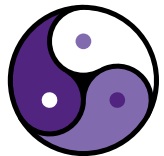


Fuzion

A new Programming Language for Safety



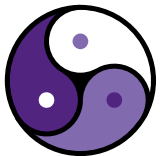
Who is this guy?

Fridtjof Siebert



Email: siebert@tokiwa.software
github: [fridis](https://github.com/fridis)
twitter: [@fridi_s](https://twitter.com/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, ...

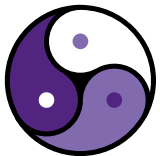
→ Fuzion has one concept: a feature

Today's compilers and tools are more powerful

→ Tools make better decisions

Systems are safety-critical

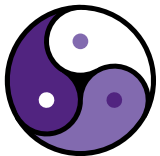
→ we need to ensure correctness



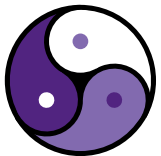
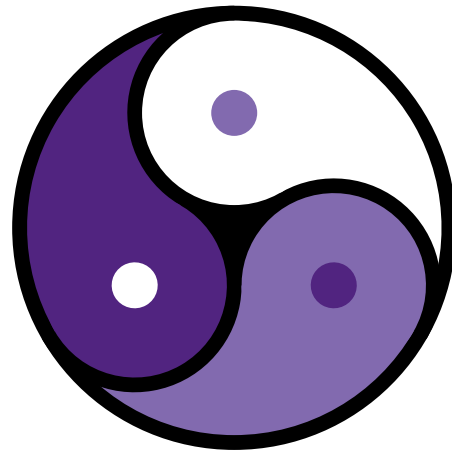
Fuzion Summary

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**



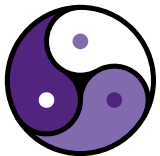
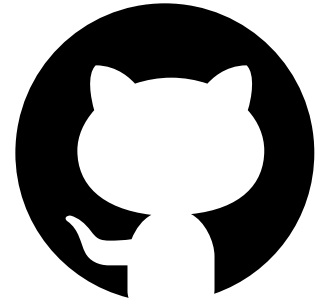
Fuzion Logo



Fuzion Resources

Fuzion available

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



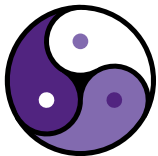
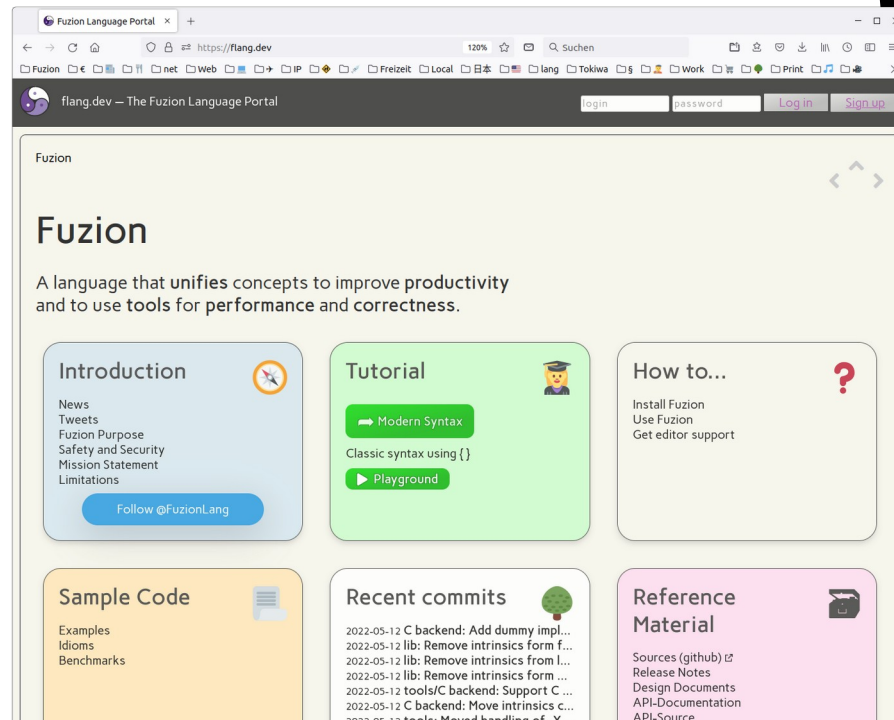
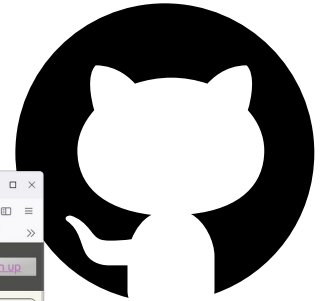
Fuzion Resources

Fuzion available

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

→ Website: flang.dev

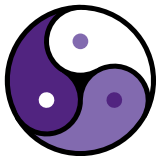
- tutorial
- design
- examples
- ...



Backing Company

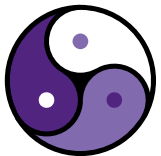


- supports development of Fuzion
- currently four employees
- hiring
- searching for funding



Safety vs. Security

Both ‚Sicherheit‘ in German

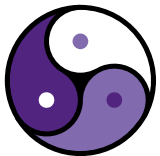


Safety vs. Security

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

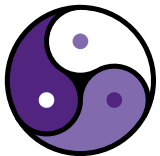
Both ‚Sicherheit‘ in German

→ **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.

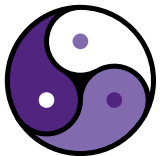


—wikipedia

→ **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.



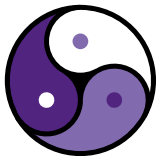
—wikipedia

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



—wiki

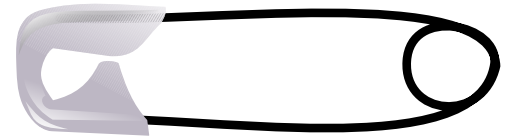
clipart from openclipart.org by Anonymous and rygle



Safety vs. Security

Code examples

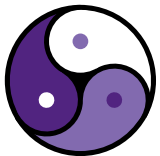
→ *Safety*



→ *Security*



clipart from openclipart.org by Anonymous and rygle



Safety vs. Security

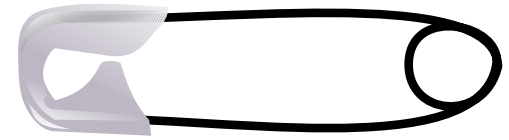
Code examples (C code)

