Fuzion — Java developer’s intro

Mapping Java's Features to Simpler Mechanisms

Fridtjof Siebert
Tokiwa Software GmbH

FOSDEM, 5. Feb 2023, Brussels

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

John Backus:

> *[My] work in functional programming languages failed, and would likely always fail, because it was easy to do hard things but incredibly difficult to do simple things.*

Source: Grady Booch, Twitter
https://twitter.com/Grady_Booch/status/1016041695501139968

Motivation: Fuzion Language

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 Resources

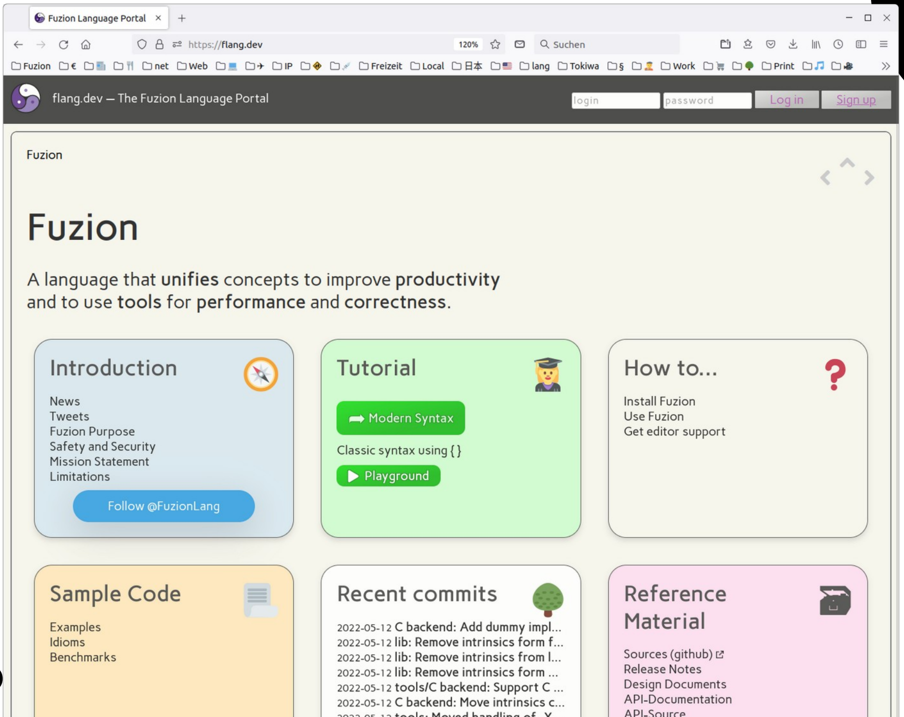
Fuzion available

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

Fuzion Resources

Fuzion available

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



Fuzion Language Portal

https://flang.dev

flang.dev — The Fuzion Language Portal

Fuzion

Fuzion

A language that **unifies** concepts to improve **productivity** and to use **tools** for **performance** and **correctness**.

Introduction
News
Tweets
Fuzion Purpose
Safety and Security
Mission Statement
Limitations
Follow @FuzionLang

Tutorial
➡ Modern Syntax
Classic syntax using { }
▶ Playground

How to...
Install Fuzion
Use Fuzion
Get editor support

Sample Code
Examples
Idioms
Benchmarks

Recent commits
2022-05-12 C backend: Add dummy impl...
2022-05-12 lib: Remove intrinsics form f...
2022-05-12 lib: Remove intrinsics from l...
2022-05-12 lib: Remove intrinsics form ...
2022-05-12 tools/C backend: Support C ...
2022-05-12 C backend: Move intrinsics c...

Reference Material
Sources (github)
Release Notes
Design Documents
API-Documentation
API-Source

Backing Company

➡ supports development of Fuzion

➡ currently four employees

➡ hiring

➡ searching for funding

This Talk

Fuzion and Algebraic Effects

- ➡ Quick Fuzion Intro
- ➡ Dangers of side-effects
- ➡ Algebraic Effects
- ➡ Examples / defining your own Effect / more examples

Short Fuzion Language Intro

Everything is a feature

Java equivalent

Short Fuzion Language Intro

Everything is a feature

```
demo is
```

Java equivalent

```
package demo;
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
```

Java equivalent

```
package demo;
class hello {



}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet unit is
```

Java equivalent

```
package demo;
class hello {
  void greet() {

}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet unit is
```

Java equivalent

```
package demo;
class hello {
  void greet() {

}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet unit is
```

Java equivalent

```
package demo;
class hello {
  void greet() {

}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet unit is
      say "Hello World!"
```

Java equivalent

```
package demo;
class hello {
  void greet() {
    System.out.println("Hello World!");
}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet unit is
      say "Hello World!"
```

Java equivalent

```
package demo;
class hello {
  void greet() {
    System.out.println("Hello World!");
}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet ⇒
      say "Hello World!"
```

Java equivalent

```
package demo;
class hello {
  void greet() {
    System.out.println("Hello World!");
}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet ⇒
      say "Hello World!"
```

Java equivalent

```
package demo;
class hello {
  void greet() {
    System.out.println("Hello World!");
}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet ⇒
      say "Hello World!"
```

```
demo.hello.greet
```

Java equivalent

```
package demo;
class hello {
  void greet() {
    System.out.println("Hello World!");
}}
```

```
class universe {
  public static void main(String[] args) {
    new demo.hello().greet();
}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet(a String) ⇒
      say "Hello $a!"



demo.hello.greet "World"
```

Java equivalent

```
package demo;
class hello {
  void greet(String a) {
    System.out.println("Hello "+a+"!");
}}



class universe {
  public static void main(String[] args) {
    new demo.hello.greet("World");
}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet(a String) ⇒
      say "Hello $a!"
```

```
demo.hello.greet "World"
```

Java equivalent

```
package demo;
class hello {
  void greet(String a) {
    System.out.println("Hello "+a+"!");
}}
```

```
class universe {
  public static void main(String[] args) {
    new demo.hello.greet("World");
}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet(a String) ⇒
      say "Hello $a!"
  hello2(def String) : hello is
    run ⇒ greet def

h := demo.hello2 "World"
h.run
```

Java equivalent

```
package demo;
class hello {
  void greet(String a) {
    System.out.println("Hello "+a+"!");
}}
class hello2 extends hello {
  String def;
  hello2(String d) { def = d; }
  void run() { greet(def); }
}
class universe {
  public static void main(String[] args) {
    var h = new demo.hello2("World");
    h.run();
}}
```

Short Fuzion Language Intro

Everything is a feature

```
demo is
  hello is
    greet(a String) ⇒
      say "Hello $a!"
  hello2(def String) : hello is
    run ⇒ greet def

h := demo.hello2 "World"
h.run
```

