# OhMyREPL.jl

*This is my REPL. There are many like it, but this one is mine.*


## Kristoffer Carlsson

https://kristofferc.github.io

kristoffer.carlsson@chalmers.se

@KristofferC

@KristofferC89

# Outline

- The Julia REPL (0.6)
- OhMyREPL.jl
- Demo
- Summary

# The Julia REPL

# The Julia REPL

- REPL - Read Eval Print Loop
- Written in Julia since #6270 (Mar 2014)
- Features
  - History
  - Search mode (reverse / forward)
  - Keybindings (customizable)
  - Tab completion (Unicode, fields)
  - Extensible (`RCall.jl` REPL mode)
  - Some color customization (prompt color, text color)

# The Julia REPL 0.6

## Prompt pasting (#17599)

- Common with code snippets starting with the `julia> prompt`
  - Doctests
  - Code copied from a REPL session

```
julia> Char(0x110000)
'\U110000': Unicode U+110000 (category Cn: Other, not assigned)

julia> isvalid(Char, 0x110000)
false
```

- Annoying to manually have to scrub code before executing it
- Now, automatically detect the `julia>` prompt when pasting, remove the prompt and output

# The Julia REPL

# 0.6

New stack traces (#19569)

– Old:

```
julia> test_error()
ERROR: BoundsError: attempt to access 5-element Array{Float64,1}:
 0.346979
 0.949856
 0.788565
 0.198736
 0.387921
  at index [6]
 [inlined code] from ./random.jl:329
 in ff(::Int64) at ./none:1
 in g(::ThisisALongTypeThatWeMightNotWantToSee, ::ThisisALongTypeThatWeMightNotW
antToSee, ::Int64, ::Float64, ::Float64, ::ThisisALongTypeThatWeMightNotWantToSe
e, ::Int64) at ./none:3
 in g(::ThisisALongTypeThatWeMightNotWantToSee, ::ThisisALongTypeThatWeMightNotW
antToSee, ::Int64, ::Float64, ::Float64, ::ThisisALongTypeThatWeMightNotWantToSe
e, ::Int64) at ./none:5 (repeats 4 times)
 in h(::Float64) at ./none:1
 in test_it(::Int64, ::Vararg{Int64}) at ./none:2
 in test_error() at ./none:1
 in eval(::Module, ::Any) at ./boot.jl:243
```

# The Julia REPL

# 0.6

New stack traces (#19569)

- New:

```
julia> test_error()
ERROR: BoundsError: attempt to access 5-element Array{Float64,1} at index [6]
Stacktrace:
 [1] ff(::Int64) at ./REPL[10]:1
 [2] g(::ThisisALongType{Float64,Int32}, ::ThisisALongType{Float64,Int32}, ::Int
64, ::Float64, ::Float64, ::ThisisALongType{Float64,Int32}, ::Int64) at ./REPL[1
1]:3
 [3] g(::ThisisALongType{Float64,Int32}, ::ThisisALongType{Float64,Int32}, ::Int
64, ::Float64, ::Float64, ::ThisisALongType{Float64,Int32}, ::Int64) at ./REPL[1
1]:5 (repeats 4 times)
 [4] h(::Float64) at ./REPL[12]:1
 [5] test_error() at ./REPL[14]:1
```

- Numbered list of stackframes
- Open editor at stackframe with shortcut `Ctrl +Q`

# The Julia REPL 0.6

New stack traces (#19569)

– New:

```
julia> test_error()
ERROR: BoundsError: attempt to access 5-element Array{Float64,1} at index [6]
Stacktrace:
 [1] ff(::Int64) at ./REPL[10]:1
 [2] g(::ThisisALongType{Float64,Int32}, ::ThisisALongType{Float64,Int32}, ::Int
64, ::Float64, ::Float64, ::ThisisALongType{Float64,Int32}, ::Int64) at ./REPL[1
1]:3
 [3] g(::ThisisALongType{Float64,Int32}, ::ThisisALongType{Float64,Int32}, ::Int
64, ::Float64, ::Float64, ::ThisisALongType{Float64,Int32}, ::Int64) at ./REPL[1
1]:5 (repeats 4 times)
 [4] h(::Float64) at ./REPL[12]:1
 [5] test_error() at ./REPL[14]:1
```

```
ENV["JULIA_STACKFRAME_LINEINFO_COLOR"] = :cyan;
ENV["JULIA_STACKFRAME_FUNCTION_COLOR"] = :yellow;
```

