# Quo Vadis?

Andrei Alexandrescu

Research Scientist

Facebook

# aspirations

# Principled & Practical

- Slices
- Approach to modularity
- Support for generic programming
- Error handling done right: `scope`
- Qualifiers
- Approach to `@safe`ty

# “Meh” features

- `@property`
- `synchronized`’s interplay with `shared`
- `delete`
- `foreach_reverse`
- qualified postblit

# Features that don't exist (but work)

- Attribute inference
- Compile-Time Function Evaluation
- Scoped `imports`
- Value Range Propagation (no-pain conversions)
- Relaxed purity

# Functional Factorial (yawn)

```
ulong factorial(uint n) {
    return n <= 1 ? 1 : n * factorial(n - 1);
}
```

- It's PSPACE!
- Somebody should do hard time for this

# However, it's pure

```
pure ulong factorial(uint n) {
    return n <= 1 ? 1 : n * factorial(n - 1);
}
```

- Pure is good

# Functional Factorial, Fixed

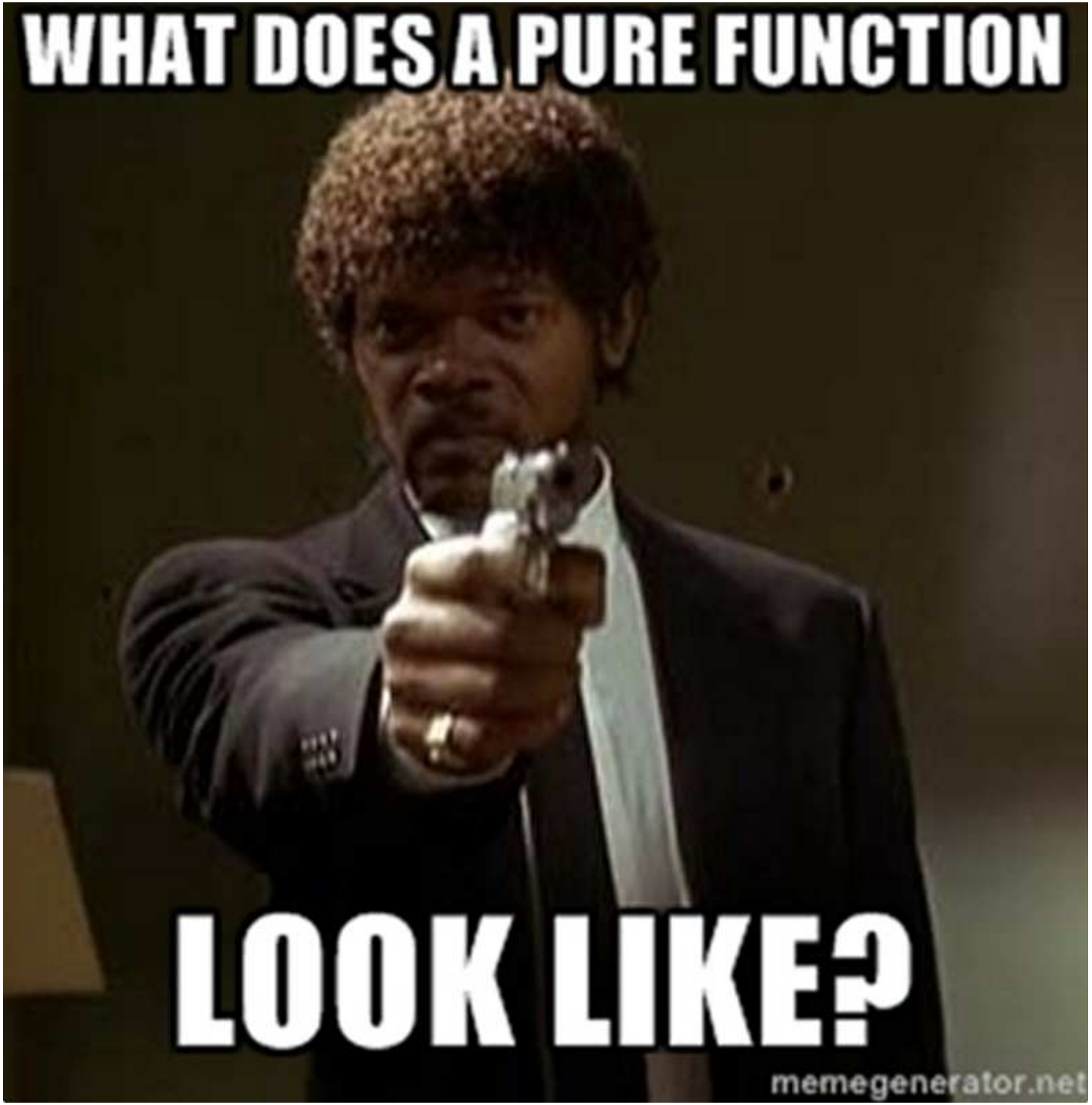
```
pure ulong factorial(uint n) {
    ulong crutch(uint n, ulong result) {
        return n <= 1
            ? result
            : crutch(n - 1, n * result);
    }
    return crutch(n, 1);
}
```