Java equivalent

```
package demo;
class hello {
  void greet(String a) {
    System.out.println("Hello "+a+"!");
}}
class hello2 extends hello {
  String def;
  hello2(String d) { def = d; }
  void run() { greet(def); }
}
class universe {
  public static void main(String[] args) {
    var h = new demo.hello2("World");
    h.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

(Side-) Effects and Safety / Security

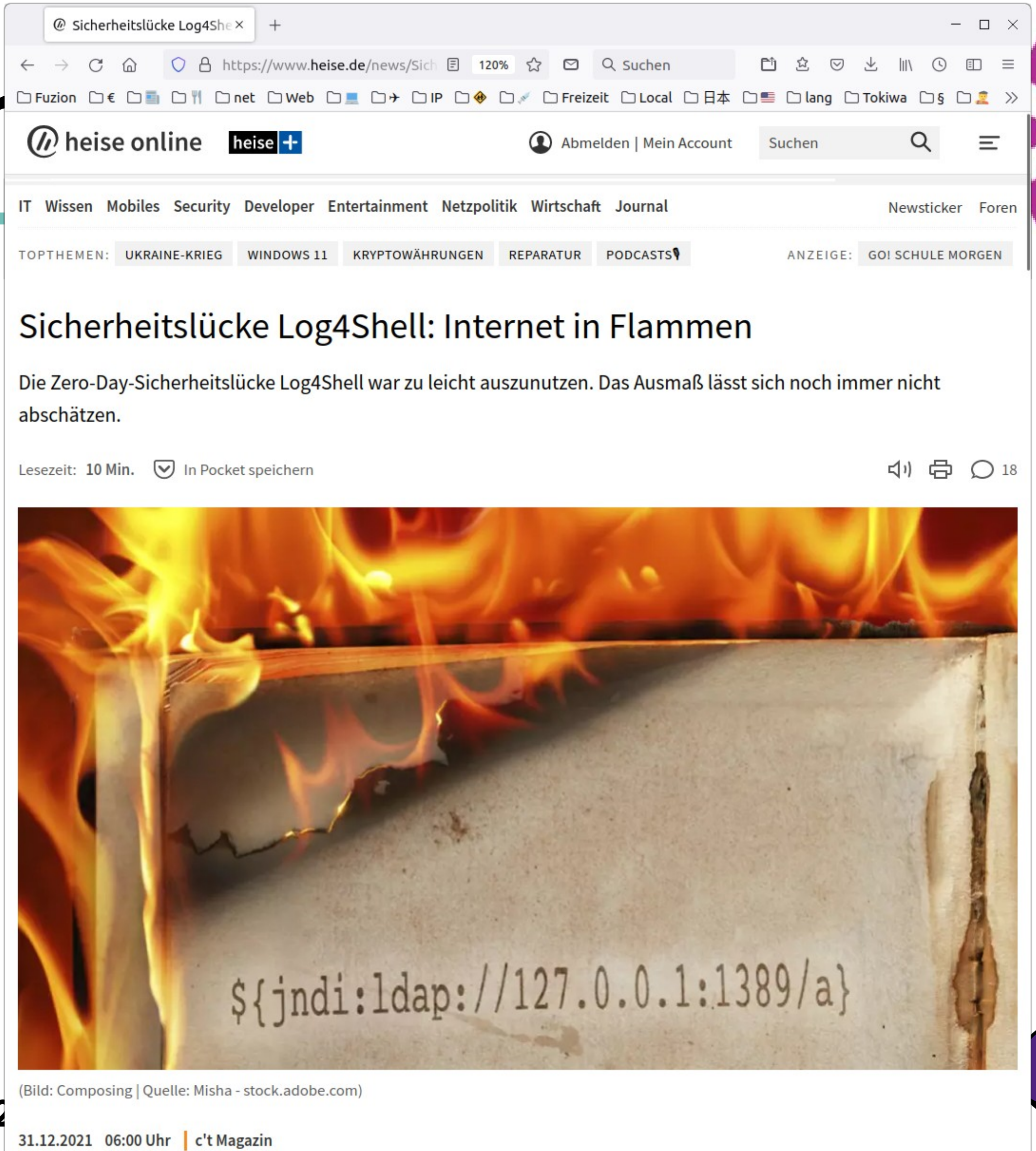
(Side-) Effects and Safety / Security

Recent security alerts

(Side-) Effects ar

Recent security alerts

➡ log4shell



Sicherheitslücke Log4She ×

https://www.heise.de/news/Sich 120% Suchen

Fuzion € net Web IP Freizeit Local 日本 lang Tokiwa §

heise online heise+ Abmelden | Mein Account Suchen

IT Wissen Mobiles Security Developer Entertainment Netzpolitik Wirtschaft Journal Newsticker Foren

TOPTHEMEN: UKRAINE-KRIEG WINDOWS 11 KRYPTOWÄHRUNGEN REPARATUR PODCASTS ANZEIGE: GO! SCHULE MORGEN

Sicherheitslücke Log4Shell: Internet in Flammen

Die Zero-Day-Sicherheitslücke Log4Shell war zu leicht auszunutzen. Das Ausmaß lässt sich noch immer nicht abschätzen.

Lesezeit: 10 Min. In Pocket speichern 18

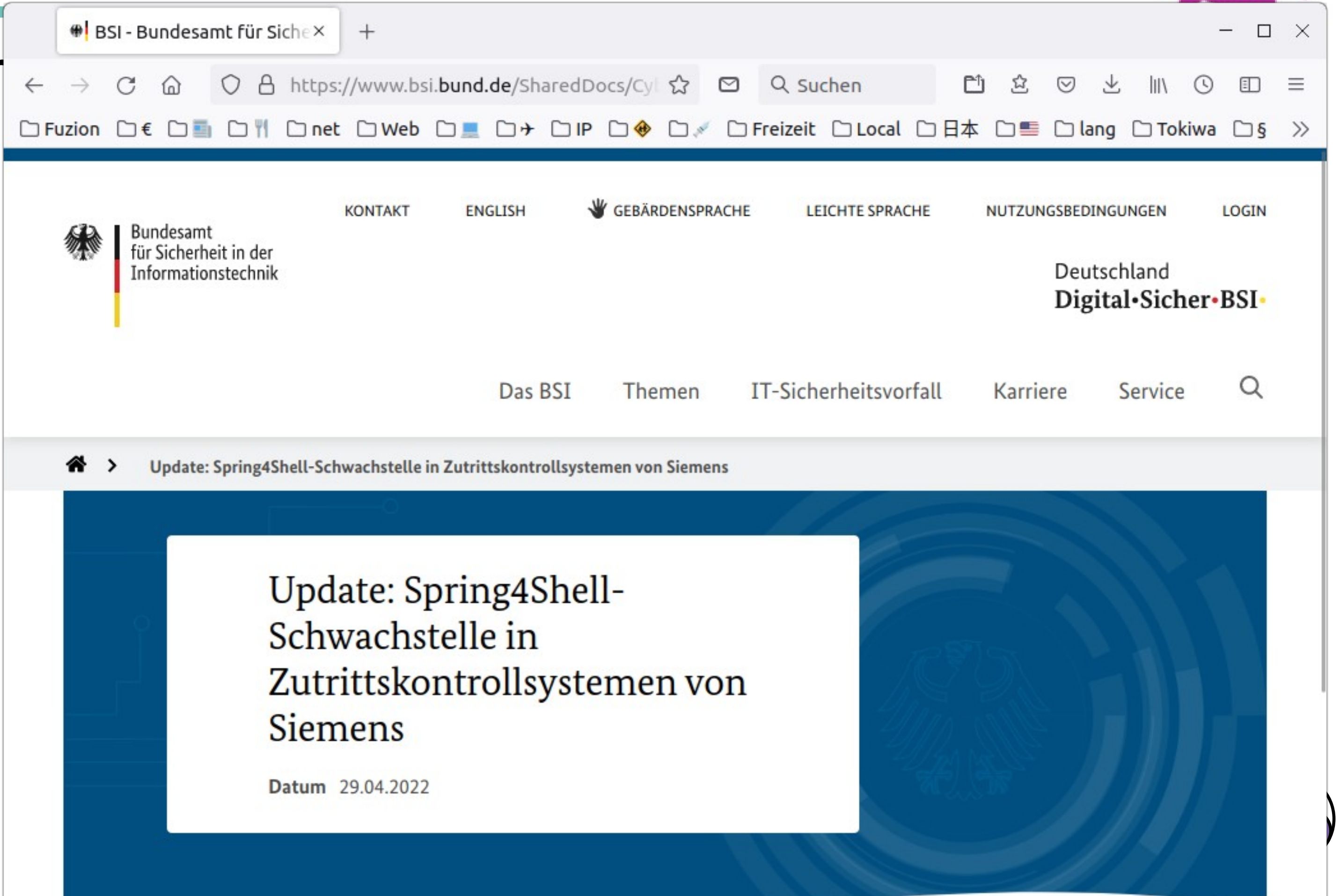
(Bild: Composing | Quelle: Misha - stock.adobe.com)

31.12.2021 06:00 Uhr | c't Magazin

(Side-) Effects and Safety / Security

Recent security aler

- ➡ log4shell
- ➡ SpringShell


BSI - Bundesamt für Siche

https://www.bsi.bund.de/SharedDocs/Cy

KONTAKT ENGLISH GEBÄRDENSPRACHE LEICHTE SPRACHE NUTZUNGSBEDINGUNGEN LOGIN

Bundesamt für Sicherheit in der Informationstechnik

Deutschland Digital•Sicher•BSI•

Das BSI Themen IT-Sicherheitsvorfall Karriere Service

Update: Spring4Shell-Schwachstelle in Zutrittskontrollsystemen von Siemens

Update: Spring4Shell-Schwachstelle in Zutrittskontrollsystemen von Siemens