# OhMyREPL.jl

# OhMyREPL.jl

*Passes* - Modify color/style of entered text

- Syntax Highlighting
- (Active) Bracket Highlighting
- Rainbow brackets

```
f(x::String) = x * "foo"
( [  ]   )
(( [[[   ]  ]]  {  ) } )
```

*Other*

- Bracket completion
- Prompt changing

```
0.6.0> f(x) = x^2
>> f (generic function with 1 method)

0.6.0> f(5)
>> 25
```

# Passes

- What is needed (e.g for syntax highlighter)?
  - Tokenizer / Lexer - `Tokenize.jl`[1]
  - Simple interface for colors in terminal - `Crayons.jl`[2]

[1]: https://github.com/KristofferC/Tokenize.jl

[2]: https://github.com/KristofferC/Crayons.jl

# Tokenize.jl

- Tokenization: text (source code) -> "words with meaning"
- Interactive usage:
  - non error throwing
- Roundtrippable:
  - keep whitespace
- Backend for
  - `CSTParser.jl` (@ZacLN):

  VSCode Julia plugin:
  - Juno (Atom IDE):

  Module detection

```
julia> collect(tokenize("""
       function f()
           return 1.1.1
       end"""))
13-element Array{Tokenize.Tokens.Token,1}:
 1,1-1,8          KEYWORD        "function"
 1,9-1,9          WHITESPACE     " "
 1,10-1,10        IDENTIFIER     "f"
 1,11-1,11        LPAREN         "("
 1,12-1,12        RPAREN         ")"
 1,13-2,4         WHITESPACE     "\n    "
 2,5-2,10         KEYWORD        "return"
 2,11-2,11        WHITESPACE     " "
 2,12-2,15        ERROR          "1.1."
 2,16-2,16        INTEGER        "1"
 2,17-3,0         WHITESPACE     "\n"
 3,1-3,3          KEYWORD        "end"
 3,4-3,3          ENDMARKER      ""
```

# Crayons.jl

- Colored / styled text in terminal is done by printing special "ANSI Codes"

```
julia> print("\e[34mBLUE TEXT")
BLUE TEXT
```

- Nice to have a higher level system for terminal colors than raw strings.

