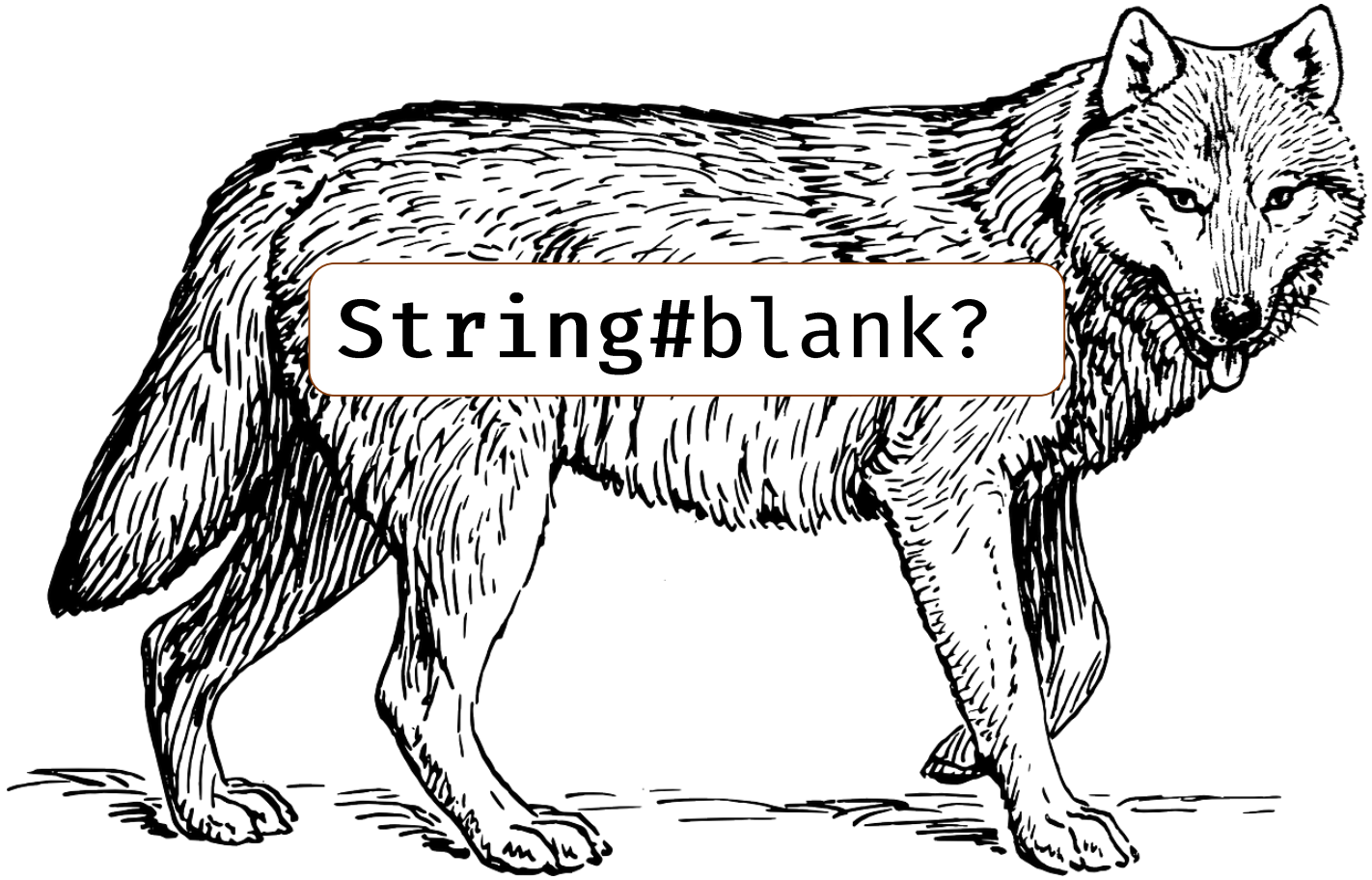
# Es war einmal...

```
def blank?
    /\A[[:space:]]*\z/ === self
end
```




```ruby
def blank?
  /\A[[:space:]]*\z/ === self
end
```