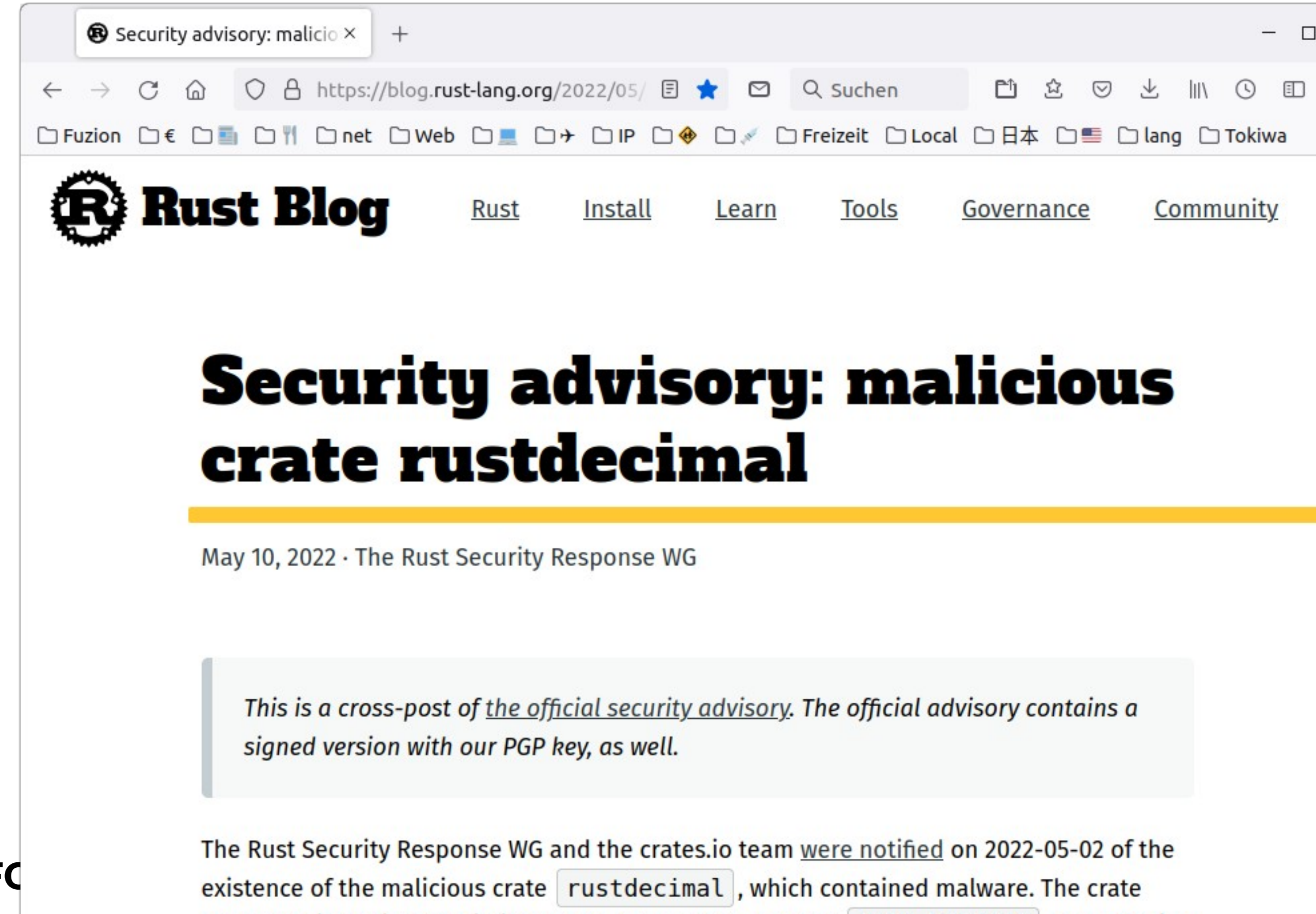
Datum 29.04.2022

TOKIWA software

(Side-) Effects and Safety / Security

Recent security alerts

- ➡ log4shell
- ➡ SpringShell
- ➡ rustdecimal crate



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

TOKIWA software

FC

(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

Algebraic Effects

Algebraic Effects

An Algebraic Effect is

- ➡ a set of (non-functional) **operations** code may perform

Algebraic Effects

An Algebraic Effect is

➡ a set of (non-functional) **operations** code may perform

- Java has one effect: **throws** with one operation **throw**

Algebraic Effects

An Algebraic Effect is

➡ a set of (non-functional) **operations** code may perform

Algebraic Effects

An Algebraic Effect is

- ➡ a set of (non-functional) **operations** code may perform
- ➡ the operations can **resume** or **abort**

Algebraic Effects

An Algebraic Effect is

- ➡ a set of (non-functional) **operations** code may perform
- ➡ the operations can **resume** or **abort**
- ➡ the operations can be implemented by an **effect handler**

Algebraic Effects

An Algebraic Effect is

➡ a set of (non-functional) **operations** code may perform

➡ the operations can **resume** or **abort**

➡ the operations can be implemented by an **effect handler**

➡ Effects may be **nested**

Algebraic Effects

An Algebraic Effect is

- ➡ a set of (non-functional) **operations** code may perform
- ➡ the operations can **resume** or **abort**
- ➡ the operations can be implemented by an **effect handler**
- ➡ Effects may be **nested**
- ➡ Effects may be seen as required **capabilities**
  - code that throws exception requires capability to catch

Example: my_exc effect

Exception Effect

Example: my_exc effect

Exception Effect