→ *Safety*

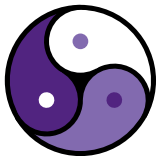
```
printf(str);
```

— may crash

→ *Security*



clipart from openclipart.org by Anonymous and rygle



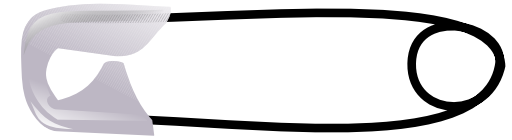
Safety vs. Security

Code examples (C code)

→ *Safety*

```
printf(str);
```

— may crash



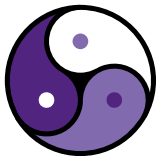
→ *Security*

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

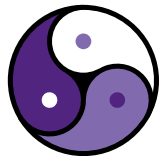
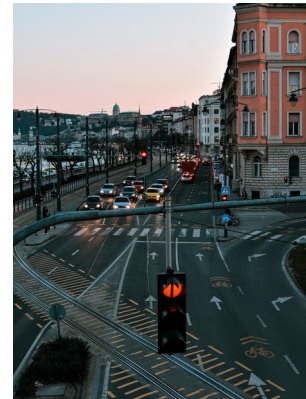
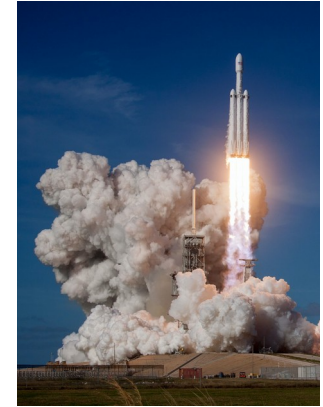
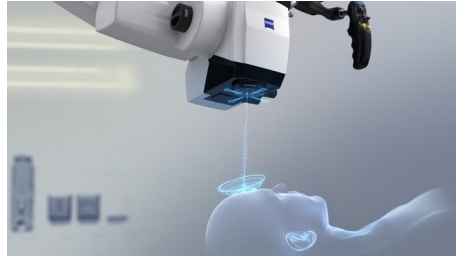
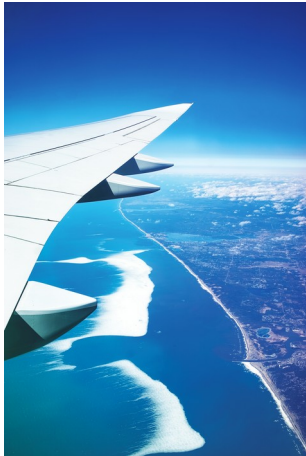
— may give root access



clipart from openclipart.org by Anonymous and rygle



Safety-Critical Systems



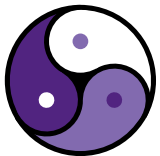
Pictures by Simon Maage, Timi Keszthelyi, Winston Chen, Andrey Metelev, Lenny Kuhne, SpaceX, Zeiss, unsplash.com

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

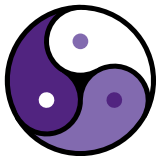
- ➔ often require certification (IEC61508, DO178C, etc.)



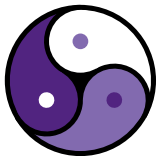
Safety-Critical Systems

Certification typically requires

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



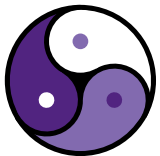
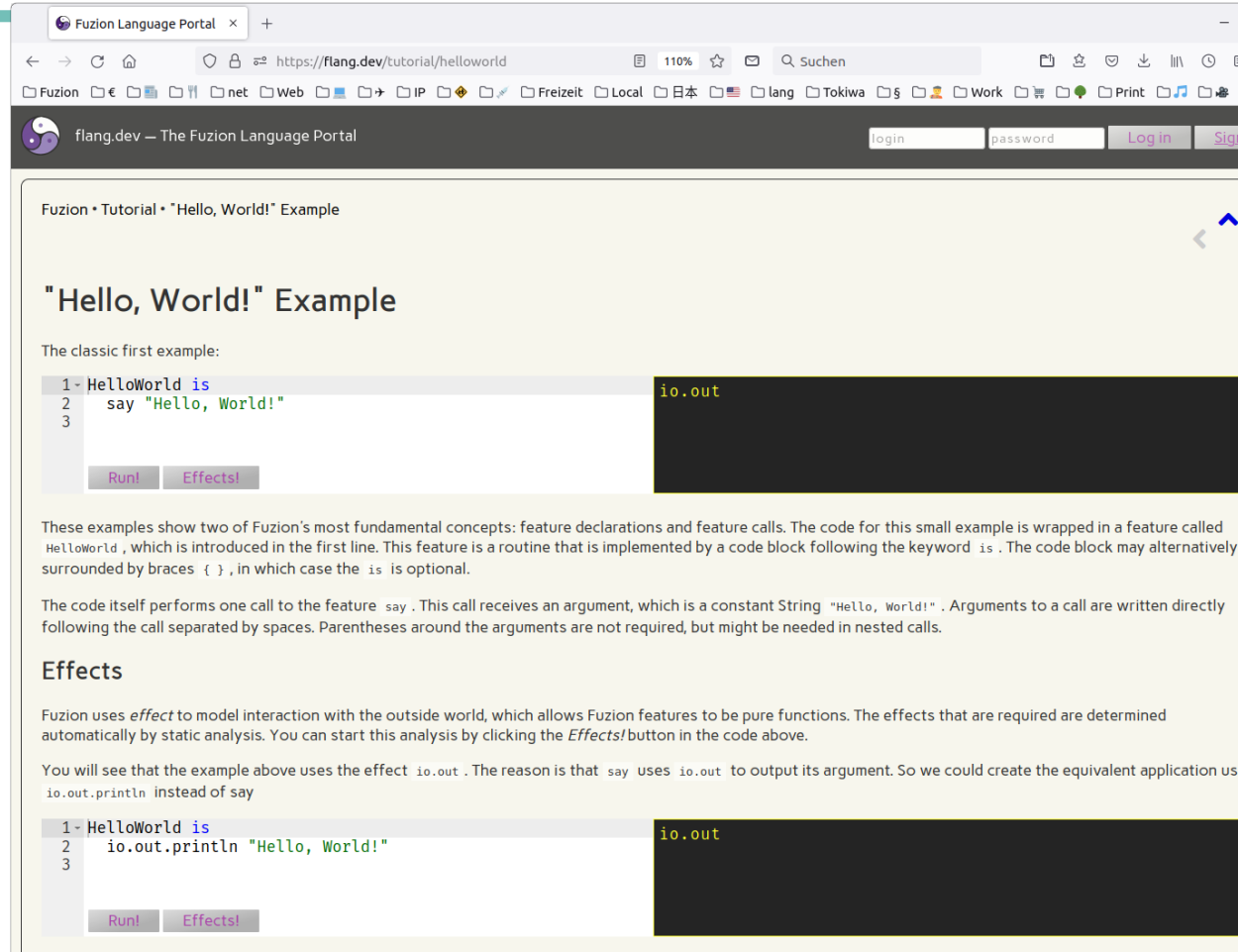
Fuzion Language



Fuzion Language Tutorial

Not part of this talk

→ online at flang.dev

The screenshot shows a web browser at <https://flang.dev/tutorial/helloworld>. The page title is "Fuzion • Tutorial • 'Hello, World!' Example".