- Threads state through as parameters
- You know what? I don't care for it

# Honest Factorial

```
ulong factorial(uint n) {
    ulong result = 1;
    for (uint i = 2; i <= n; ++i) {
        result *= i;
    }
    return result;
}
```

- But no longer pure!
- Well allow me to retort


WHAT DOES A PURE FUNCTION
LOOK LIKE?
memegenerator.net

# Pure is as pure does

- “Pure functions always return the same result for the same arguments”
- No reading and writing of global variables
  - (Global *immutables* okay)
- No calling of im`pure` functions
- Who said anything about local, transient state *inside the function?*

# Transitive State

```
pure void reverse(T)(T[] a) {
  foreach (i; 0 .. data.length / 2) {
    swap(data[i], data[$ - i - 1]);
  }
}
```

- Possibility: disallow
- More useful: relaxed rule
- Operate with transitive closure of state reachable through parameter
- Not functional pure, but an *interesting superset*
- No need for another annotation, it's all in the signature!

# User-defined types

```
pure BigInt factorial(uint n) {
    BigInt result = 1;
    foreach (i; 1 .. n + 1) {
        result *= i;
    }
    return result;
}
```

- Works, but not in released version
- Not all stdlib “purified” yet

# Aftermath

- If parameters reach mutable state:
  - Relaxed pure—no globals, no I/O, no impure calls
- If parameters can't reach mutable state:
  - "Haskell-grade" *observed* purity
  - Yet imperative implementation possible
  - As long as it's local only

comprehensive != big

# comprehensive != perfect

# Allocators

- We want a comprehensive design
- (not a big/complex/perfect one)
- It should work

# Allocation Archetypes

- Garbage collected
- Garbage collected + `free`
- `malloc`-based
- Region-based

# Composability

- Allocators must stack efficiently
- Example: freelist over region
- (Related work: HeapLayers)

# Safety

- Some allocators safe (GC)
- Some unsafe (`malloc`)
- Some can be made safe (regions)

image courtesy of Lockheed Martin

mando gomez | mandolux@yahoo.com

# vision

# Fable

*Neophyte:* “What should I do to be a better person?”

# Fable

*Guru:* “Do what a good person does.”

# Applied

*We:* “What should we do to scale to 1M users?”

# That means

- Stability
- Quality, quality, quality
- Expanding platform base
- Operational professionalism