```
my_exc : simpleEffect is
```

Example: my_exc effect

Exception Effect

```
my_exc : simpleEffect is
  throw ⇒ abort
```

Example: my_exc effect

Exception Effect

```
my_exc : simpleEffect is
  throw ⇒ abort

f ⇒


  my_exc.env.throw
```

Example: my_exc effect

Exception Effect

```
my_exc : simpleEffect is
  throw ⇒ abort

f ! my_exc ⇒

  my_exc.env.throw
```

Example: my_exc effect

Exception Effect

```
my_exc : simpleEffect is
  throw ⇒ abort

f ! my_exc ⇒
  say "before throw"
  my_exc.env.throw
  say "after throw  *** not reachable ***"
```

Example: my_exc effect

Exception Effect

```
my_exc : simpleEffect is
  throw ⇒ abort

f ! my_exc ⇒
  say "before throw"
  my_exc.env.throw
  say "after throw  *** not reachable ***"


my_exc.use ()→f
```

Example: my_exc effect

Exception Effect

```
my_exc : simpleEffect is
  throw ⇒ abort

f ! my_exc ⇒
  say "before throw"
  my_exc.env.throw
  say "after throw  *** not reachable ***"

say "install my_exc"
my_exc.use ()→f
say "done with my_exc"
```

Example: my_exc effect

Exception Effect

```
my_exc : simpleEffect is
  throw => abort

f ! my_exc =>
  say "before throw"
  my_exc.env.throw
  say "after throw  *** not reachable ***"

say "install my_exc"
my_exc.use ()->f
say "done with my_exc"
```

```
> fz exception.fz
```

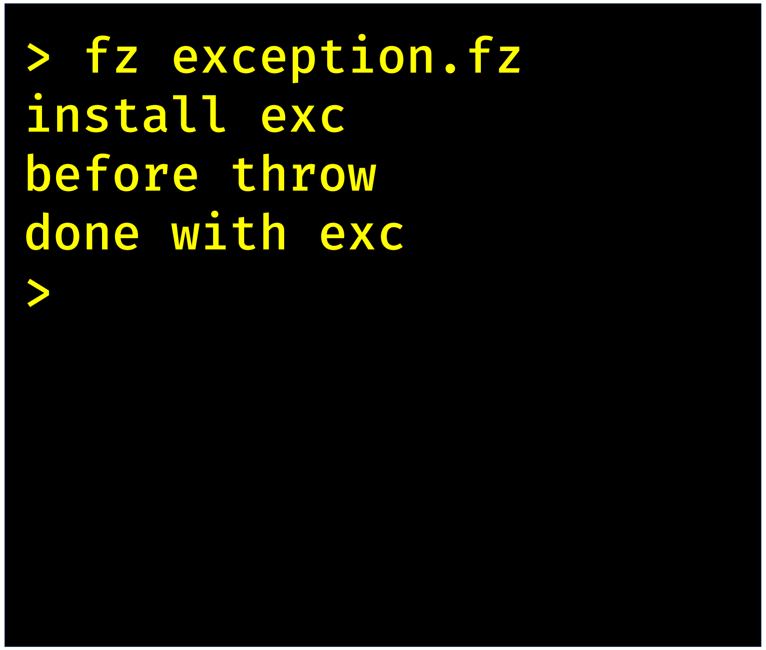
Example: my_exc effect

Exception Effect

```
my_exc : simpleEffect is
  throw ⇒ abort

f ! my_exc ⇒
  say "before throw"
  my_exc.env.throw
  say "after throw  *** not reachable ***"

say "install my_exc"
my_exc.use ()→f
say "done with my_exc"
```



```
> fz exception.fz
install exc
before throw
done with exc
>
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
>
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
x := 123
say x
x := 2*x
say x
```

```
>
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
x := 123
say x
x := 2*x
say x
```

```
> fz mutate1.fz
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
x := 123
say x
x := 2*x
say x
```

```
> fz mutate1.fz
123
246
>
```

Fuzion and Mutation

Fields in Fuzion are immutable

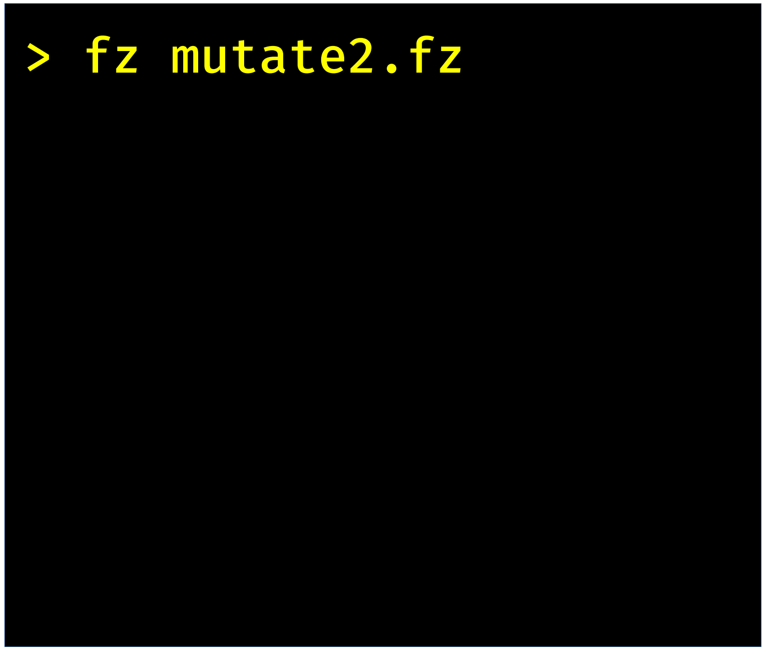
```
show_x ⇒ say x
x := 123
show_x
x := 2*x
show_x
```

```
>
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
show_x ⇒ say x
x := 123
show_x
x := 2*x
show_x
```

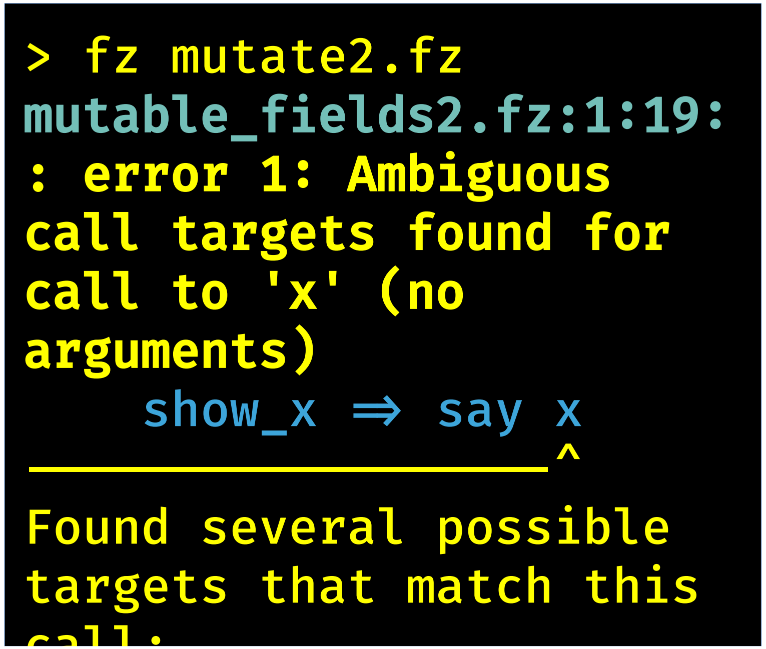