"Hello, World!" Example

The classic first example:

```

1- HelloWorld is
2   say "Hello, World!"
3

```

Buttons: Run! Effects!

Output: io.out

These examples show two of Fuzion's most fundamental concepts: feature declarations and feature calls. The code for this small example is wrapped in a feature called `HelloWorld`, which is introduced in the first line. This feature is a routine that is implemented by a code block following the keyword `is`. The code block may alternatively be surrounded by braces `{ }`, in which case the `is` is optional.

The code itself performs one call to the feature `say`. This call receives an argument, which is a constant String `"Hello, World!"`. Arguments to a call are written directly following the call separated by spaces. Parentheses around the arguments are not required, but might be needed in nested calls.

Effects

Fuzion uses *effect* to model interaction with the outside world, which allows Fuzion features to be pure functions. The effects that are required are determined automatically by static analysis. You can start this analysis by clicking the *Effects!* button in the code above.

You will see that the example above uses the effect `io.out`. The reason is that `say` uses `io.out` to output its argument. So we could create the equivalent application using `io.out.println` instead of `say`

```

1- HelloWorld is
2   io.out.println "Hello, World!"
3

```

Buttons: Run! Effects!

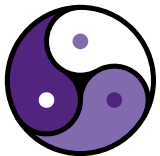
Output: io.out

Fuzion Feature

A Feature is the main abstraction mechanism

→ generalizes concepts like

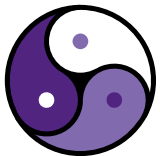
- package
- class, interface, trait
- method
- record, struct
- etc.



Feature Examples

Feature as routine with code

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



Feature Examples

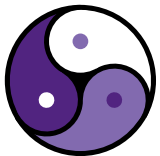
Nesting of Features

HelloWorld is

hw =>

say "Hello World!"

hw

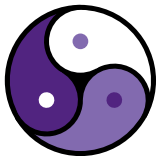


Feature Examples

Features with arguments

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

```
hw "World"
```

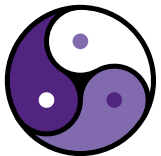


Feature Examples

Features with inner features

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

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



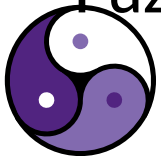
Feature Examples

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.

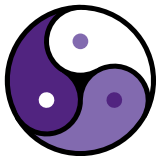


Indentation vs. { }

Fuzion uses indentation and white space

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

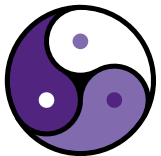
```
x := hw "World"
x.run
```



Indentation vs. { }

Fuzion uses indentation and white space

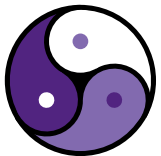
```
HelloWorld is
| hw(name string) is
| | run =>
| | | say "Hello $name!"
|
x := hw "World"
x.run
```



Indentation vs. { }

Fuzion uses indentation and white space

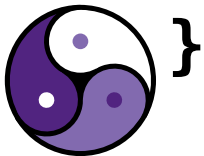
```
HelloWorld is
| hw(name string) is
| | run =>
| | | say "Hello $name!"
|
x := hw "World"
x.run
```



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?

Capabilities considered harmful:

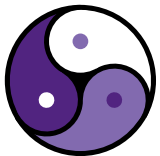
- Dynamic Loading
- Macros
- Reflection
- Pointer Arithmetic
- (uncontrolled) Mutability
- Exceptions

Reasons:

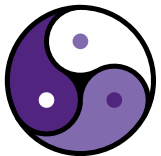
- We must know what code does
- Static Analysis
- Safety
- Performance



