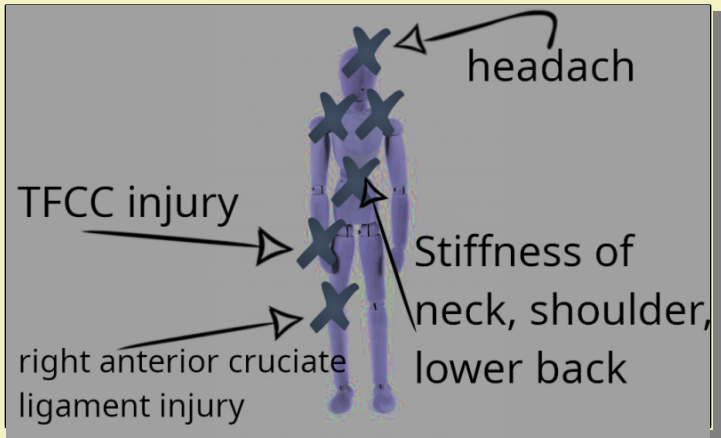
My body will be gone

Multiline editor



This is my last work of Heisei era

Multiline editor



C'mon, Reiwa era...