```
> fz mutate2.fz
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
show_x ⇒ say x
x := 123
show_x
x := 2*x
show_x
```

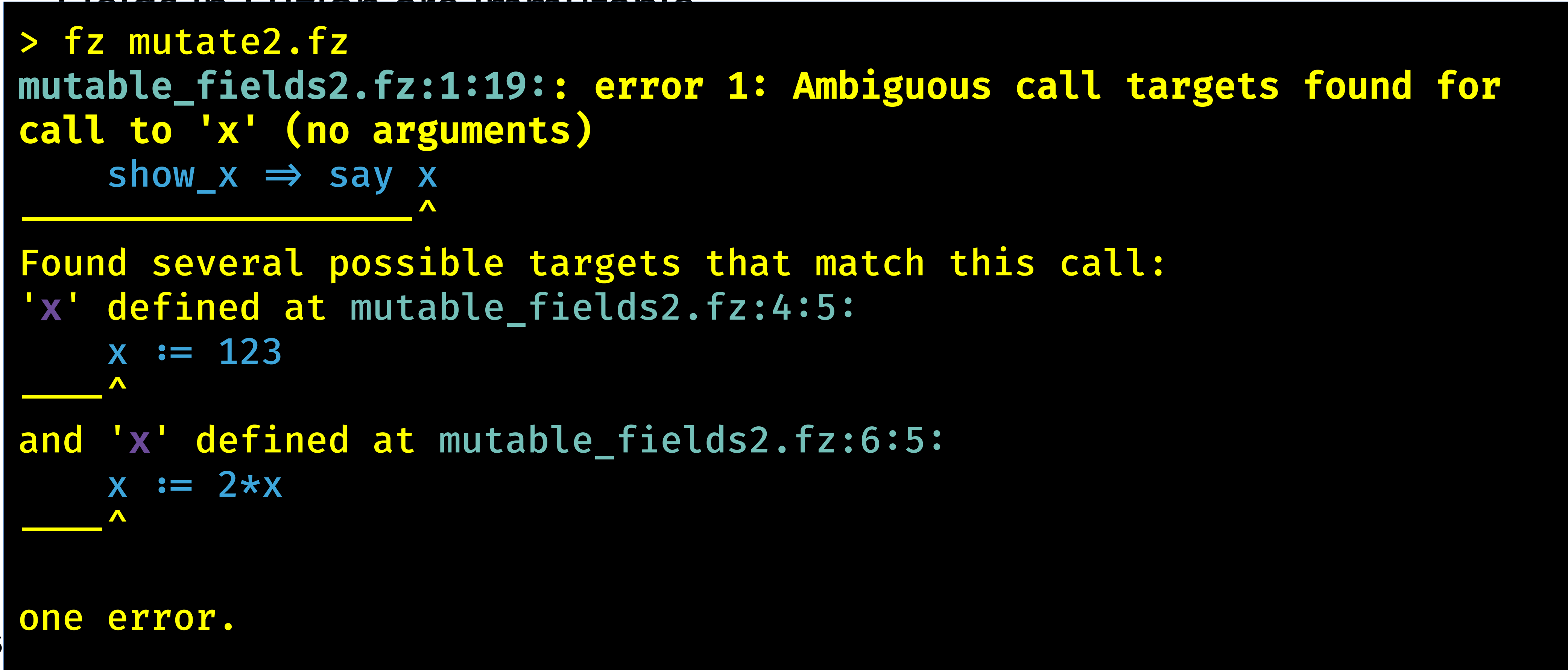


```
> fz mutate2.fz
mutable_fields2.fz:1:19:
: error 1: Ambiguous
call targets found for
call to 'x' (no
arguments)
    show_x ⇒ say x
__________________^

Found several possible
targets that match this
call:
```

Fuzion and Mutation

Fields in Fuzion are immutable



```
> fz mutate2.fz
mutable_fields2.fz:1:19:: error 1: Ambiguous call targets found for
call to 'x' (no arguments)
    show_x ⇒ say x
__________________^

Found several possible targets that match this call:
'x' defined at mutable_fields2.fz:4:5:
    x := 123
____^
and 'x' defined at mutable_fields2.fz:6:5:
    x := 2*x
____^


one error.
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
    show_x ⇒ say x
    x := 123
    show_x
    x := 2*x
    show_x
```

```
>
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
show_x ⇒ say x
x := mut 123
show_x
x ← 2 * x.get
show_x
```

```
>
```

Fuzion and Mutation

Fields in Fuzion are immutable

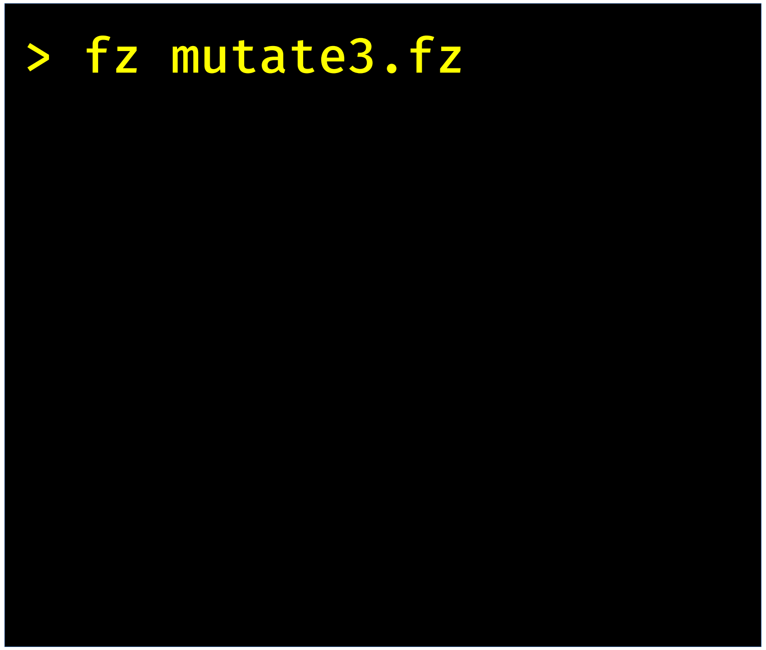
```
show_x ⇒ say x
x := mut 123
show_x
x ← 2 * x.get
show_x
```

