Fuzion

A new Programming Language for Safety

Who is this guy?

Fridtjof Siebert

Email: siebert@tokiwa.software
github: fridis
twitter: @fridi_s

‘90-‘94	AmigaOberon, AMOK PD
‘97	FEC Eiffel Sparc / Solaris
‘98-‘99	OSF: TurboJ Java Compiler
‘00-‘01	PhD on real-time GC
‘02-‘19	JamaicaVM real-time JVM based on CLASSSPATH / OpenJDK, VeriFlux static analysis tool
‘20-…	Fuzion
‘21-…	Tokiwa Software

Motivation

Many languages overloaded with concepts like classes, methods, interfaces, constructors, traits, records, structs, packages, values, …

➡ Fuzion has one concept: a feature

Today's compilers and tools are more powerful

➡ Tools make better decisions

Systems are safety-critical

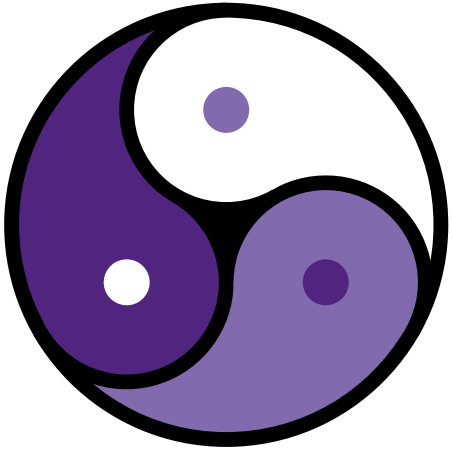
➡ we need to ensure correctness

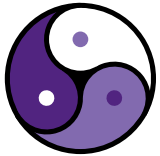
Fuzion Summary

Fuzion

- ➡ uses the **feature** as its main concept
- ➡ is a **statically typed functional** language
- ➡ has **inheritance** and **redefinition**
- ➡ uses **value types** and **dynamic (ref) types**
- ➡ fields **immutable**, uses **effects** for non-functional aspects
- ➡ offloads tasks and decisions from developers to **tools**

Fuzion Logo





Fuzion Resources

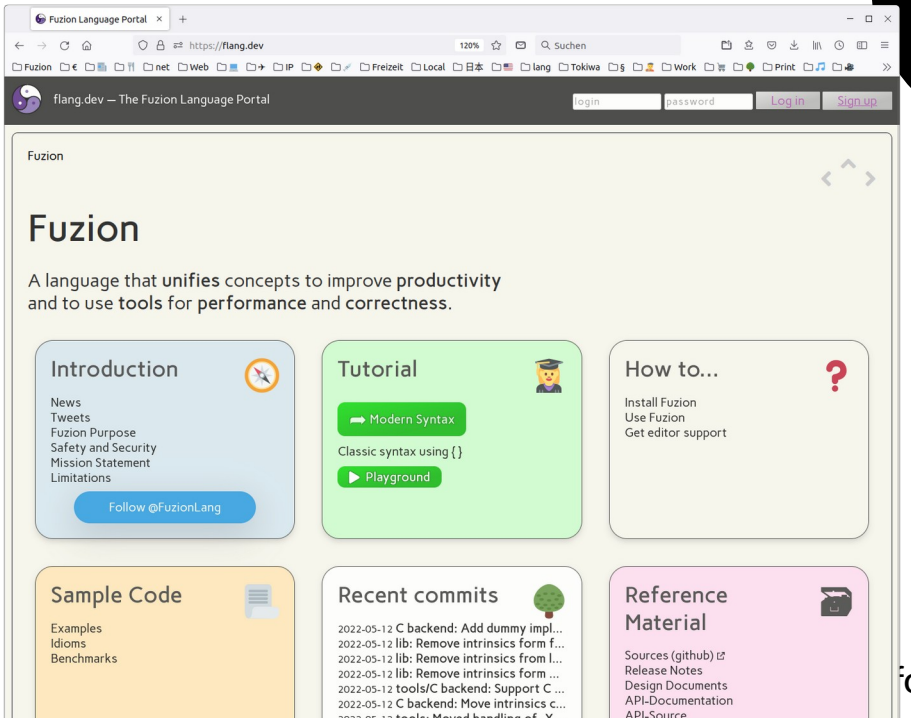
Fuzion available

➡ sources: github.com/tokiwa-software/fuzion

Fuzion Resources

Fuzion available

- ➡ sources: github.com/tokiwa-software/fuzion
- ➡ Website: flang.dev
  - tutorial
  - design
  - examples
  - …



Backing Company

TOKIWA software

- supports development of Fuzion
- currently four employees
- hiring
- searching for funding

Safety vs. Security

Both ‚Sicherheit‘ in German

Safety vs. Security

Both ‚Sicherheit‘ in German

➡ ***Safety*** is the state of being "safe", the condition of being protected from harm or other danger.

—wikipedia

Safety vs. Security

Both ‚Sicherheit‘ in German

➡ ***Safety*** is the state of being "safe", the condition of being protected from harm or other danger.

—wikipedia

➡ ***Security*** is protection from, or resilience against, potential harm caused by others, by restraining the freedom of others to act.

—wikipedia

Safety vs. Security

Both ‚Sicherheit‘ in German

➡ ***Safety*** is the state of being "safe", the condition of being protected from harm or other danger.

—wikipedia

➡ ***Security*** is protection from, or resilience against, potential harm caused by others, by restraining the freedom of others to act.

—wikipedia

clipart from openclipart.org by Anonymous and rygle

Safety vs. Security

Both ‚Sicherheit‘ in German

➡ ***Safety*** is the state of being "safe", the condition of being protected from harm or other danger.

—wikipedia

➡ ***Security*** is protection from, or resilience against, potential harm caused by others, by restraining the freedom of others to act.