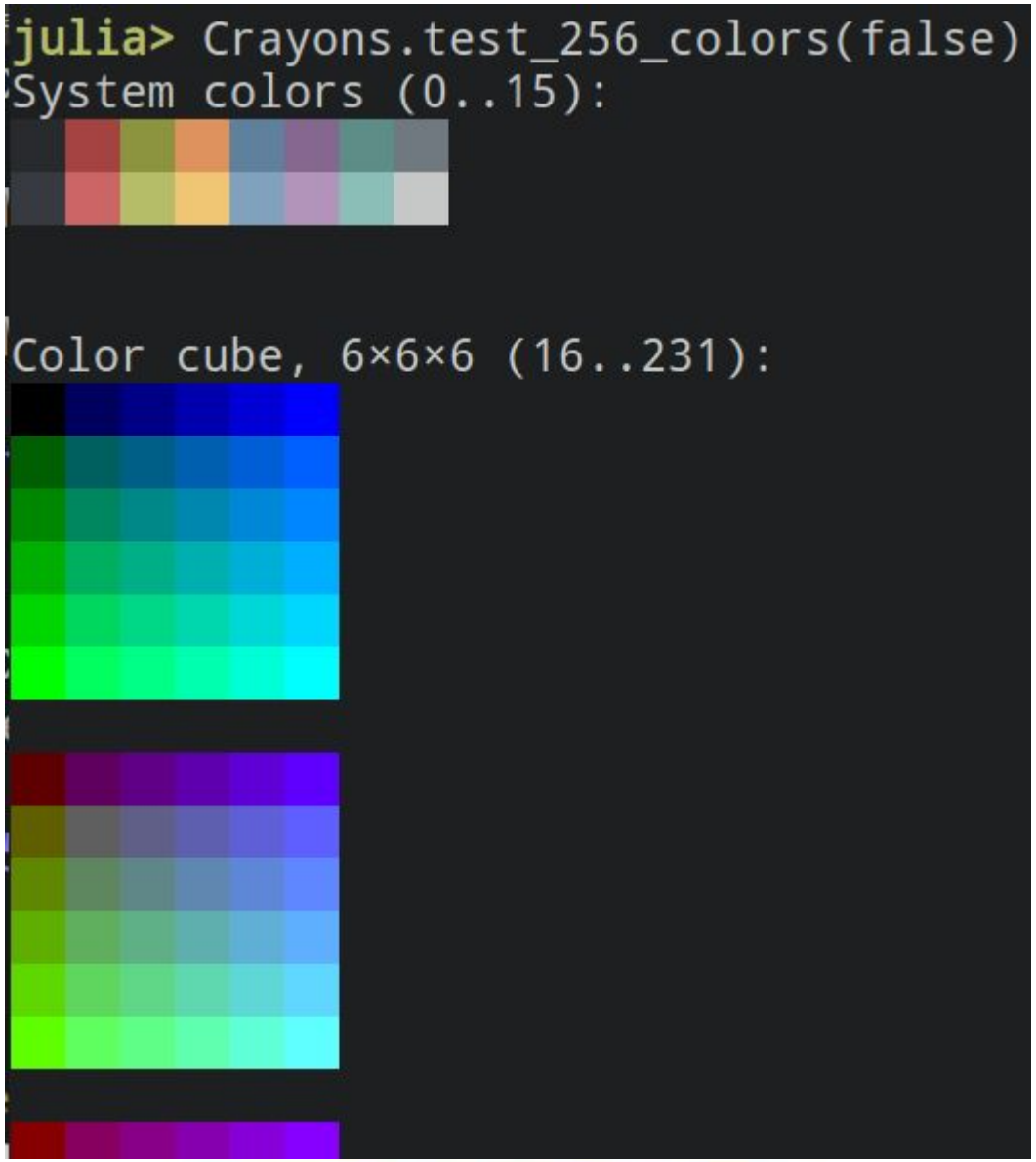
# Crayons.jl

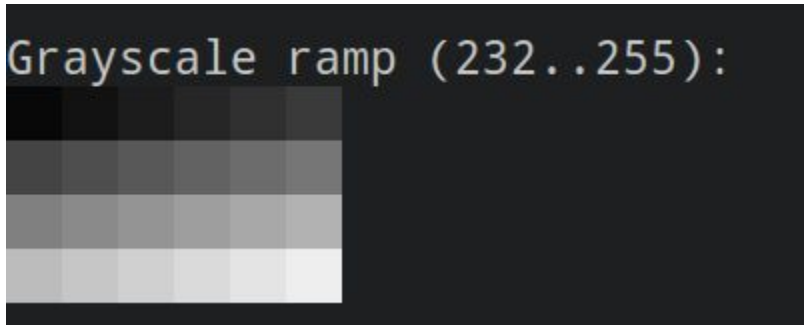
```
julia> c_red = Crayon(foreground = :light_red, bold = true)
\e[91;1m

julia> c_red("Red and bold")
Red and bold

julia> c_green = Crayon(foreground = (0, 255, 0), underline = true)
\e[38;2;0;255;0;4m

julia> c_red("Red", c_green(" and green "), "and red again")
Red and green and red again

julia> c_green * c_red
\e[91;1;4m
```