```
>
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
show_x ⇒ say x
x := mut 123
show_x
x ← 2 * x.get
show_x
```

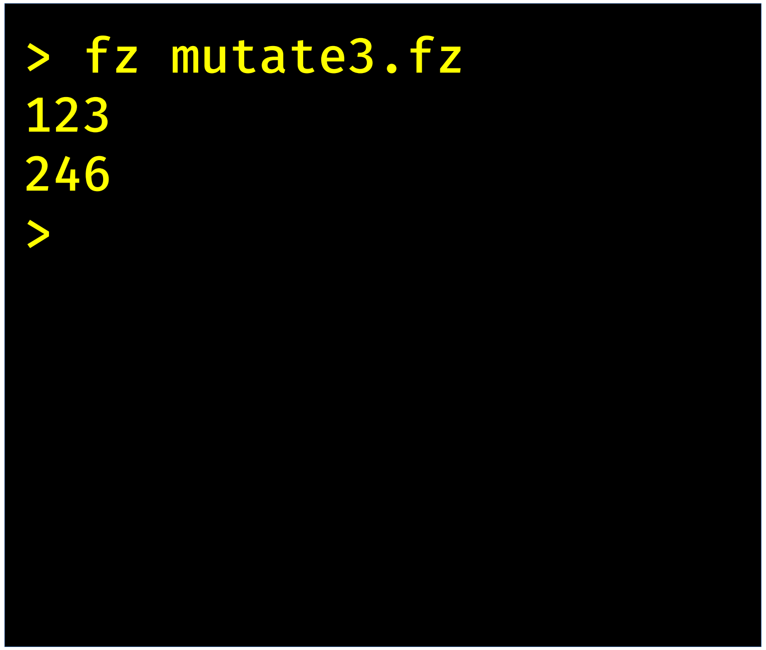


```
> fz mutate3.fz
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
show_x ⇒ say x
x := mut 123
show_x
x ← 2 * x.get
show_x
```



```
> fz mutate3.fz
123
246
>
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
show_x ⇒ say x
x := mut 123
show_x
x ← 2 * x.get
show_x
```

```
> fz mutate3.fz
123
246
> fz -effects mutate3.fz
```

Fuzion and Mutation

Fields in Fuzion are immutable

```
show_x ⇒ say x
x := mut 123
show_x
x ← 2 * x.get
show_x
```

```
> fz mutate3.fz
123
246
> fz -effects mutate3.fz
io.out
mutate
>
```

Fuzion and Mutation

Loop index variables

```
for
  i := 0, i + 1
while i < 10 do
  say i
say "done."
```

```
>
```

Fuzion and Mutation

Loop index variables

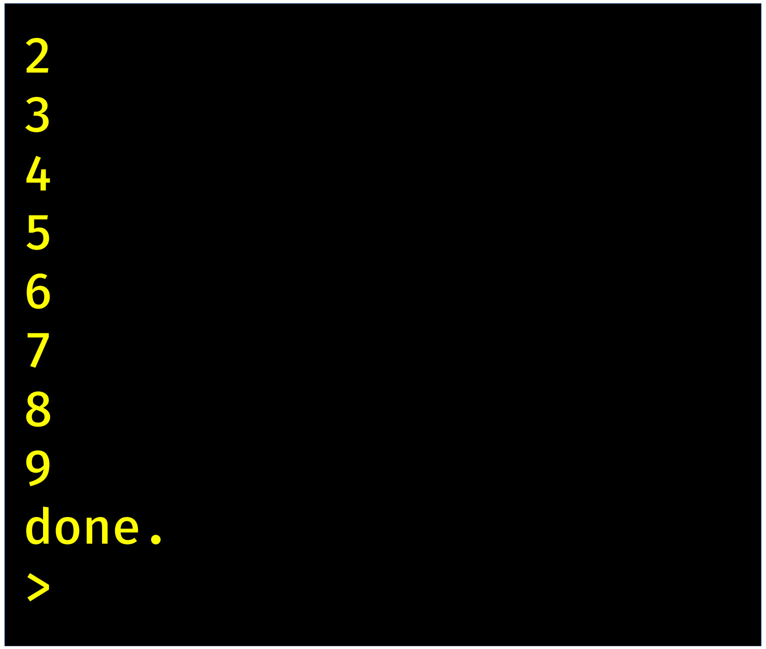
```
for
  i := 0, i + 1
while i < 10 do
  say i
say "done."
```

```
>  loop.fz
```

Fuzion and Mutation

Loop index variables

```
for
  i := 0, i + 1
while i < 10 do
  say i
say "done."
```

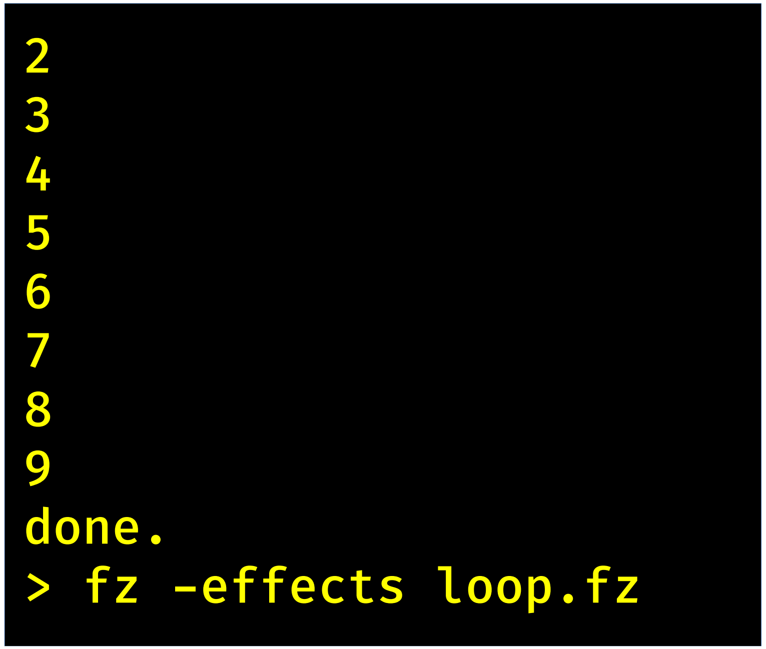


```
2
3
4
5
6
7
8
9
done.
>
```

Fuzion and Mutation

Loop index variables

```
for
  i := 0, i + 1
while i < 10 do
  say i
say "done."
```

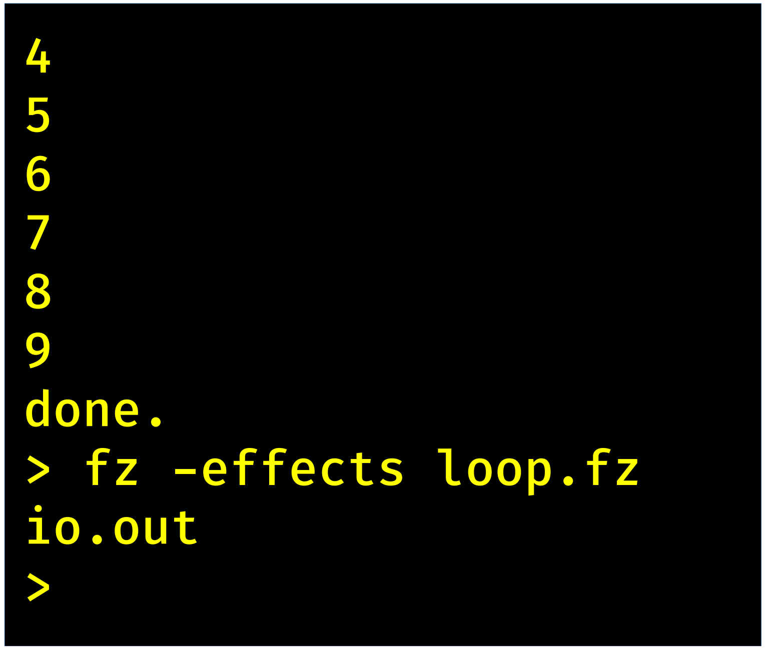