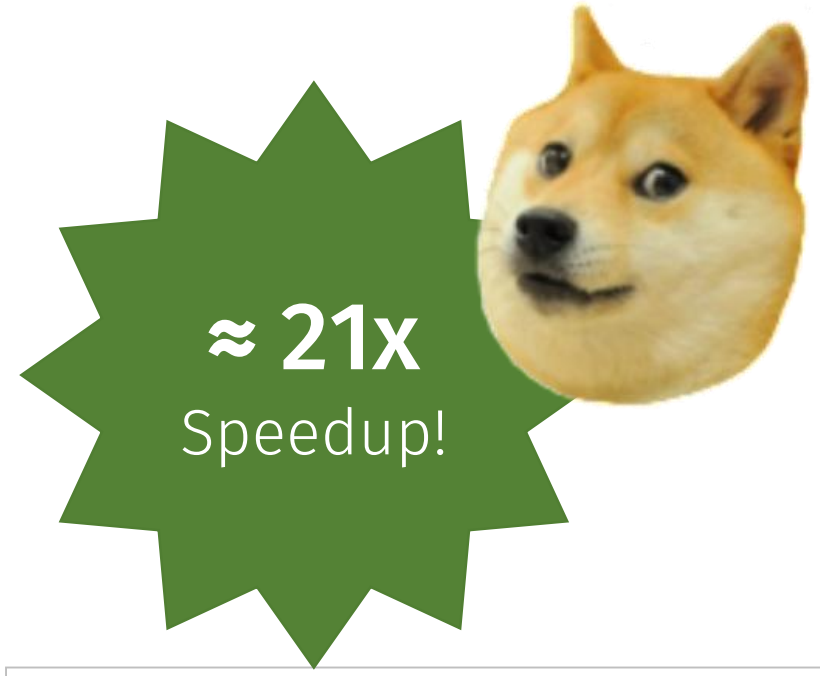
```c
static VALUE
rb_str_blank(VALUE str)
{
  rb_encoding *enc;
  char *s, *e;

  enc = STR_ENC_GET(str);
  s = RSTRING_PTR(str);
  if (!s || RSTRING_LEN(str) == 0) return Qtrue;

  e = RSTRING_END(str);
  while (s < e) {
    int n;
    unsigned int cc = rb_enc_codepoint_len(s, e, &n, enc);

    switch (cc) {
      case 9: case 0xa: case 0xb: case 0xc: case 0xd: case 0x20:
      case 0x85: case 0xa0: case 0x1680: case 0x2000: case 0x2001:
      case 0x2002: case 0x2003: case 0x2004: case 0x2005: case 0x2006:
      case 0x2007: case 0x2008: case 0x2009: case 0x200a: case 0x2028:
      case 0x2029: case 0x202f: case 0x205f: case 0x3000:
        /* found */
        break;
      default:
        return Qfalse;
    }
    s += n;
  }
  return Qtrue;
}
```

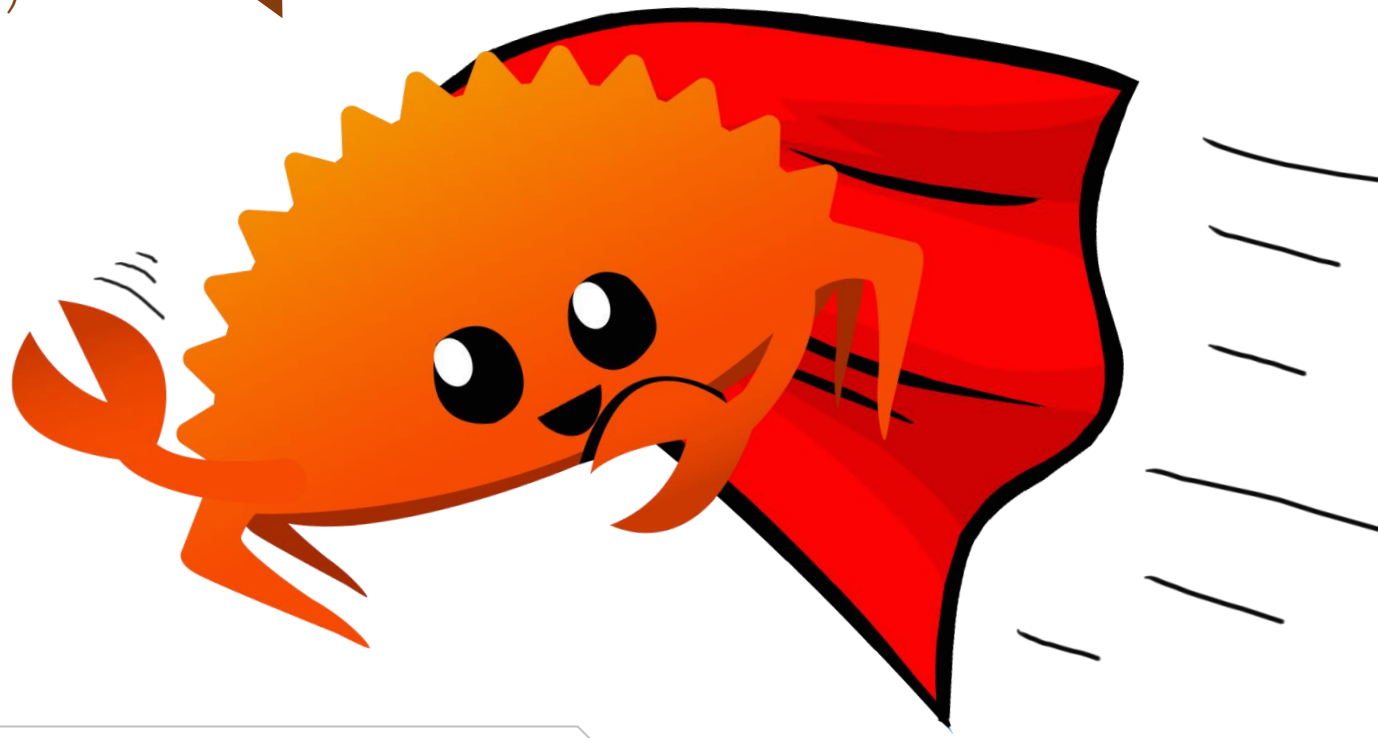
**≈ 20x**
Speedup!