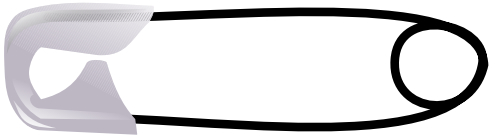
—wikipedia

clipart from openclipart.org by Anonymous and rygle

Safety vs. Security

Code examples

➡ ***Safety***



➡ ***Security***

clipart from openclipart.org by Anonymous and rygle

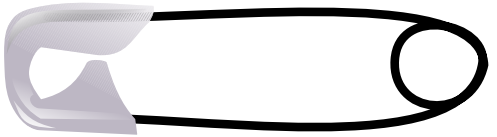
Safety vs. Security

Code examples (C code)

➡ ***Safety***

```
printf(str);
```

— may crash



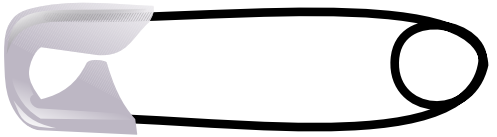
➡ ***Security***

clipart from openclipart.org by Anonymous and rygle

Safety vs. Security

Code examples (C code)

➡ ***Safety***



```
printf(str);
```

— may crash

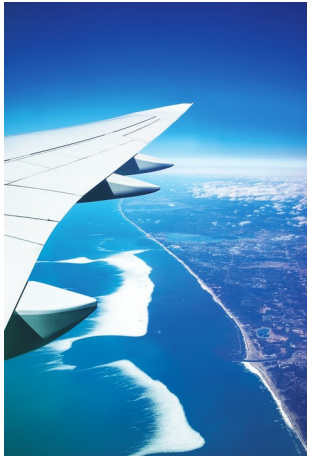
➡ ***Security***

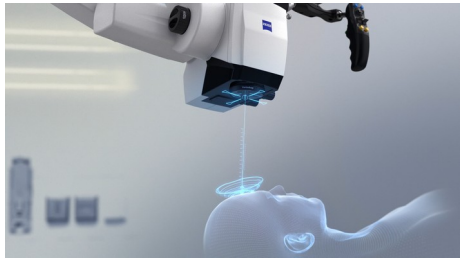
```
str = readString();
printf(str);
```

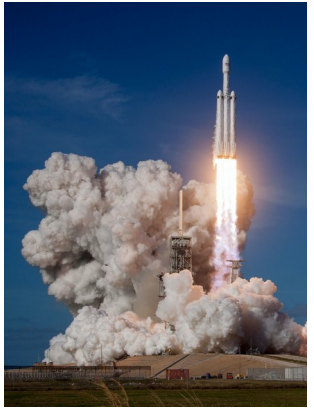
— may give root access

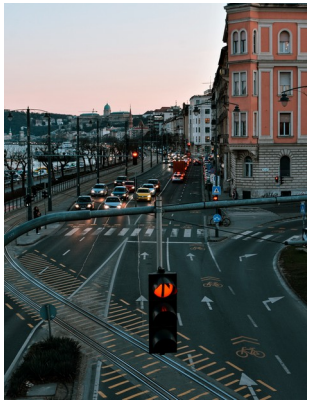
clipart from openclipart.org by Anonymous and rygle

Safety-Critical Systems









Pictures by Simon Maage, Timi Keszthelyi, Winston Chen, Andrey Metelev, Lenny Kuhne, SpaceX, Zeiss, unsplash.com

Safety-Critical Systems

Definition (Wikipedia)

➡ a system whose failure or malfunction may result in [..]:

- death or serious injury to people
- loss or severe damage to equipment/property
- environmental harm

➡ often require certification (IEC61508, DO178C, etc.)

Safety-Critical Systems

Certification typically requires

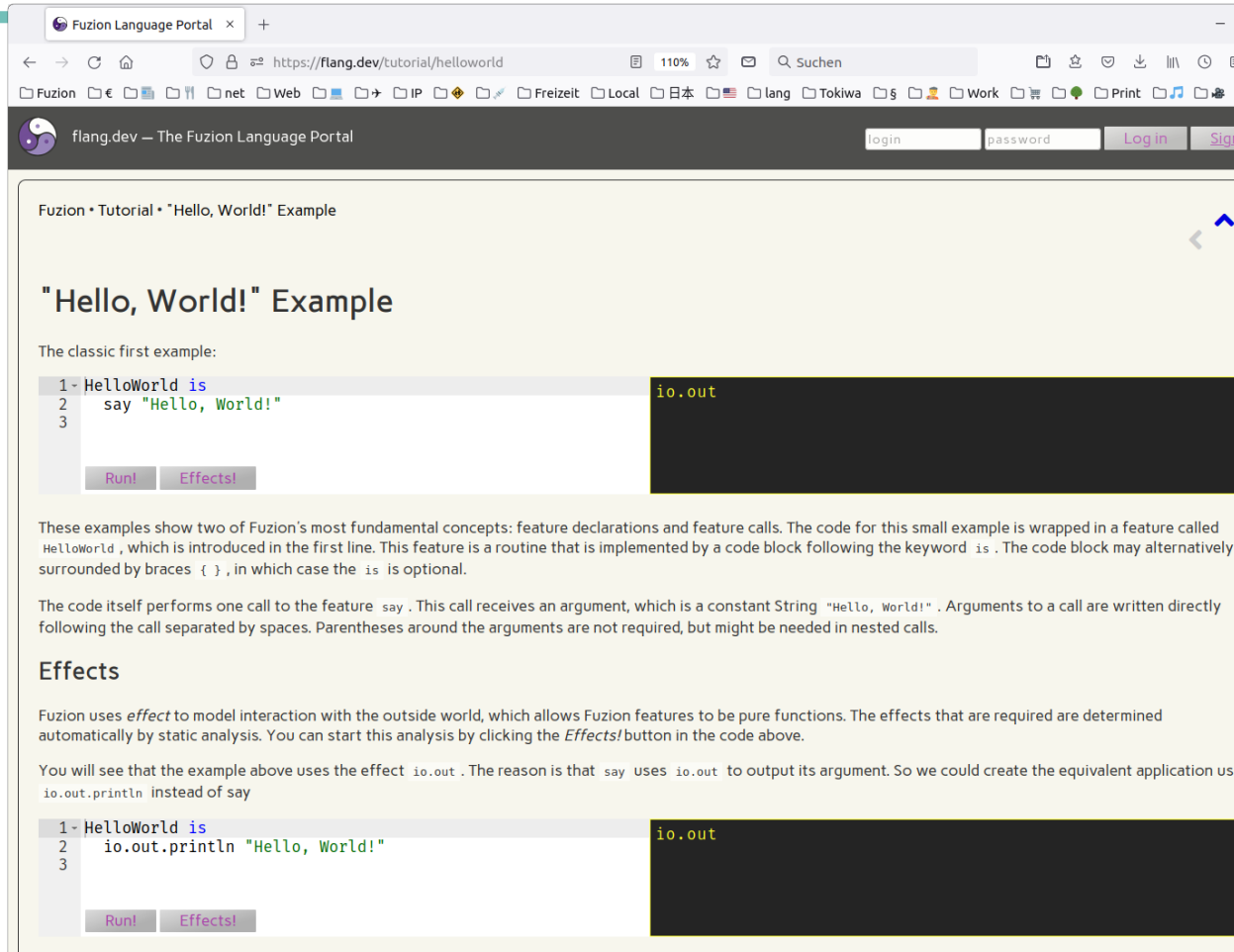
- ➡ defined SW development process
- ➡ traceability
  - requirements ↔ code ↔ validation ↔ results
- ➡ rigorous verification and validation
  - static analysis can help

Fuzion Language

Fuzion Language Tutorial

Not part of this talk

➡ online at **flang.dev**



Fuzion Feature

A Feature is the main abstraction mechanism

➡ generalizes concepts like

- package
- class, interface, trait
- method
- record, struct
- etc.