```
2
3
4
5
6
7
8
9
done.
> fz -effects loop.fz
```

Fuzion and Mutation

Loop index variables

```
for
  i := 0, i + 1
while i < 10 do
  say i
say "done."
```



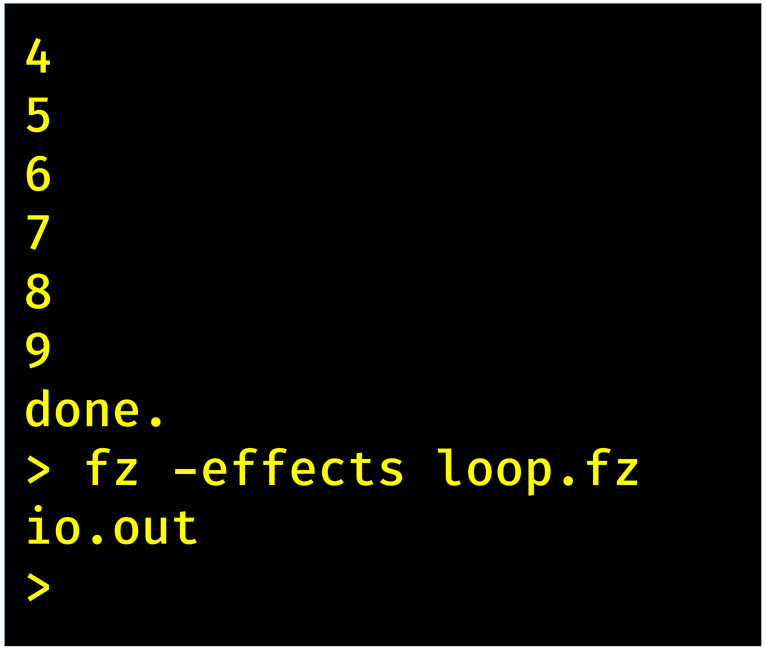
```
4
5
6
7
8
9
done.
> fz -effects loop.fz
io.out
>
```

Fuzion and Mutation

Loop index variables

```
for
  i := 0, i + 1
while i < 10 do
  say i
say "done."
```

No variable is mutated, a new instance is created per iteration.



```
4
5
6
7
8
9
done.
> fz -effects loop.fz
io.out
>
```

Error Handling

```
>
```

Error Handling

Division by zero

```
    divide (a, b i32) ⇒
      a / b
```

```
>
```

Error Handling

Division by zero

```
divide (a, b i32) ⇒
  a / b




show_div(a, b i32) ⇒
  v := divide a b
  say "result is $v"


show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Error Handling

Division by zero

```
divide (a, b i32) ⇒
  a / b




show_div(a, b i32) ⇒
  v := divide a b
  say "result is $v"


show_div 100 12
show_div 100 0
show_div 10 100
```

```
> fz div.fz
```

Error Handling

Division by zero

```
divide (a, b i32) ⇒
  a / b




show_div(a, b i32) ⇒
  v := divide a b
  say "result is $v"


show_div 100 12
show_div 100 0
show_div 10 100
```

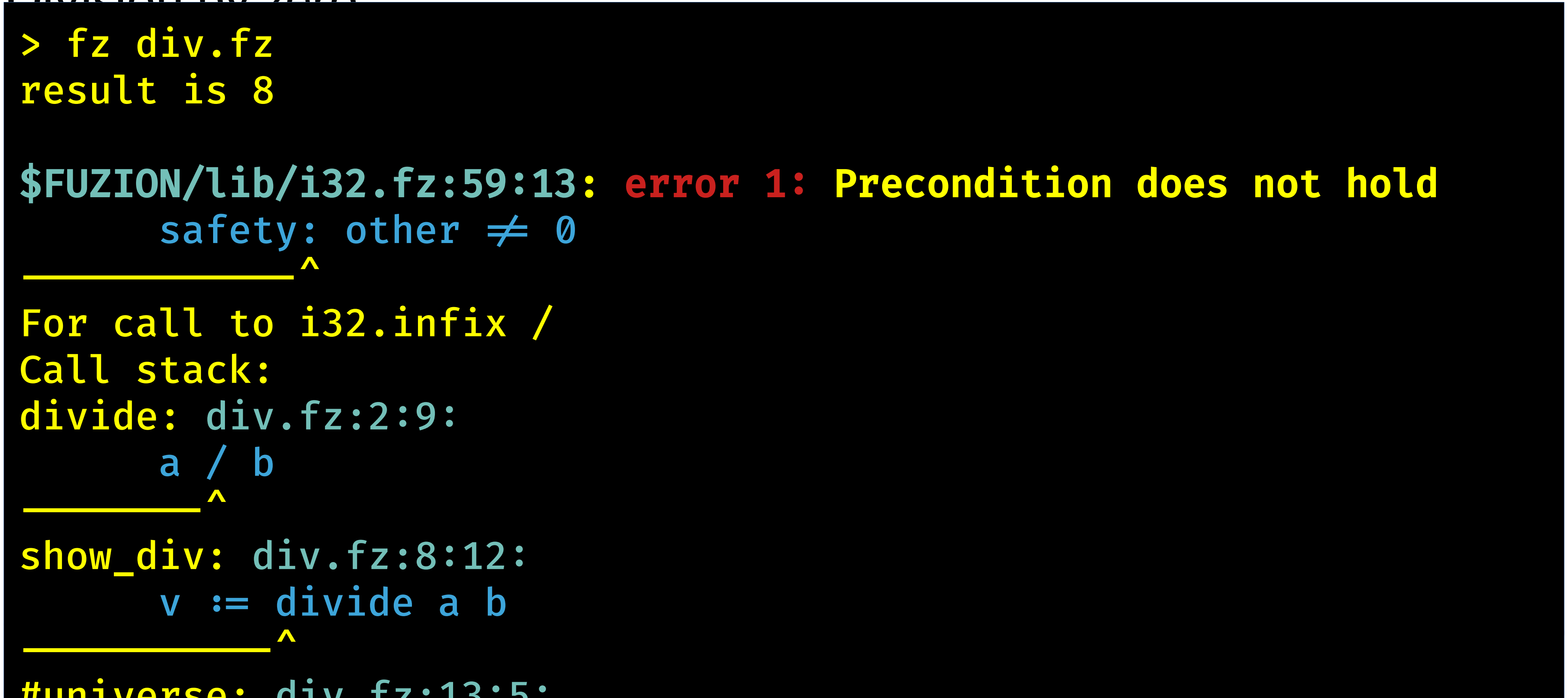
```
> fz div.fz
result is 8

$FUZION/lib/
i32.fz:59:13: error 1:
Precondition does not
hold
      safety: other ≠ 0
____________^

For call to i32.infix /
Call stack:
```

Error Handling

Division by zero



```
> fz div.fz
result is 8

$FUZION/lib/i32.fz:59:13: error 1: Precondition does not hold
      safety: other ≠ 0
____________^
For call to i32.infix /
Call stack:
divide: div.fz:2:9:
      a / b
________^
show_div: div.fz:8:12:
      v := divide a b
___________^
#universe: div.fz:13:5:
```