clipart by Juhele @ openclipart.org



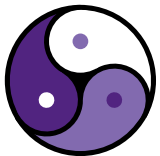
Design by Contract



Design by Contract

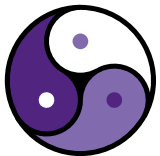
Features define their behavior

- pre-condition: what has to hold before a call?
- post-condition: what guarantee is given after the call?
- concept presented by Bertrand 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
  ...
```



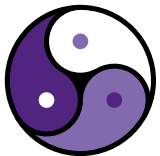
Controlling Contract Checks

Checking contracts dynamically

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

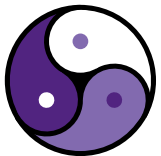
Solution

- qualified contracts



Qualified Contracts

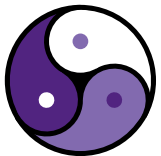
```
sqrt(a i32) i32
  pre
    debug: a >= 0
  post
    debug 5 : result * result <= a,
    debug 5 : (result + 1) * (result + 1) > a
is
  ...
```



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 \forall x \rightarrow x \leq result

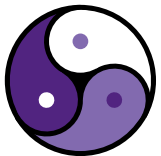
debug: a \exists x \rightarrow x = result

analysis: \forall i32 x \rightarrow x \in a : x \leq result

analysis: \exists i32 x \rightarrow x \in a && x = result

is

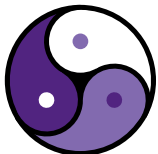
...



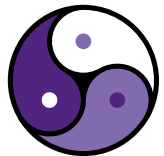
Design-by-Contract & Certification

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



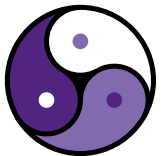
(Side-) Effects and Safety / Security



(Side-) Effects and Safety / Security



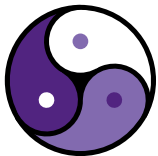
Recent security alerts



(Side-) Effects are

Recent security alerts

→ log4shell