(vgl. [2])

# Ferris

(Rust-Maskottchen)

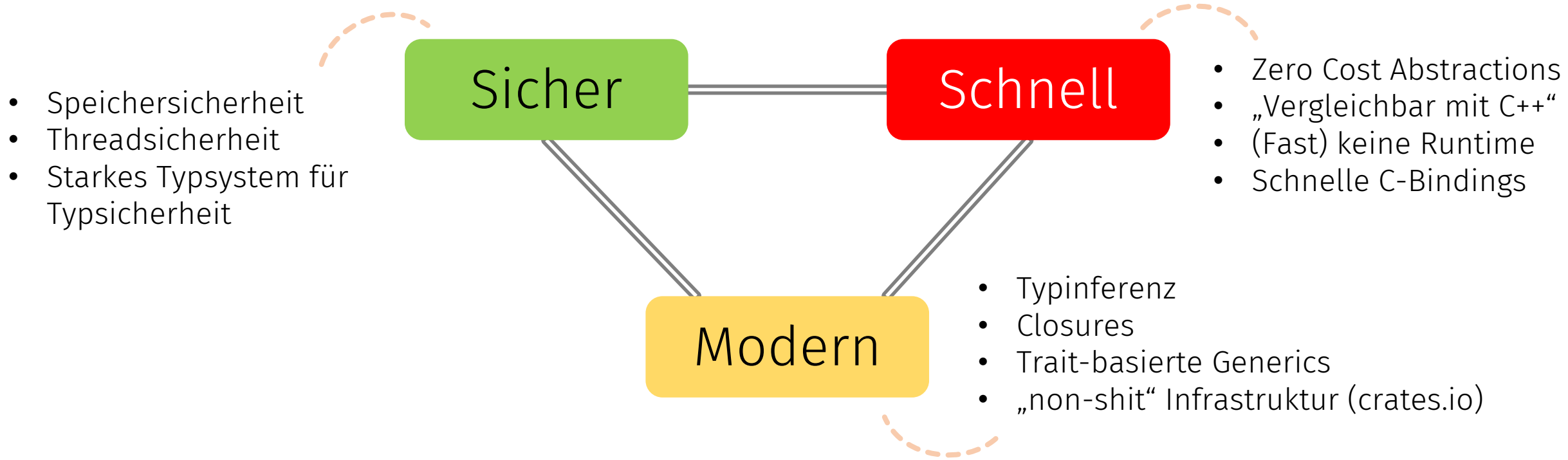


```
pub extern "C" fn fast_blank(buf: Buf) -> bool {
    buf.as_slice().chars().all(|c| c.is_whitespace())
}
```

“**Rust** is a systems programming language that runs blazingly fast, prevents segfaults, and guarantees thread safety.”

— Rust Website [3]



# **Rust:** „Hack without Fear“

- Speichersicherheit, d.h. kein:
  - „*use after free*“ oder „*double free*“
  - Zugriff auf nicht-initialisierten Speicher
  - *Dangling pointer*
  - `null` Dereferenzierung
  - *Undefined behavior*
- Keine *Data Races*

(Fast) alle Garantien zur Kompilierzeit!

- Weniger Debugging
- Keine klassischen Sicherheitslücken
- Höhere Zuverlässigkeit

Gut für *High- und Low-*Level Programmierer!

# Referenzen & Links


[1] „Introducing Helix“, http://blog.skylight.io/introducing-helix/

[2] „Why Rust?“, http://www.integer32.com/2016/07/11/why-rust.html

> „I felt like I would have to study for years before I could be trusted to write production C without security holes and memory leaks everywhere. I feel confident and empowered programming in Rust.”

[3] Rust Website, https://www.rust-lang.org

[4] „Why Rust?“, O'Reilly, http://www.oreilly.com/programming/free/files/why-rust.pdf

[5] Rust Maskottchen, http://www.rustacean.net/

[6] „It's been fun and frustrating learning Rust.”,
http://jimfulton.info/site/2016/Sep/25/experiment-compare-zodb-file-storage-iteration-with-python-and-rust/

[Rotkäppchen Bild] http://jobolino.eu/index.php?id_product=268&controller=product
[Jäger Bild] von Kenzie Lamar

# Ziele dieses Kurses

## Rust

(duh)

- Fast alle Features kennenlernen
- Gelerntes durch Aufgaben vertiefen
- Rust evtl. als Standardsprache

➔ *Selbstsicher* programmieren

## Konzepte & Grundlagen

- Stack, Heap, Pointer, …
- Geschwindigkeitsoptimierung
- …

„[Rust] has helped me to understand how to use pointers more than any other explanation I've heard." [2]

Ziele, keine Versprechen ;-)