

```

use regex::Regex;
use lazy_regex::{regex, regex_is_match};

fn fib(n: usize) -> usize {
    if n < 2 {
        n
    } else {
        fib(n - 1) + fib(n - 2)
    }
}

fn main() {
    Regex::new(r"[a-zA-F0-9_]\s(.*)$");
    let a = regex!(r"[a-zA-F0-9_]\s(.*)$");
    if regex_is_match!(/* comment */ r"[a-zA-F0-9_]\s(.*)$"i, r"raw text \s[a-f]") {
        return;
    }
}

import "fmt"

// comment
func Main() {
    fmt.Println("Hello, World!")
}

```

Inline code is also supported: `fn main() {}`. This may be useful to reference types like `i32` or functions like `foo()` or even things like regular expressions `'[a-zA-F0-9_]\s(.*)$'` in text.

Languages that syntastica doesn't support will continue to be highlighted by Typst's native highlighting logic (using syntect)

= Chapter 1

```

#let hi = "Hello World"

def fib(n):
    if n < 0:
        return None
    if n == 0 or n == 1:
        return n
    return fib(n-1) + fib(n-2)

```

You can also combine `syntastica` with other show rules. Here is the RegEx

`[a-zA-F0-9_]\s(.*)$` again.

```

.intel_syntax
.global _start

.section .text

_start:
    call    main..main
    mov     %rdi, 0
    call    exit

main..main:
    push    %rbp
    mov     %rbp, %rsp

```

```

sub    %rsp, 32
mov    qword ptr [%rbp-8], 3
lea    %rax, qword ptr [%rbp-8]
mov    qword ptr [%rbp-16], %rax
lea    %rax, qword ptr [%rbp-16]
mov    qword ptr [%rbp-24], %rax
mov    %rax, qword ptr [%rbp-24]
mov    %rax, qword ptr [%rax]
mov    qword ptr [%rbp-32], %rax
mov    %rdi, qword ptr [%rbp-24]
mov    %rdi, qword ptr [%rdi]
mov    %rdi, qword ptr [%rdi]
mov    %rsi, qword ptr [%rbp-24]
mov    %rsi, qword ptr [%rsi]
mov    %rsi, qword ptr [%rsi]
call   __rush_internal_pow_int
mov    %rdi, %rax
mov    %rax, qword ptr [%rbp-32]
mov    qword ptr [%rax], %rdi
mov    %rdi, qword ptr [%rbp-24]
mov    %rdi, qword ptr [%rdi]
mov    %rdi, qword ptr [%rdi]
call   exit
main..main.return:
leave
ret

```

It can also be used in combination with `algo` for even prettier listings:

```

1 fn fib(n: usize) -> usize {
2   |   if n < 2 {
3     |     n
4   |   } else {
5     |     fib(n - 1) + fib(n - 2)
6   |   }
7 }

```