Feature Examples

Feature as routine with code

```
HelloWorld is
  say "Hello World!"
```

Feature Examples

Nesting of Features

```
HelloWorld is
  hw =>
    say "Hello World!"

  hw
```

Feature Examples

Features with arguments

```
HelloWorld is
  hw(name string) =>
    say "Hello $name!"

  hw "World"
```

Feature Examples

Features with inner features

```
HelloWorld is
  hw(name string) is
    run =>
      say "Hello $name!"

  x := hw "World"
  x.run
```

Feature Examples

Features with inner features

```
HelloWorld is
  hw(name string) is
    run =>
      say "Hello $name!"

  x := hw "World"
  x.run
```

Fuzion code consists of feature declarations and feature calls.

Indentation vs. { }

Fuzion uses indentation and white space

```
HelloWorld is
  hw(name string) is
    run =>
      say "Hello $name!"

  x := hw "World"
  x.run
```

Indentation vs. { }

Fuzion uses indentation and white space

```
HelloWorld is
  hw(name string) is
    run =>
      say "Hello $name!"

  x := hw "World"
  x.run
```

Indentation vs. { }

Fuzion uses indentation and white space

```
HelloWorld is
  hw(name string) is
    run =>
      say "Hello $name!"

  x := hw "World"
  x.run
```

Indentation vs. { }

Fuzion also permits `{ }` and `;` — indentation must match nesting.

```
HelloWorld {
  hw(name string) {
    run => {
      say("Hello $name!");
    }
  }
  x := hw("World");
  x.run();
}
```

What does Fuzion not have?

clipart by Juhele @ openclipart.org

Capabilities considered harmful:

- ➡ Dynamic Loading
- ➡ Macros
- ➡ Reflection
- ➡ Pointer Arithmetic
- ➡ (uncontrolled) Mutability
- ➡ Exceptions

Reasons:

- ➡ We must know what code does
- ➡ Static Analysis
- ➡ Safety
- ➡ Performance

Design by Contract

Design by Contract

Features define their behavior

- ➡ pre-condition: what has to hold before a call?
- ➡ post-condition: what guarantee is given after the call?
- ➡ concept presented by Betrand Meyer back in 1986

Design by Contract: Example

```
sqrt(a i32) i32
  pre
    a >= 0
  post
    result * result <= a,
    (result + 1) * (result + 1) > a
is
  ...
```

Controlling Contract Checks

Checking contracts dynamically

- ➡ will introduce run-time overhead
- ➡ may be prohibitively expensive
- ➡ may be required for safety

Solution

- ➡ qualified contracts

Qualified Contracts

```
sqrt(a i32) i32
  pre
    debug: a >= 0
  post
    debug 5 : result * result <= a,
    debug 5 : (result + 1) * (result + 1) > a
is
  ...
```

Contract Qualifiers

Fuzion contract qualifiers

- ➡ `safety`
- ➡ `debug`
- ➡ `debug n`
- ➡ `pedantic`
- ➡ `analysis`

Contracts for Static Analysis

```
max(a Sequence<i32>) i32
  pre
    debug: !a.isEmpty
  post
    debug: a ∀ x -> x <= result
    debug: a ∃ x -> x = result
    analysis: ∀ i32 x -> x ∈ a : x <= result
    analysis: ∃ i32 x -> x ∈ a && x = result
is
  ...
```

Design-by-Contract & Certification

Contracts provide

- ➡ direct way to add formal requirements to code
- ➡ means to verify these requirements at runtime
- ➡ means to define (or generate) tests
- ➡ formal analysis tools may verify code implements contract