The screenshot shows a web browser window with the URL <https://www.heise.de/news/Sich>. The page is from Heise online and features a navigation bar with categories like IT, Wissen, Mobiles, Security, Developer, Entertainment, Netzpolitik, Wirtschaft, and Journal. Below the navigation bar, there are top themes such as UKRAINE-KRIEG, WINDOWS 11, KRYPTOWÄHRUNGEN, REPARATUR, and PODCASTS. The main article title is "Sicherheitslücke Log4Shell: Internet in Flammen". The text below the title reads: "Die Zero-Day-Sicherheitslücke Log4Shell war zu leicht auszunutzen. Das Ausmaß lässt sich noch immer nicht abschätzen." Below the text, there is a reading time of "10 Min." and a "In Pocket speichern" button. A large image shows a piece of paper with the LDAP payload `${jndi:ldap://127.0.0.1:1389/a}` written on it, with flames rising from the edges of the paper, symbolizing the "internet in flames" mentioned in the title. At the bottom of the screenshot, there is a caption "(Bild: Composing | Quelle: Misha - stock.adobe.com)" and a footer with the date "31.12.2021 06:00 Uhr" and the source "c't Magazin".

(Side-) Effects and Safety / Security

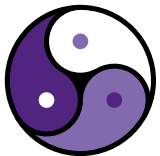


Recent security alerts

→ log4shell

→ SpringShell

A screenshot of a web browser displaying a page from the German Federal Office for Information Security (BSI). The browser's address bar shows the URL 'https://www.bsi.bund.de/SharedDocs/Cyber'. The page header includes the BSI logo and navigation links for 'KONTAKT', 'ENGLISH', 'GEBÄRDENSPRACHE', 'LEICHTE SPRACHE', 'NUTZUNGSBEDINGUNGEN', and 'LOGIN'. The main content area features a blue background with a white box containing the text: 'Update: Spring4Shell-Schwachstelle in Zutrittskontrollsystemen von Siemens' and 'Datum 29.04.2022'. The page also includes a search bar and a breadcrumb trail: 'Update: Spring4Shell-Schwachstelle in Zutrittskontrollsystemen von Siemens'.