# Stability

# Quality

# Expanding platform base

# Operational professionalism

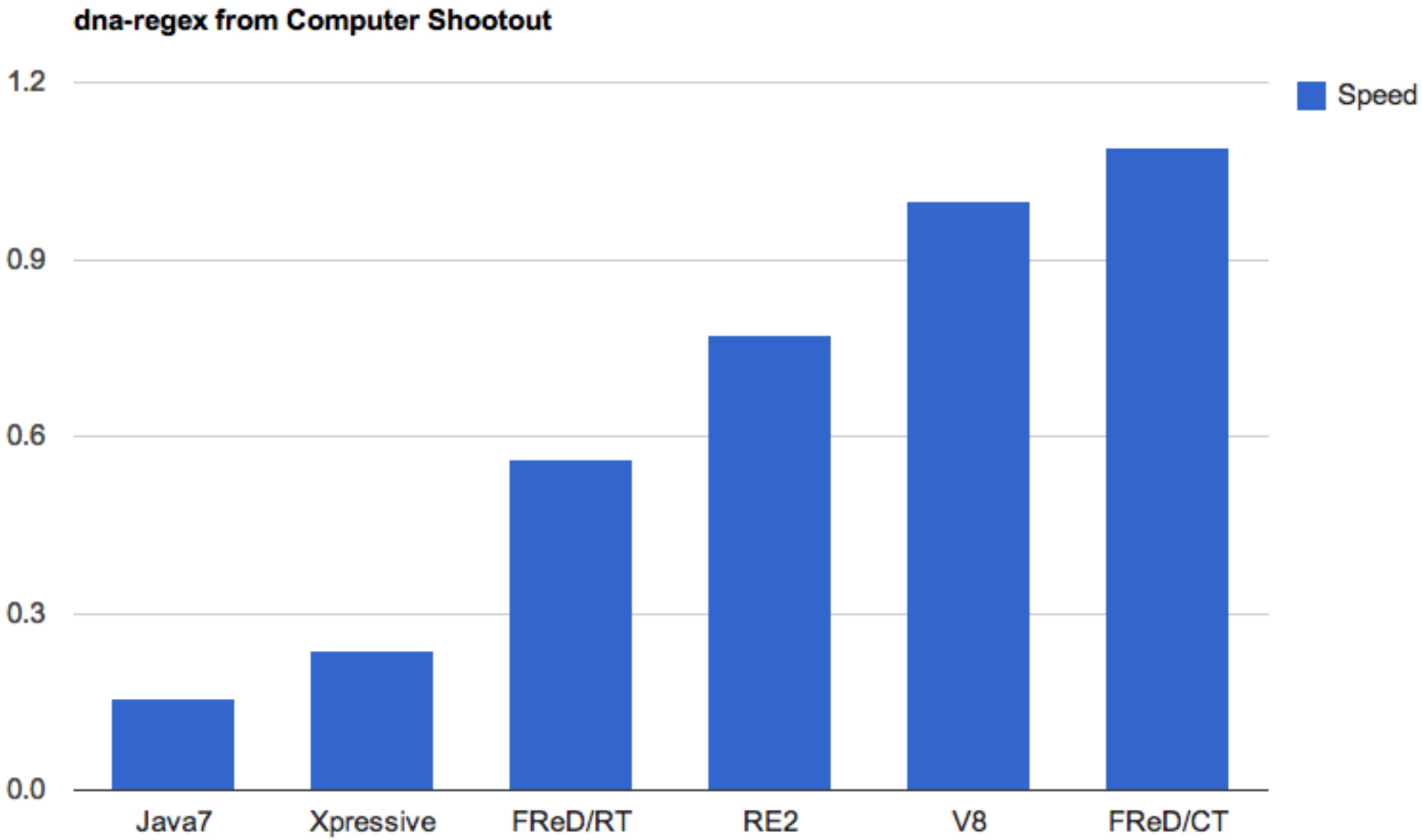
# Two-pronged Approach

- Play into strengths
- Improve on weaknesses

# Strategic strengths

- Active community
- Incredibly fast turnaround
- Compile-Time Function Execution
- Domain-Specific Embedded Languages
- Libraries
- Ranges and algorithms; bulk processing
- Concurrency and parallelism