(Side-) Effects and Safety / Security

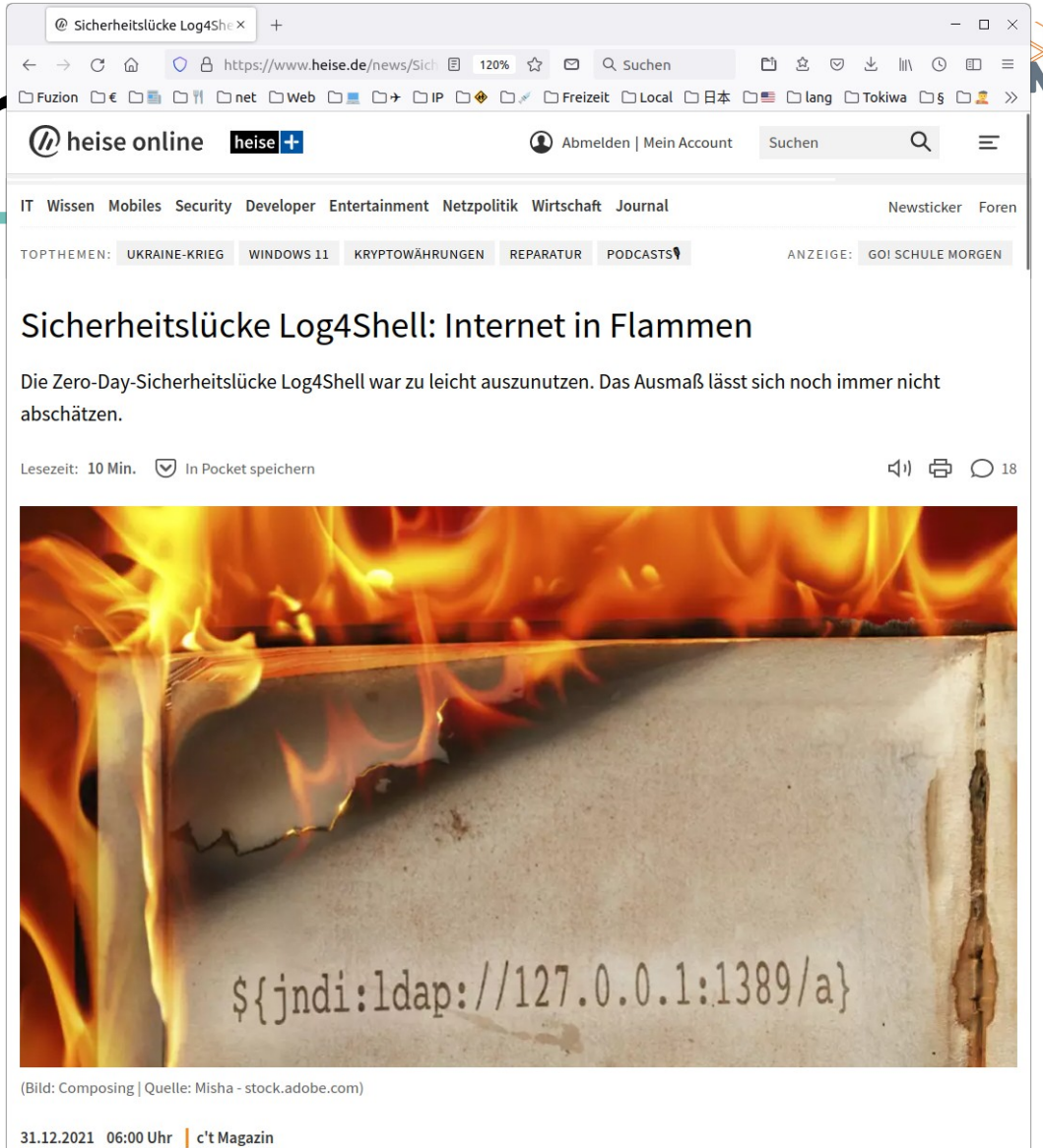
(Side-) Effects and Safety / Security

Recent security alerts

(Side-) Effects ar

Recent security alerts

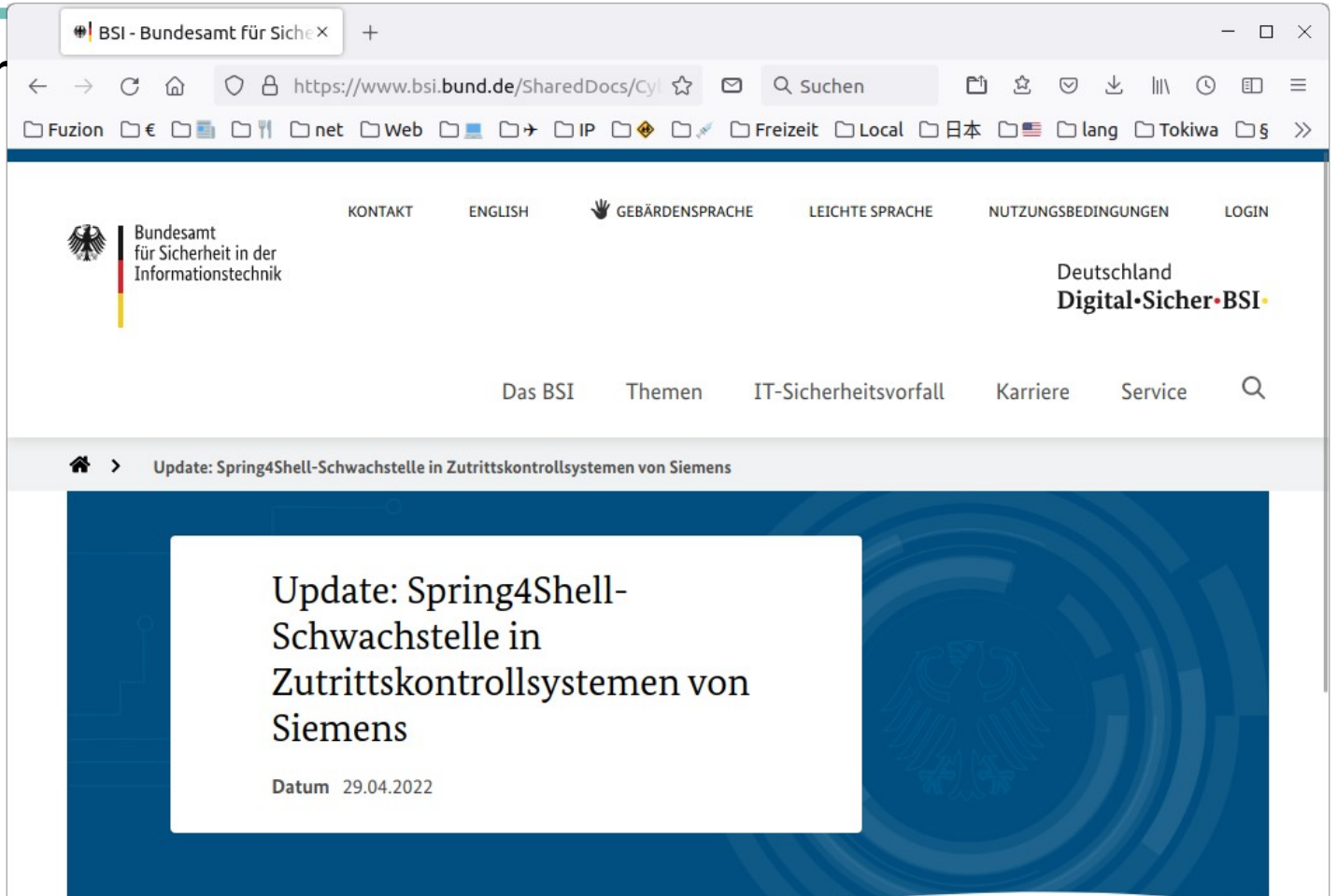
➡ log4shell



(Side-) Effects and Safety / Security

Recent security aler

- ➡ log4shell
- ➡ SpringShell



(Side-) Effects and Safety / Security

Recent security alerts

- ➡ log4shell
- ➡ SpringShell
- ➡ rustdecimal crate

Security advisory: malicio ×

https://blog.rust-lang.org/2022/05/

Suchen

Fuzion € net Web IP Freizeit Local 日本 lang Tokiwa

Rust Blog

Rust Install Learn Tools Governance Community

Security advisory: malicious crate rustdecimal

May 10, 2022 · The Rust Security Response WG

> *This is a cross-post of the official security advisory. The official advisory contains a signed version with our PGP key, as well.*

The Rust Security Response WG and the crates.io team were notified on 2022-05-02 of the existence of the malicious crate `rustdecimal`, which contained malware. The crate

(Side-) Effects and Safety / Security

Recent security alerts

- ➡ log4shell
- ➡ SpringShell
- ➡ rustdecimal crate

Common problem?