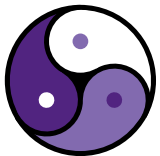


(Side-) Effects and Safety / Security



Recent security alerts

- log4shell
- SpringShell
- rustdecimal crate



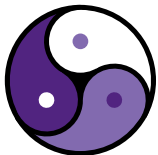
The screenshot shows a web browser window with the URL `https://blog.rust-lang.org/2022/05/`. The page title is "Security advisory: malicious crate rustdecimal". The browser's address bar shows the URL and a search bar with the text "Suchen". The page header includes the Rust logo and the text "Rust Blog" followed by navigation links: "Rust", "Install", "Learn", "Tools", "Governance", and "Community". The main heading is "Security advisory: malicious crate rustdecimal". Below the heading, the date "May 10, 2022" and the author "The Rust Security Response WG" are displayed. A light blue box contains the text: "This is a cross-post of [the official security advisory](#). The official advisory contains a signed version with our PGP key, as well." At the bottom, the text reads: "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?



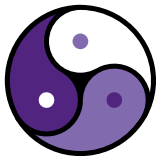
(Side-) Effects and Safety / Security

Recent security alerts

- log4shell
- SpringShell
- rustdecimal crate

Common problem

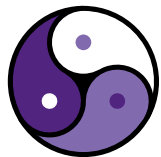
- Code has unexpected (side-) effects



(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

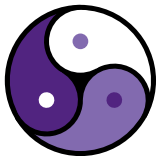
Effects are used to model non-functional aspects

→ state changes

→ I/O

→ thread communication

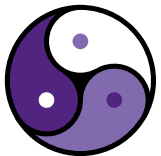
→ exceptions



Fuzion Effects

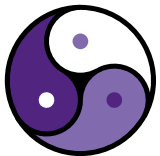
Static Analysis verifies effects

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



Fuzion Effects Example