- One-stop shop for getting work done

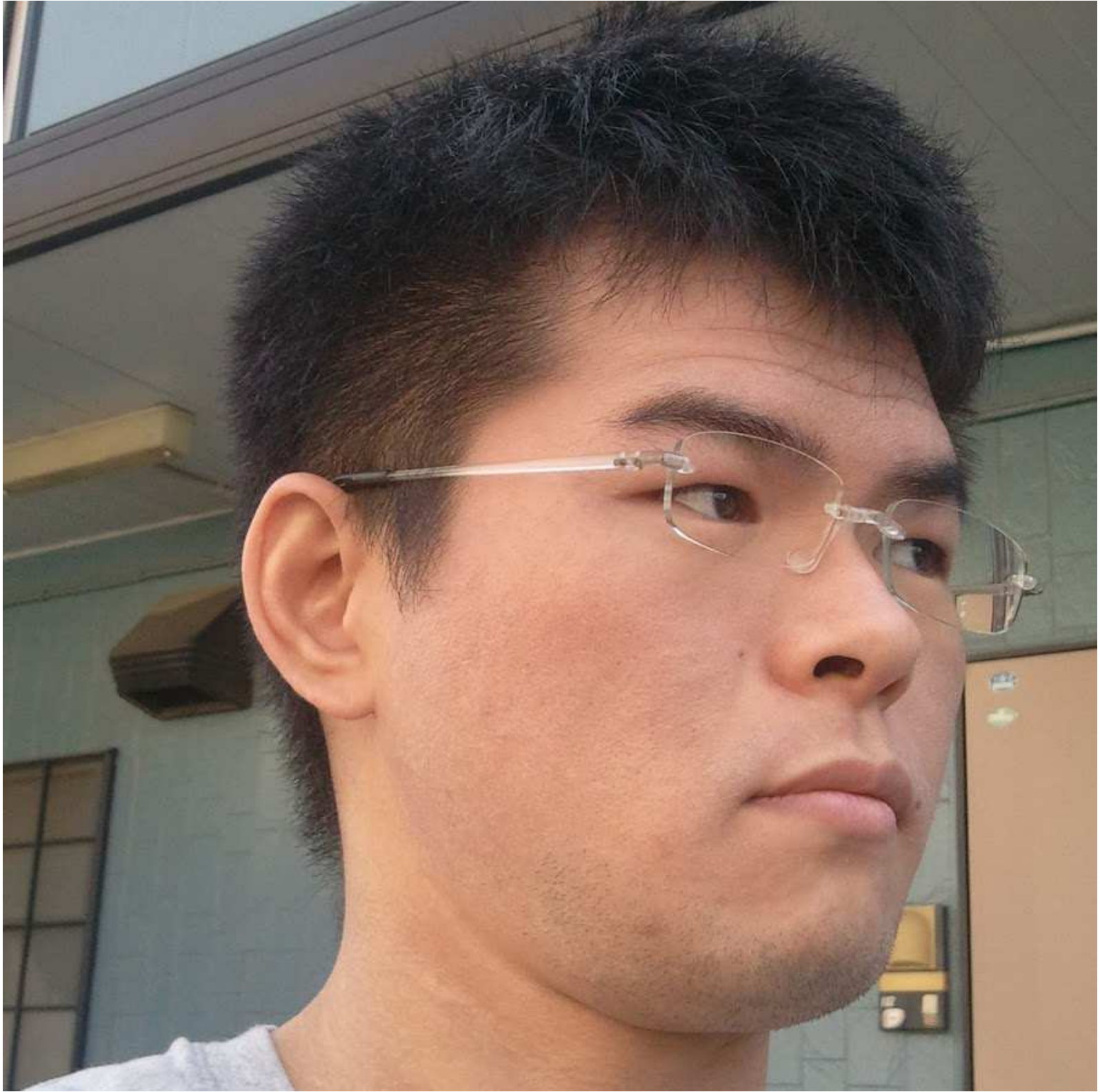

dna-regex from Computer Shootout
1.2
0.9
0.6
0.3
0.0
Speed
Java7
Xpressive
FReD/RT
RE2
V8
FReD/CT

# Weaknesses

- Quality of implementation
- Formal definition
- Available libraries; package management
- Ecosystem tools
- Documentation and tutorials
- Process and roadmap

people


DConf
Parking Only



# Make DConf an annual rallying point

# The People Connection

- Any community needs nurturing
- Focus on increasing participation
- Welcoming new community members
- D Summer of Code (DSOC)?
- `forum.dlang.org` dedicated discussions place

# “D is totally useless”

“When i tried to rewrite example to D, i was shocked. [...] Dlang is a toy in outer space. [...] One can only to write a+b program in schools in it. Now I understand, that’s why D doesn’t have popularity after 10+ years of existence.”

– Temtaime (`tinyurl.com/d-useless`)

# On second thought...

"I investigated a little more in it. Thanks to Jack Applegame, we made a copy of `gl/gl.h` and `opengl32.lib` for DMD. (`http://acomirei.ru/u/gl.d`, `http://acomirei.ru/u/opengl32.lib`). I hope it will be included in DMD, now it's first draft of our work."

– Temtaime

# Community

# Academics

# Corporate

# Summary

# Summary

aspirations

# Summary

aspirations

vision

# Summary

aspirations

vision

people