(Side-) Effects and Safety / Security

Recent security alerts

- ➡ log4shell
- ➡ SpringShell
- ➡ rustdecimal crate

Common problem

- ➡ Code has unexpected (side-) effects

clipart by j4p4n @ openclipart.org

(Side-) Effects and Safety / Security

Functional community propagates side-effect free code

- ➡ only ‘pure’ functions, no state changes
- ➡ I/O modeled using monads or effect systems
- ➡ automatic thread safety
- ➡ easy parallelization

Fuzion Effects

Fuzion Features are pure functions

- ➡ no mutation of data, no side-effects

Effects are used to model non-functional aspects

- ➡ state changes
- ➡ I/O
- ➡ thread communication
- ➡ exceptions

Fuzion Effects

Static Analysis verifies effects

- ➡ Static analysis determines all effects
- ➡ library code must list all effects
- ➡ unexpected effects are a compile-time error

Fuzion Effects Example

See demo

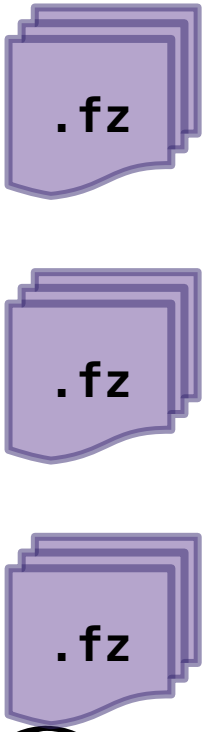
Fuzion Effects Example: random

Getting a new random number has side-effects

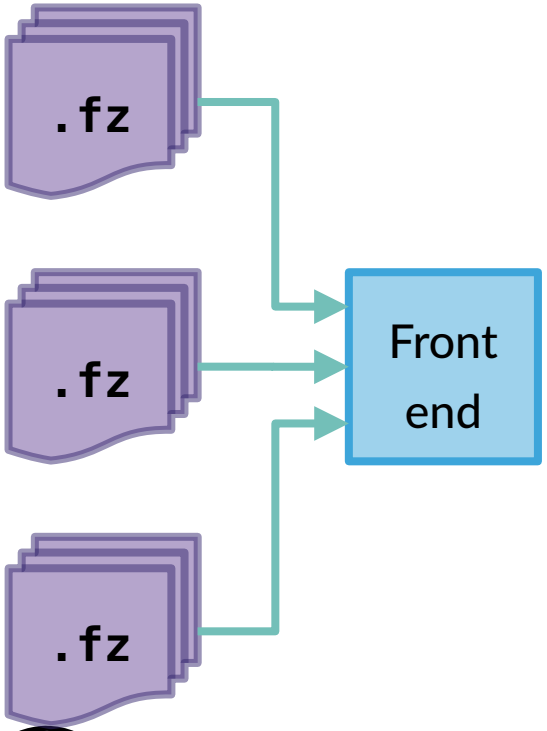
- ➡ original seed must come from somewhere
- ➡ either environment variable
- ➡ or time.nano.