# Crayons.jl

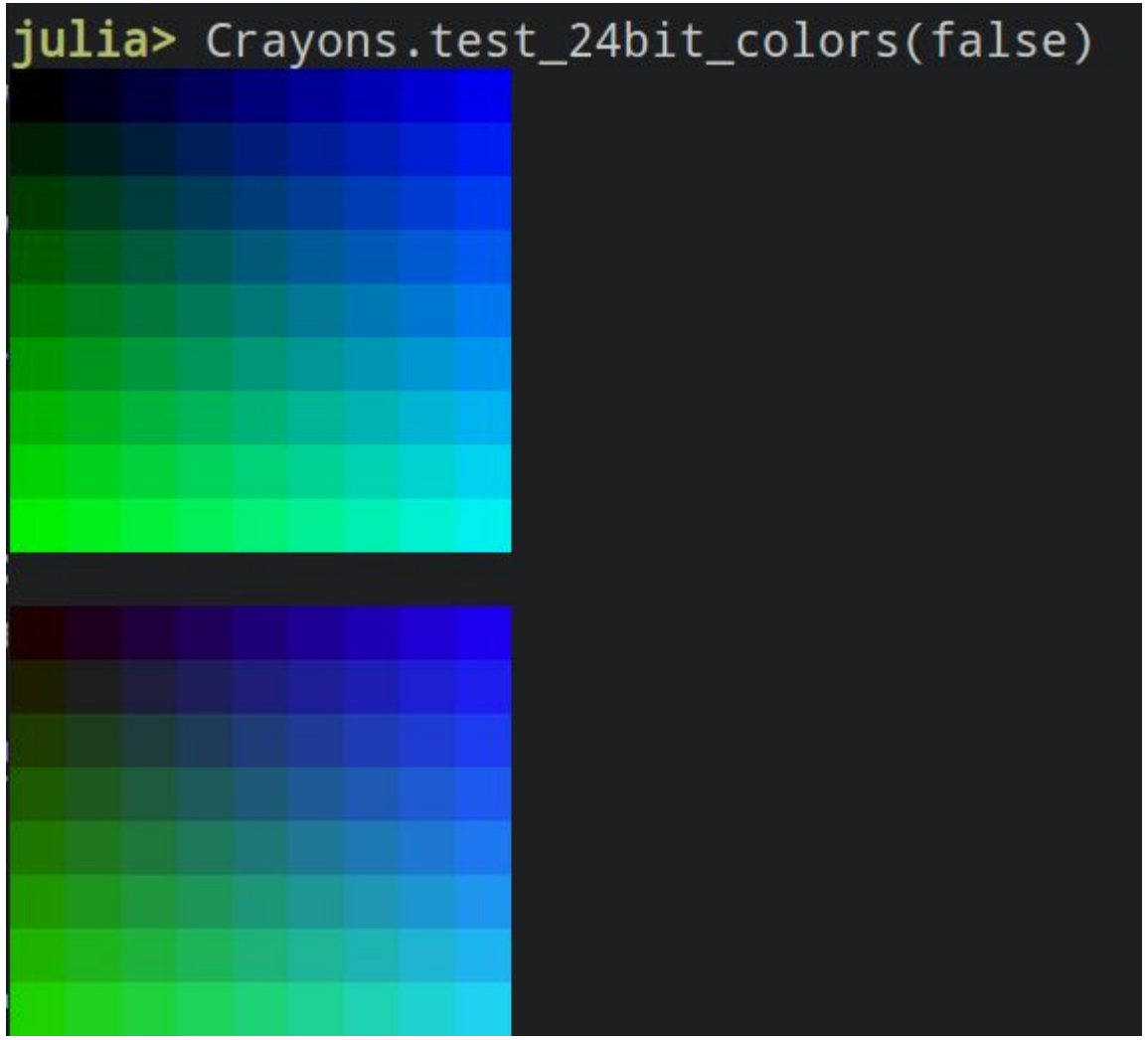
```
julia> Crayons.test_system_colors()
        default default
          green green
     light_cyan light_cyan
          white
           blue blue
  light_magenta light_magenta
      dark_gray dark_gray
      light_red light_red
   light_yellow light_yellow
           cyan cyan
     light_gray light_gray
     light_blue light_blue
         yellow yellow
        magenta magenta
            red red
          black black
    light_green light_green
```

# Crayons.jl


julia> Crayons.test_256_colors(false)
System colors (0..15):
Color cube, 6×6×6 (16..231):


Grayscale ramp (232..255):

# Crayons.jl



# Crayons.jl

```
julia> Crayons.test_styles()
Printed with bold = true
Printed with faint = true
Printed with italics = true
Printed with underline = true
Printed with blink = true
Printed with negative = true
                            <- This is concealed = true
Printed with strikethrough = true
```

## Input string

```
function f(x::Float64)
    return sqrt.([x; [1,2]])
end
```

## Cursor position

41

## Tokenization (`Tokenize.jl`)

```
18-element Array{Tokenize.Tokens.Token,1}:
 1,1-1,8          KEYWORD        "function"
 1,9-1,9          WHITESPACE     " "
 1,10-1,10        IDENTIFIER     "f"
 1,11-1,11        LPAREN         "("
 1,12-1,12        IDENTIFIER     "x"
 1,13-1,14        OP             "::"
 1,15-1,21        IDENTIFIER     "Float64"
 1,22-1,22        RPAREN         ")"
 ...
 3,4-3,3          ENDMARKER      ""
```

## Run passes (`Crayons.jl`)

Syntax highlighting

```
function f(x::Float64)
    return sqrt.([x; [1,2]])
end
```

Active brackets

```
function f(x::Float64)
    return sqrt.([x; [1,2]])
end
```

Rainbow brackets

```
function f(x::Float64)
    return sqrt.([x; [1,2]])
end
```

## Merging

```
function f(x::Float64)
    return sqrt.([x; [1,2]])
end
```

# Demo

# Summary / Acknowledgments

- Presented OhMyREPL.jl, package to customize the REPL.
- Two packages split out from package:
  - `Tokenize.jl` - Lexing of Julia code
  - `Crayons.jl` - Colors in terminal
- Thanks to:
  - Zac Nugent `@ZacNL` & Sebastian Pfitzner `@pfitzseb` for bugfixes to `Tokenize.jl`
  - Everyone who filed issues / PRs for my packages

# Thank you!
# Questions / Comments?