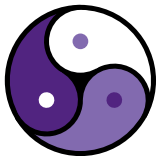
See demo



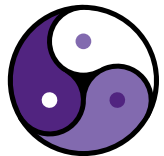
Fuzion Effects Example: random

Getting a new random number has side-effects

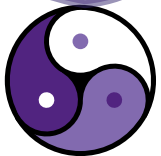
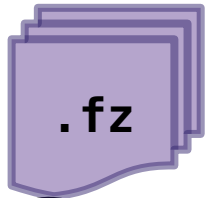
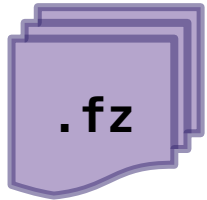
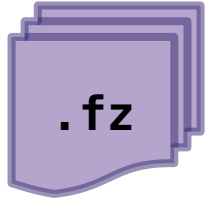
- original seed must come from somewhere
- either environment variable
- or `time.nano`.



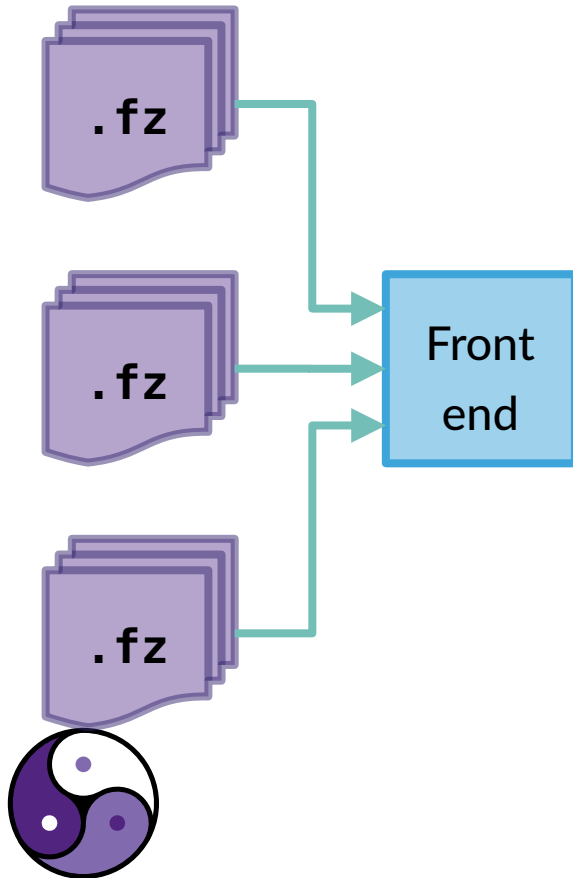
Fuzion Toolchain Design



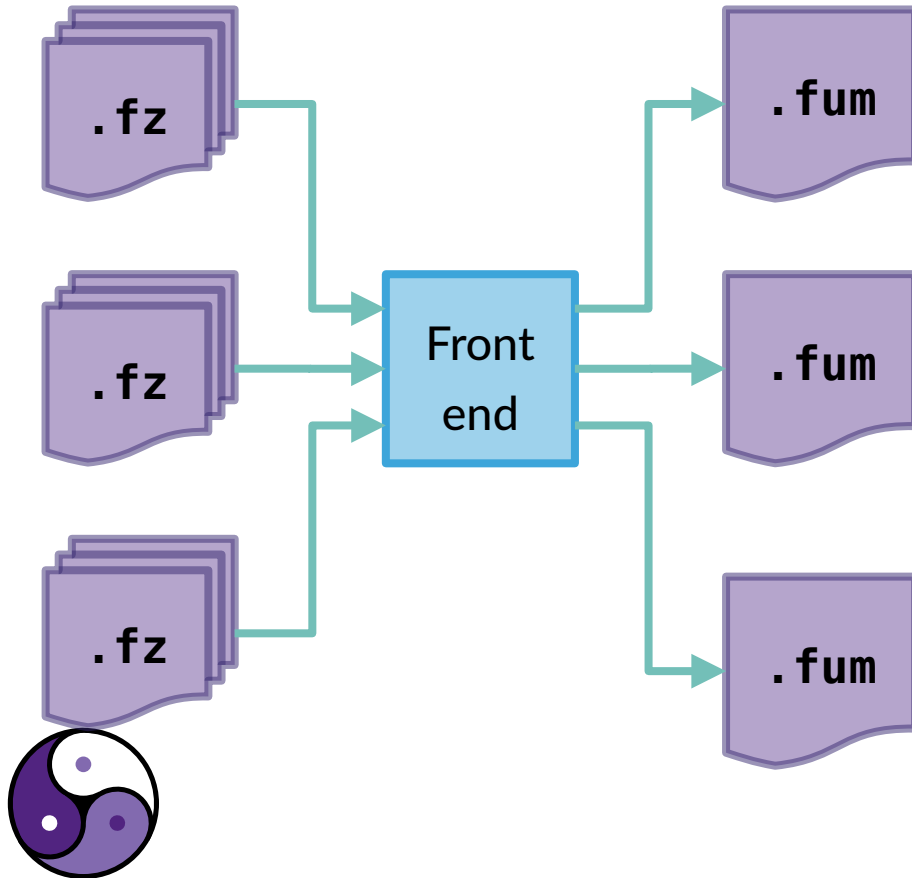
Fuzion Toolchain Design



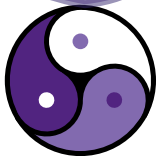
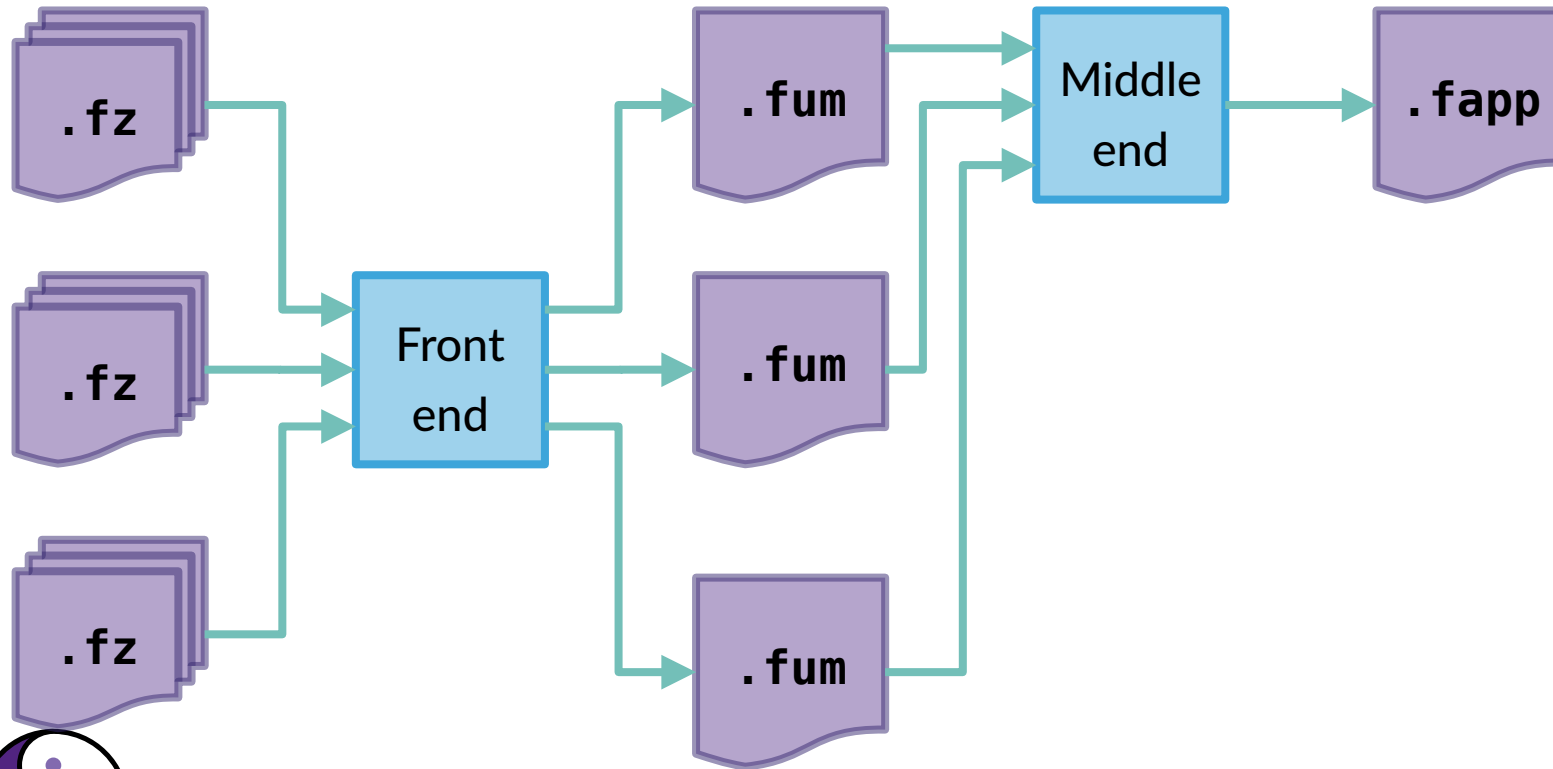
Fuzion Toolchain Design



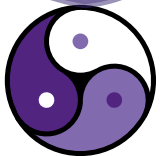
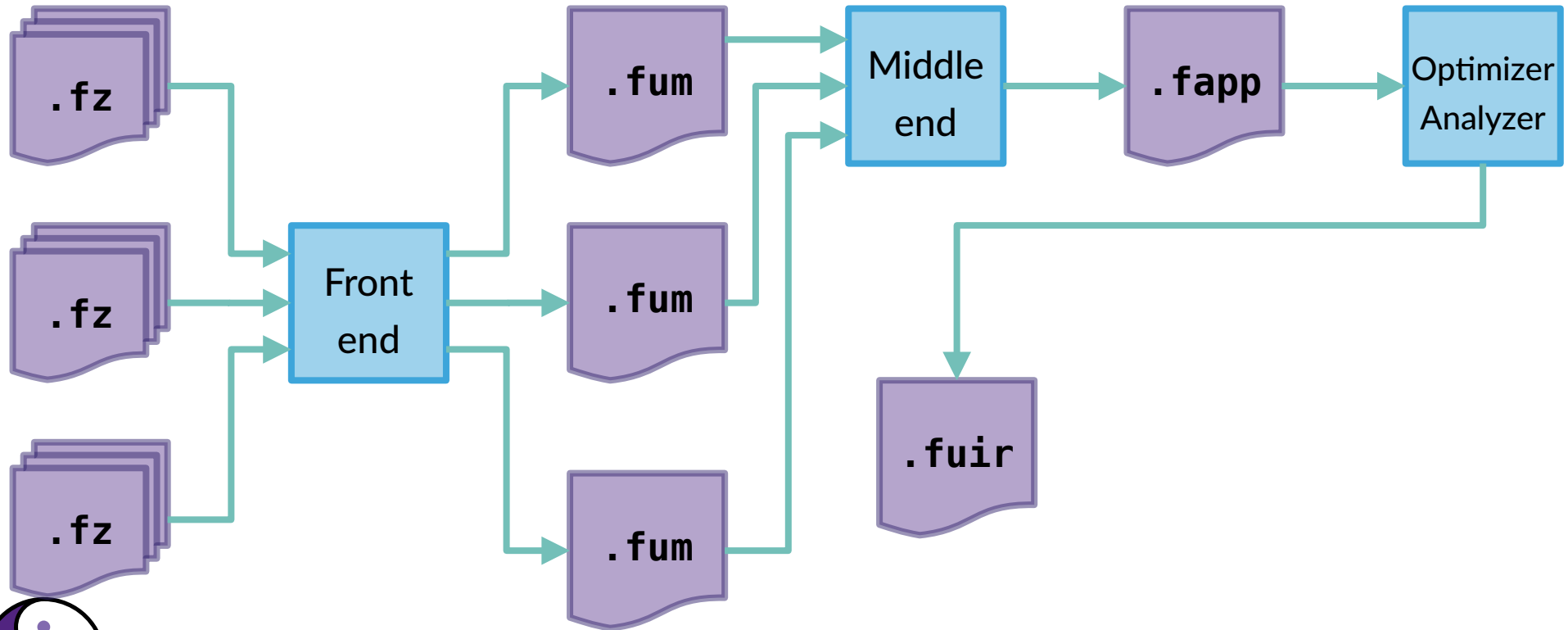
Fuzion Toolchain Design



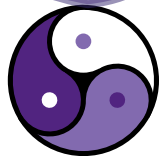
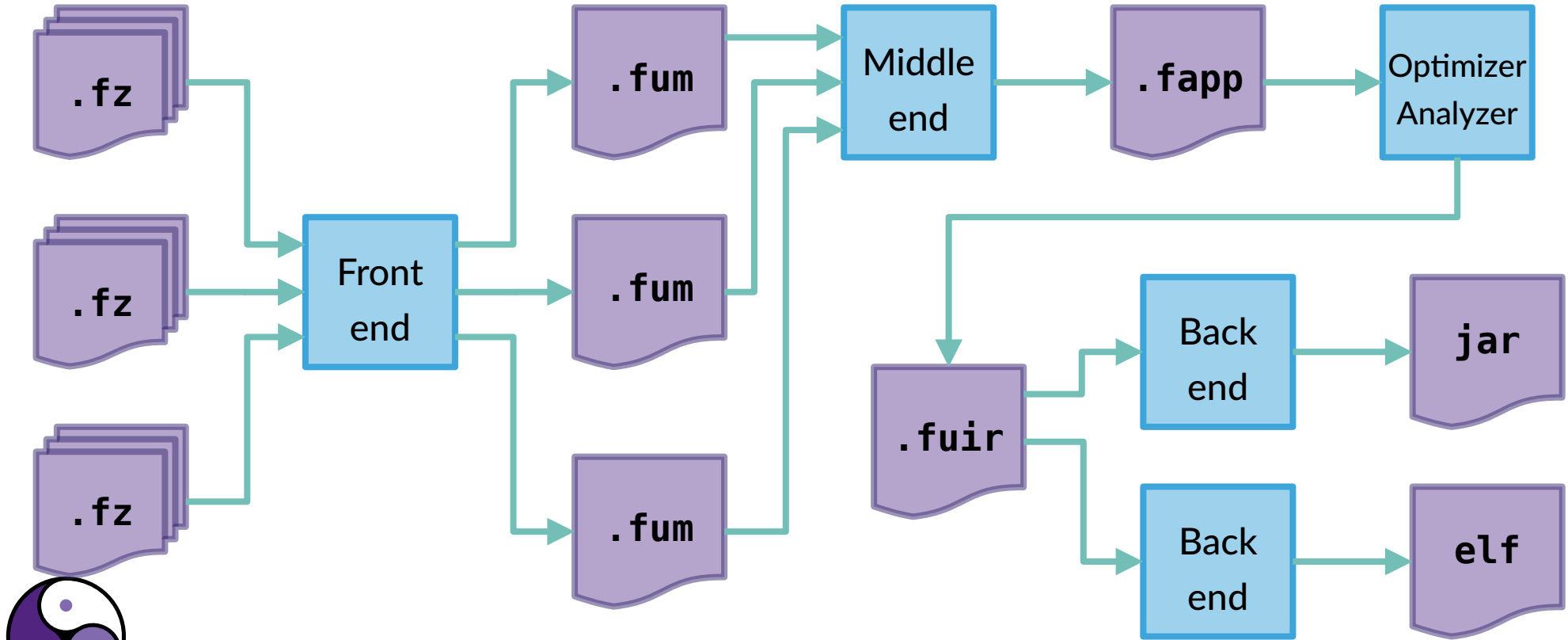
Fuzion Toolchain Design



Fuzion Toolchain Design



Fuzion Toolchain Design

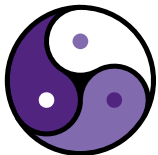


Static Analysis In Fuzion Toolchain

Static analysis currently mostly non-existent.

Will be added to

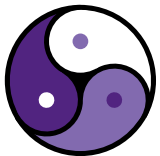
- Front End
- Middle End
- Optimizer/Analyzer



Analysis Facilitated by Simple IR

Fuzion Module files contain

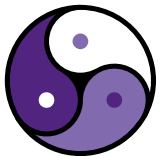
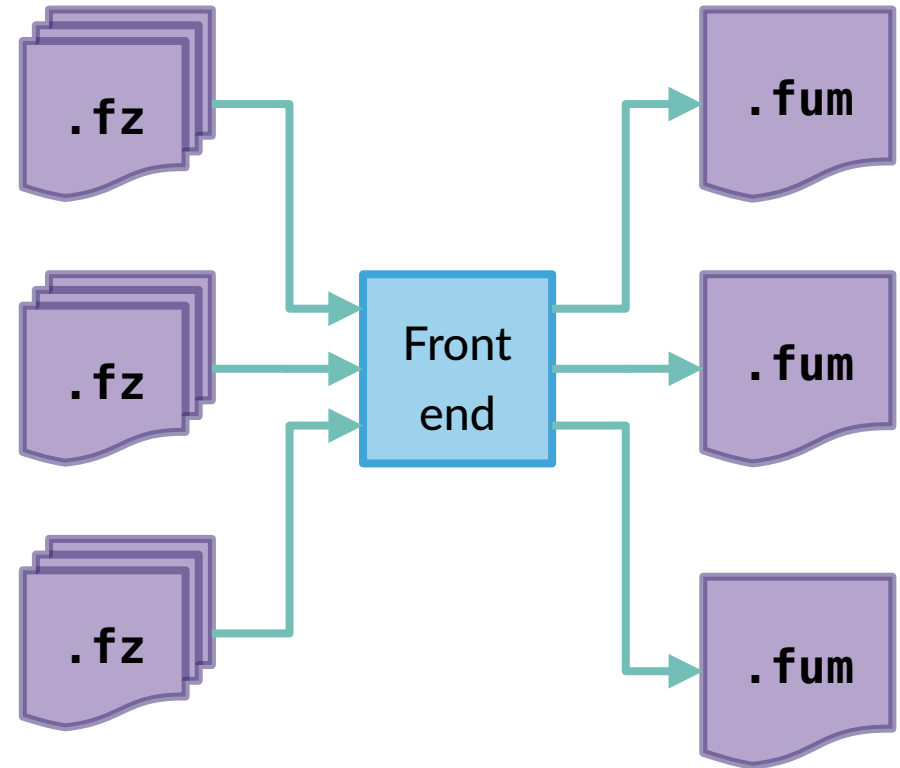
- Features
 - five kinds: **routine**, **field**, **intrinsic**, **abstract** or **choice**
 - contain name, code, types, inner features
- Types are **feature types** or **type parameters**
- Code: 10 expressions: **call**, **match**, **const**, **assign**, **pop**, ...
 - no loops, no gotos



Static Analysis in Front End

Analyze single module

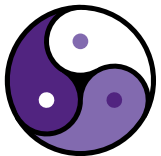
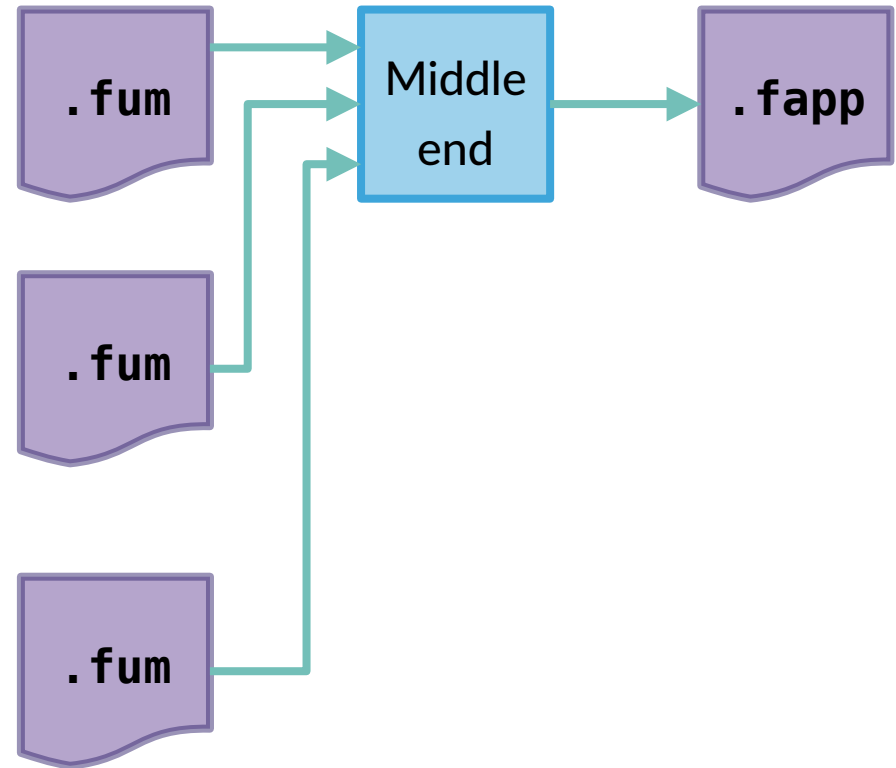