Fuzion Toolchain Design

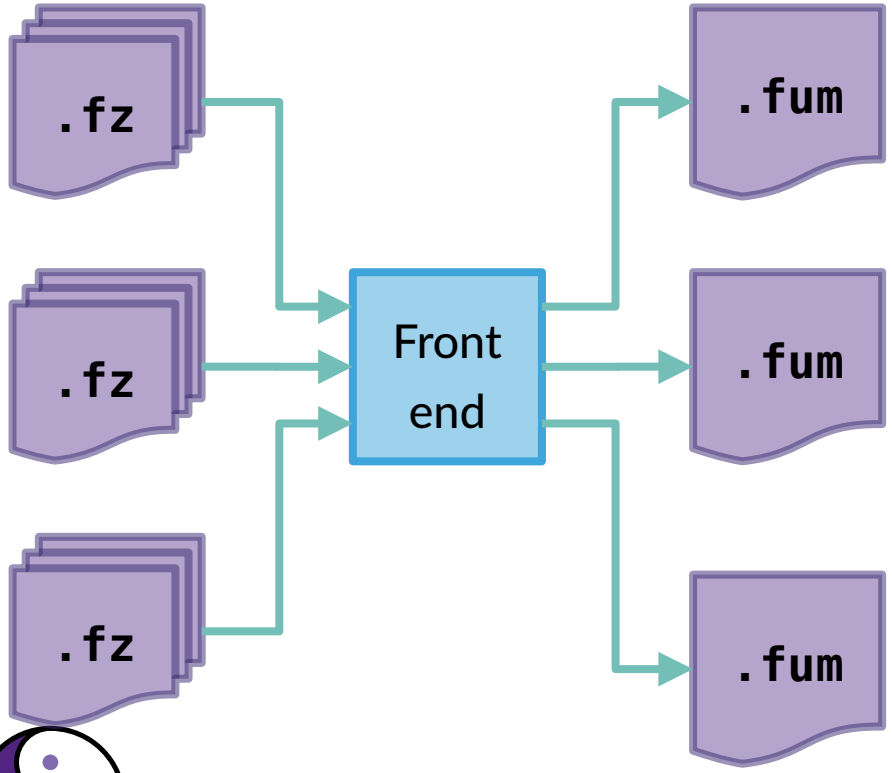
Fuzion Toolchain Design



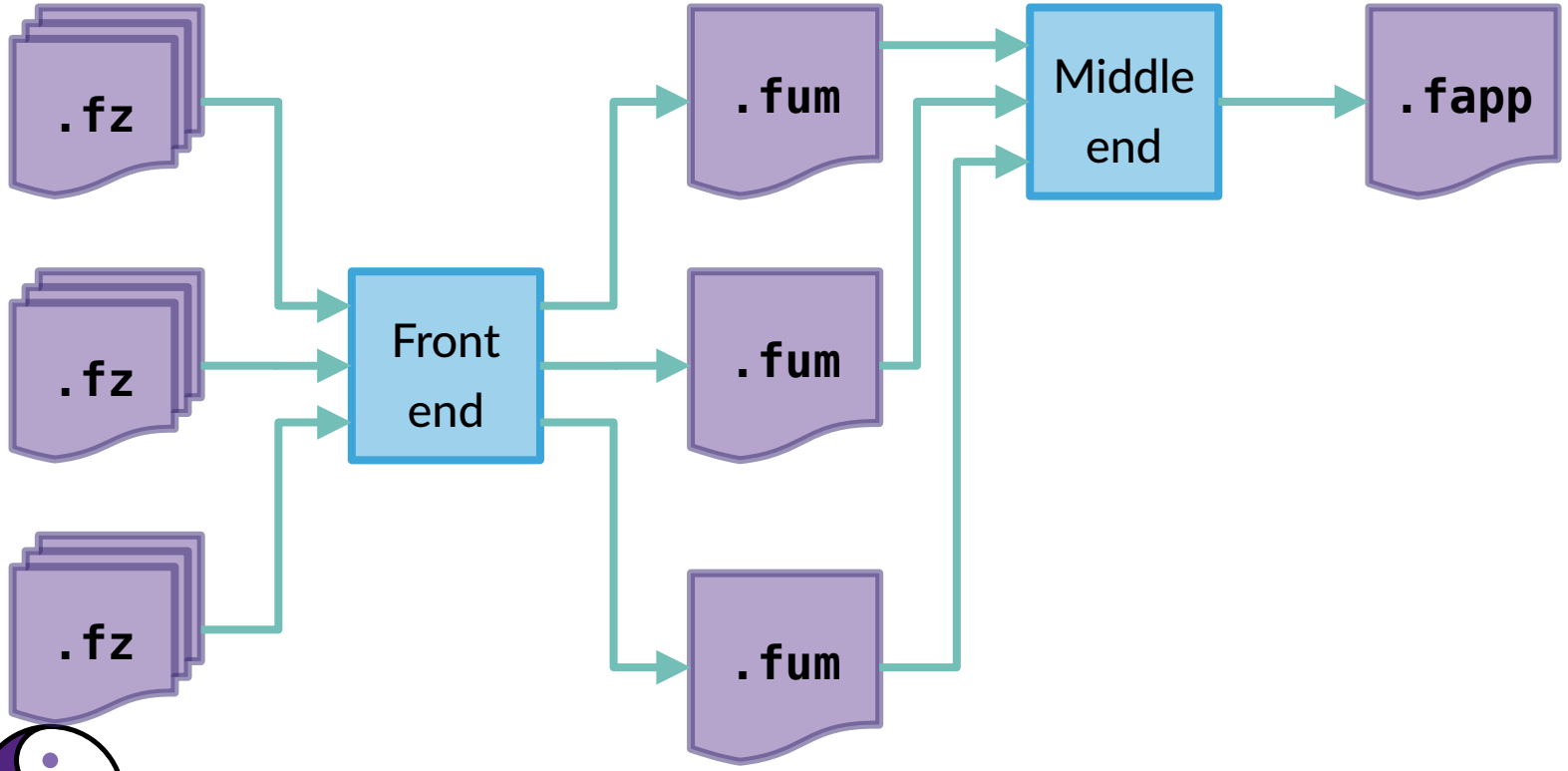
Fuzion Toolchain Design



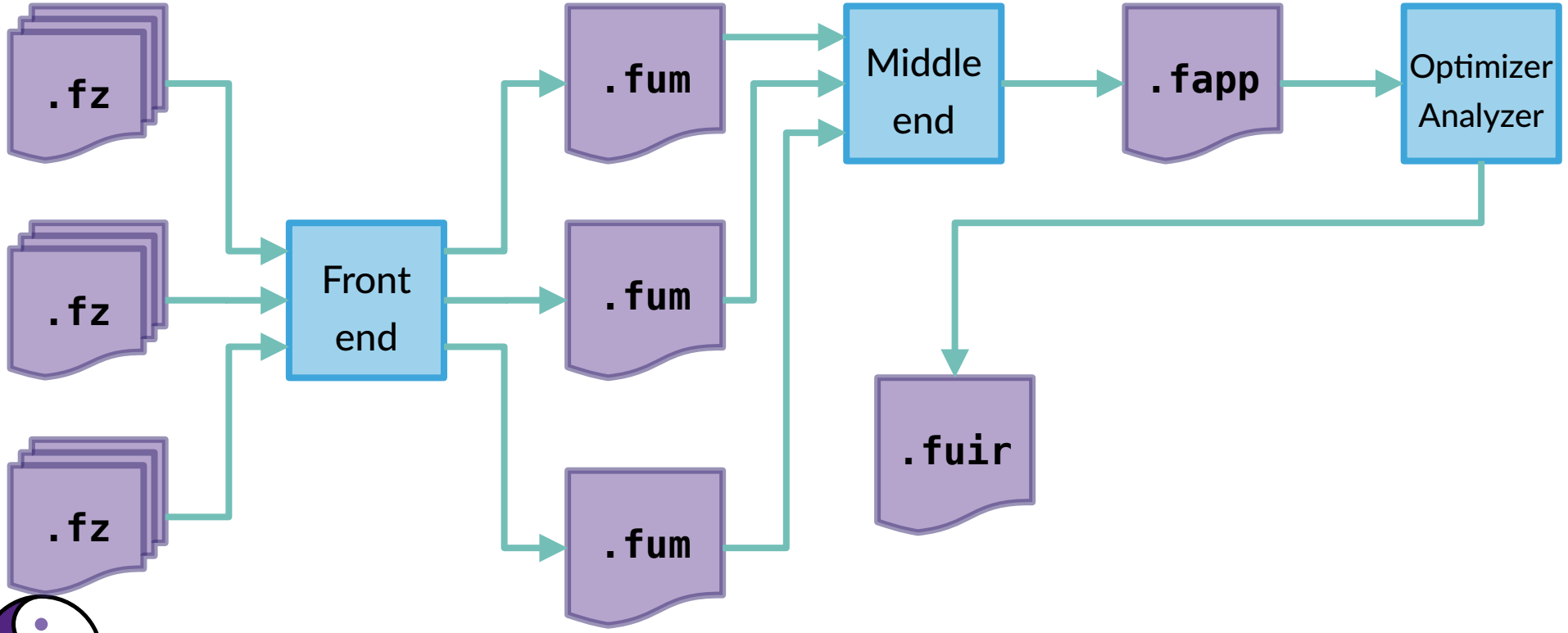
Fuzion Toolchain Design



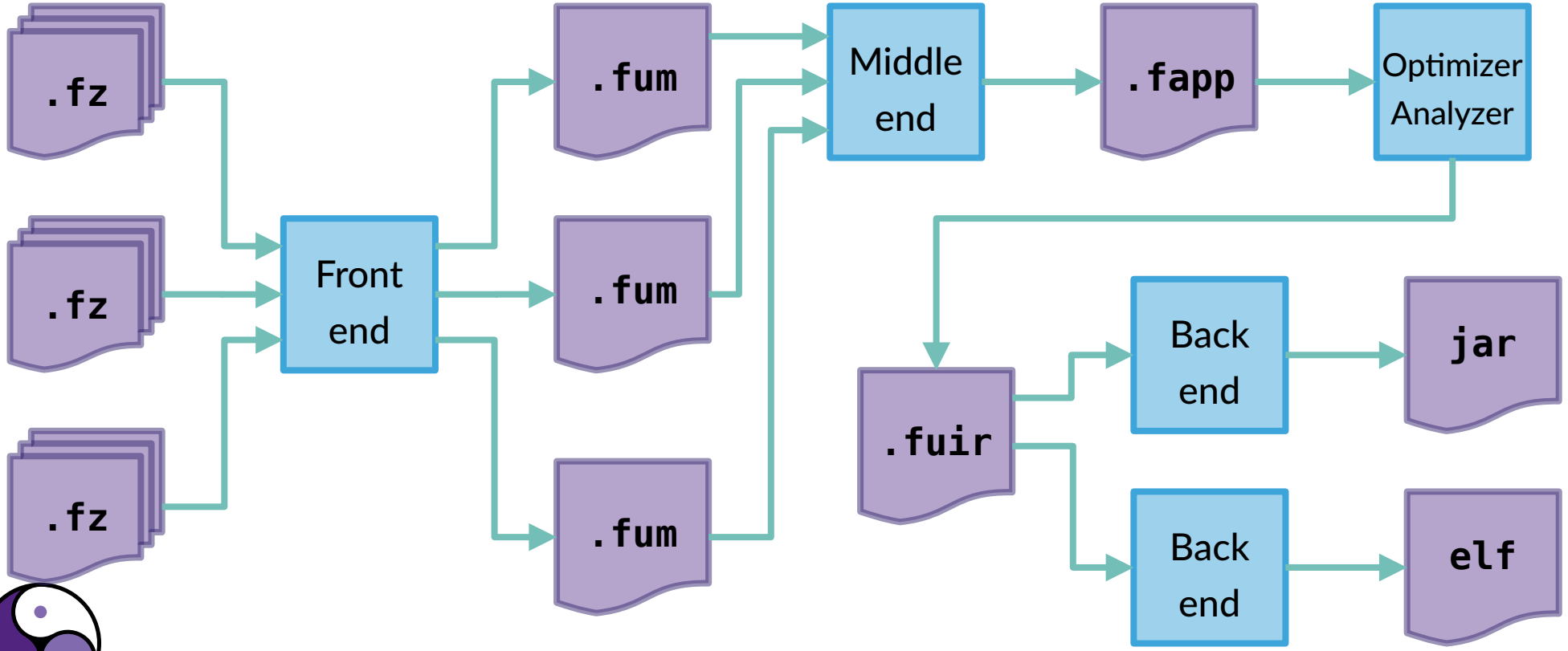
Fuzion Toolchain Design



Fuzion Toolchain Design



Fuzion Toolchain Design



Static Analysis In Fuzion Toolchain

Static analysis currently mostly non-existant.

Will be added to

- ➡ Front End
- ➡ Middle End
- ➡ Optimizer/Analyzer

Analysis Facilitated by Simple IR

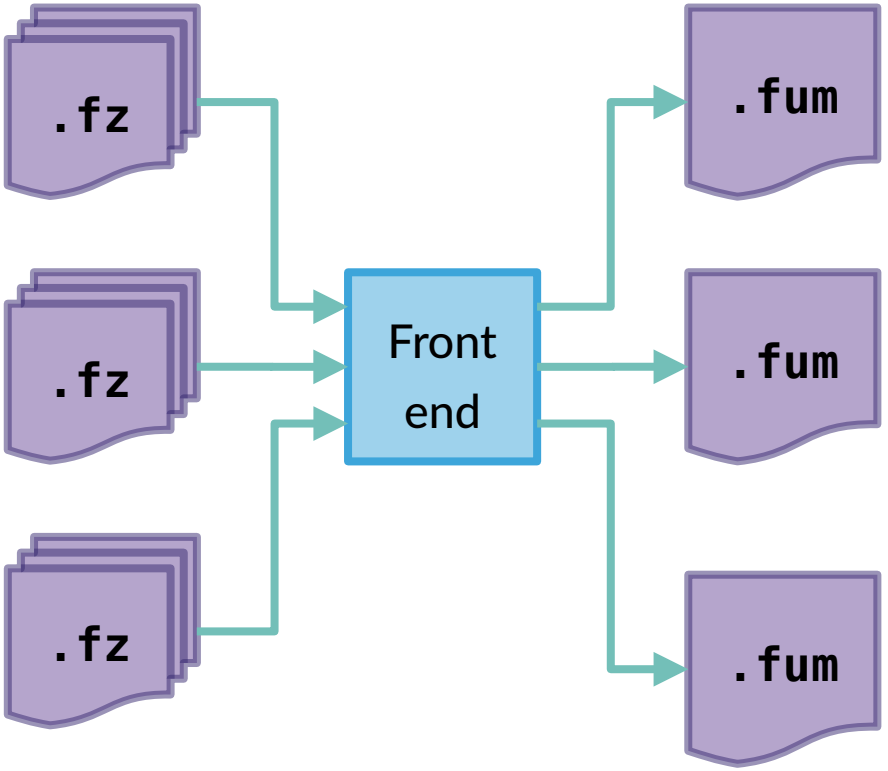
Fuzion Module files contain

- ➡ Features
  - five kinds: **routine**, **field**, **intrinsic**, **abstract** or **choice**
  - contain name, code, types, inner features
- ➡ Types are **feature types** or **type parameters**
- ➡ Code: 10 expressions: **call**, **match**, **const**, **assign**, **pop**, …
  - no loops, no gotos