- Type Checking
- Init-before-use
- Thread safety



Static Analysis in Middle End

Analyze whole application

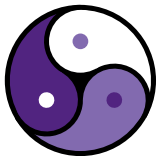
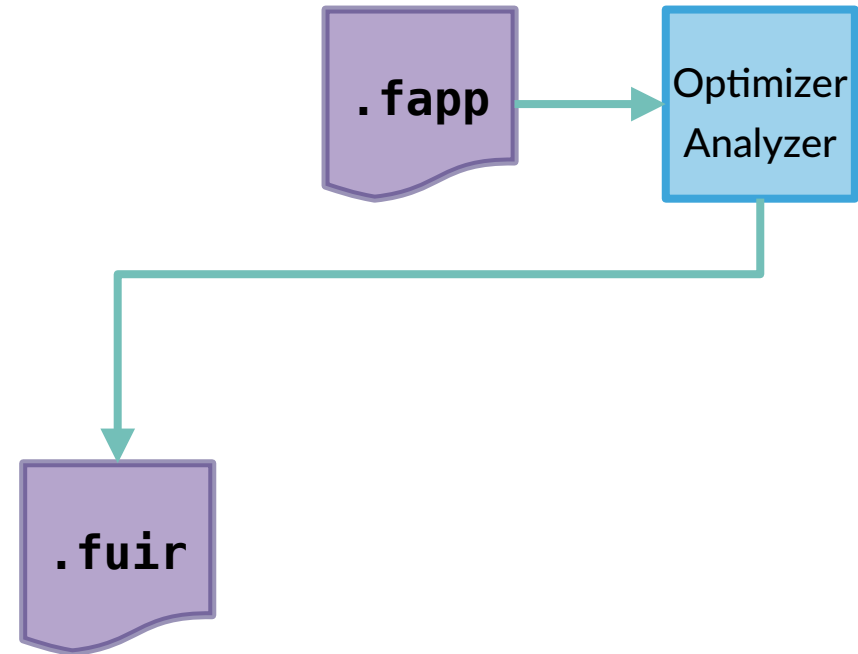
- Dead code removal
- Code Specialization
- Thread local data detection



Static Analysis in Optimizer/Analyzer

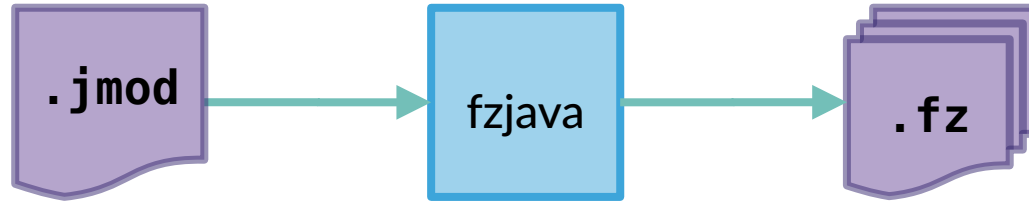
Analyze whole application

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

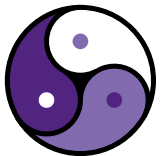


Other Tools: FZJava

Create Fuzion interface to Java module



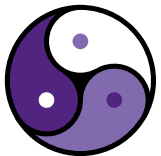
```
java.lang.System.out.println "Hello Java  !"
```



Other Tools: Language Server

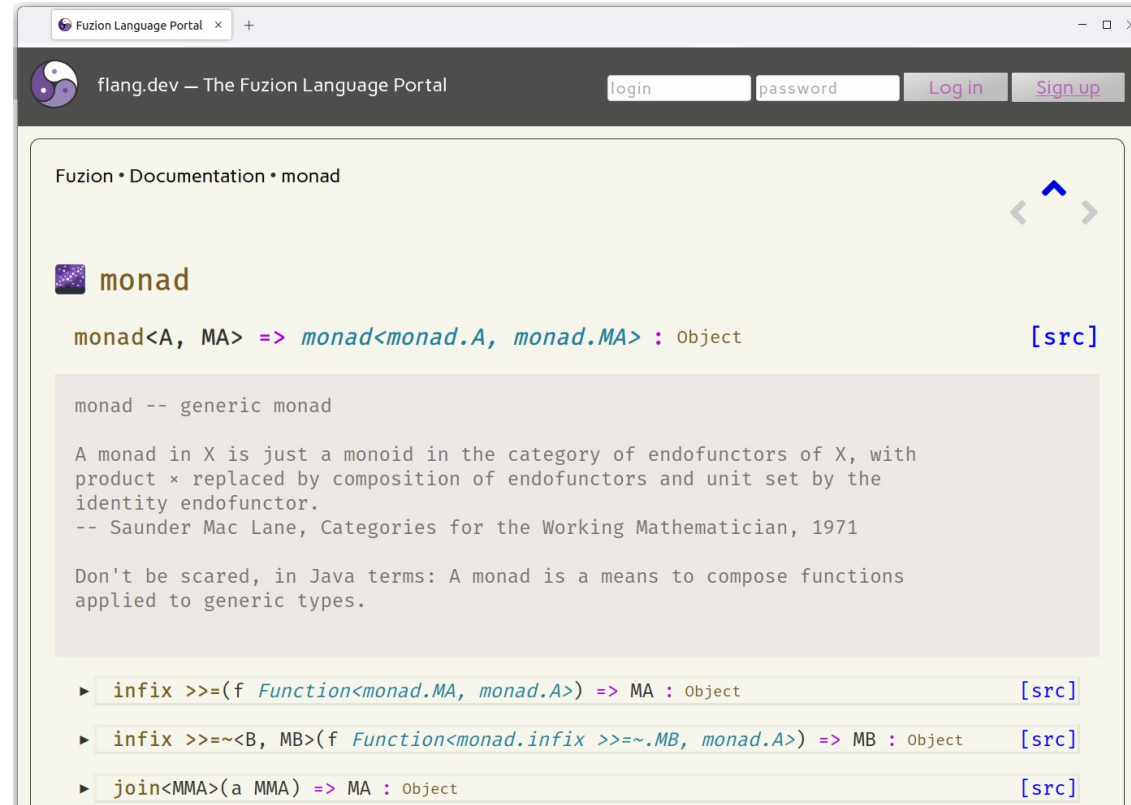
Support for IDEs and editors (vim, emacs)

- completion
- signature help
- documentation
- ...



Other Tools: FuzionDoc tool

Extract documentation from Fuzion source code



The screenshot shows a web browser window with the URL 'flang.dev - The Fuzion Language Portal'. The page displays the documentation for the 'monad' type. It includes a breadcrumb trail 'Fuzion • Documentation • monad', a title 'monad', and a signature 'monad<A, MA> => monad<monad.A, monad.MA> : Object'. Below this, there is a code block with the following content:

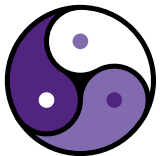
```
monad -- generic monad

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.
```

At the bottom of the page, there are three expandable sections for methods:

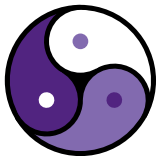
- `infix >>=(f Function<monad.MA, monad.A>) => MA : Object` [src]
- `infix >>=~<B, MB>(f Function<monad.infix >>=~.MB, monad.A>) => MB : Object` [src]
- `join<MMA>(a MMA) => MA : Object` [src]



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