Static Analysis in Front End

Analyze single module

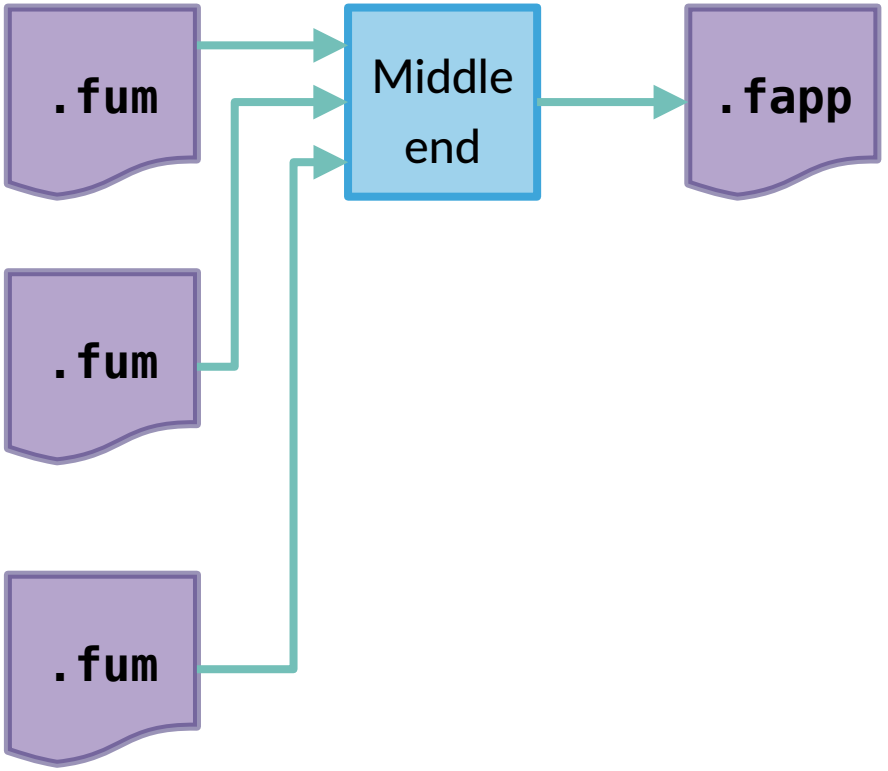
- ➡ Type Checking
- ➡ Init-before-use
- ➡ Thread safety



Static Analysis in Middle End

Analyze whole application

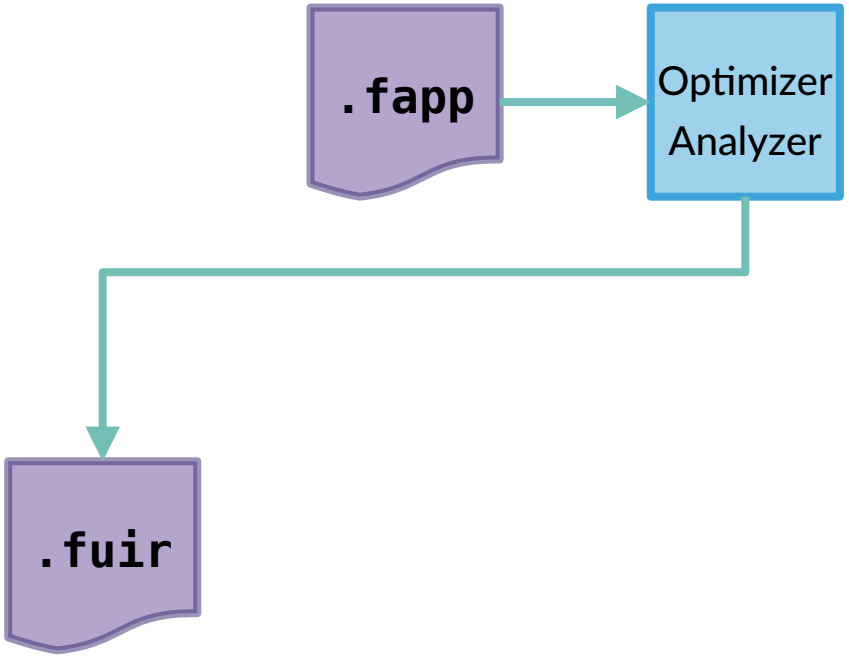
- ➡ Dead code removal
- ➡ Code Specialization
- ➡ Thread local data detection



Static Analysis in Optimizer/Analyzer

Analyze whole application

- ➡ Compile-time evaluation
- ➡ Code Specialization
- ➡ Call-graph analysis
- ➡ Lifespan analysis
  - stack vs. heap allocation
- ➡ Program-wide data flow



Other Tools: FZJava

Create Fuzion interface to Java module


```
java.lang.System.out.println "Hello Java   !"
```

Other Tools: Language Server

Support for IDEs and editors (vim, emacs)

- ➡ completion
- ➡ signature help
- ➡ documentation
- ➡ …

Other Tools: FuzionDoc tool

Extract documentation from Fuzion source code

Fuzion Language Portal

flang.dev — The Fuzion Language Portal

login | password | Log in | Sign up

Fuzion • Documentation • monad

monad

```
monad<A, MA> => monad<monad.A, monad.MA> : Object                [src]
```

```
monad -- generic monad

A monad in X is just a monoid in the category of endofunctors of X, with
product × replaced by composition of endofunctors and unit set by the
identity endofunctor.
-- Saunder Mac Lane, Categories for the Working Mathematician, 1971

Don't be scared, in Java terms: A monad is a means to compose functions
applied to generic types.
```

```
▸ infix >>=(f Function<monad.MA, monad.A>) => MA : Object                [src]
▸ infix >>=~<B, MB>(f Function<monad.infix >>=~.MB, monad.A>) => MB : Object    [src]
▸ join<MMA>(a MMA) => MA : Object                                        [src]
```

Fuzion: Next Steps

Development Plan

- ➡ intermediate files: .fum, .fapp, .fuir
- ➡ simple data-flow-based analysis tools
- ➡ C back-end: GC
  - interfacing C library code
- ➡ Standard Library

Conclusion

Fuzion is an exciting new language for safety

- ➡ focus on **simplicity**
- ➡ uses **design-by-contract** and **effects**
- ➡ prepared for **static analysis**
- ➡ we need
  - to grow our team
  - get developer feedback
  - secure long-term funding
- ➡ please get involved!

http://flang.dev
siebert@tokiwa.software
github.com/tokiwa-software/fuzion