Error Handling

Division by zero

```
divide (a, b i32) ⇒
  a / b




show_div(a, b i32) ⇒
  v := divide a b
  say "result is $v"


show_div 100 12
show_div 100 0
show_div 10 100
```

```
> fz div.fz
result is 8

$FUZION/lib/
i32.fz:59:13: error 1:
Precondition does not
hold
      safety: other ≠ 0
______________^

For call to i32.infix /
Call stack:
```

Using choice type `outcome`

Using outcome

```
divide (a, b i32) ⇒
  a / b




show_div(a, b i32) ⇒
  v := divide a b
  say "result is $v"


show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Using choice type `outcome`

Using outcome

```
divide (a, b i32) outcome i32 is
  a / b




show_div(a, b i32) ⇒
  v := divide a b
  say "result is $v"


show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Using choice type `outcome`

Using outcome

```
divide (a, b i32) outcome i32 is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  v := divide a b
  say "result is $v"


show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

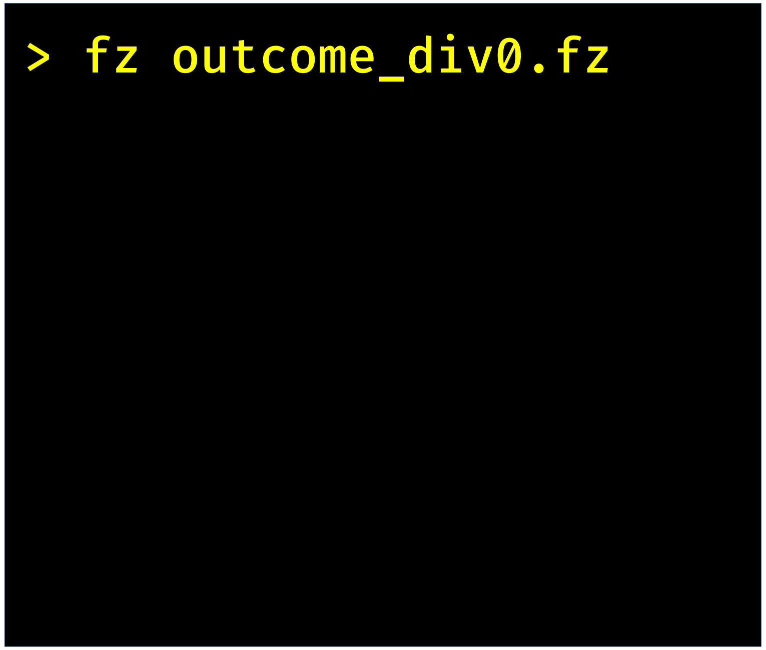
Using choice type `outcome`

Using outcome

```
divide (a, b i32) outcome i32 is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  v := divide a b
  say "result is $v"


show_div 100 12
show_div 100 0
show_div 10 100
```



```
> fz outcome_div0.fz
```

Using choice type `outcome`

Using outcome

```
divide (a, b i32) outcome i32 is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  v := divide a b
  say "result is $v"


show_div 100 12
show_div 100 0
show_div 10 100
```

```
> fz outcome_div0.fz
result is 8
result is --error: div
by 0!--
result is 0
>
```

Using choice type `outcome`

Using outcome

```
divide (a, b i32) outcome i32 is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  v := divide a b
  say "result is $v"


show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Using choice type `outcome`

Using outcome

```
divide (a, b i32) outcome i32 is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  match divide a b
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

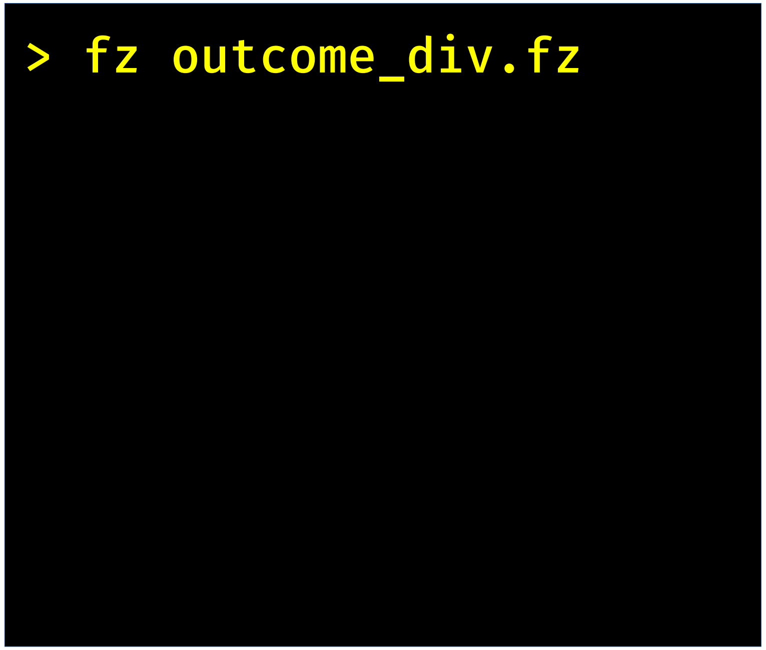
Using choice type `outcome`

Using outcome

```
divide (a, b i32) outcome i32 is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  match divide a b
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```



```
> fz outcome_div.fz
```

Using choice type `outcome`

Using outcome

```
divide (a, b i32) outcome i32 is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  match divide a b
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
> fz outcome_div.fz
ok, result is 8
not ok: error: div by 0!
ok, result is 0
>
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) outcome i32 is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  match divide a b
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) outcome i32 is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  match divide a b
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) i32 ! try is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  match divide a b
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) i32 ! try is
  if b = 0 then
    error "div by 0!"
  else
    a / b

show_div(a, b i32) ⇒
  match divide a b
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) i32 ! try is
  if b = 0 then
    try.env.raise (error "div by 0!")
  else
    a / b

show_div(a, b i32) ⇒
  match divide a b
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) i32 ! try is
  if b = 0 then
    try.env.raise (error "div by 0!")
  else
    a / b

show_div(a, b i32) ⇒
  match divide a b
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) i32 ! try is
  if b = 0 then
    try.env.raise (error "div by 0!")
  a / b


show_div(a, b i32) ⇒
  match divide a b
    v i32    ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) i32 ! try is
  if b = 0 then
    try.env.raise (error "div by 0!")
  a / b


show_div(a, b i32) ⇒
  match divide a b
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) i32 ! try is
  if b = 0 then
    try.env.raise (error "div by 0!")
  a / b


show_div(a, b i32) ⇒
  match try (() → divide a b)
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
>
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) i32 ! try is
  if b = 0 then
    try.env.raise (error "div by 0!")
  a / b


show_div(a, b i32) ⇒
  match try (() → divide a b)
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
> try_div.fz
```

Throwing Errors using try-effect

Using try-effect

```
divide (a, b i32) i32 ! try is
  if b = 0 then
    try.env.raise (error "div by 0!")
  a / b


show_div(a, b i32) ⇒
  match try (() → divide a b)
    v i32   ⇒ say "ok, result is $v"
    e error ⇒ say "not ok: $e"

show_div 100 12
show_div 100 0
show_div 10 100
```

```
> try_div.fz
ok, result is 8
not ok: error: div by 0!
ok, result is 0
>
```

Fuzion: Status

Fuzion still under development

- ➡ language definition slowly getting more stable
- ➡ base library work in progress
- ➡ current implementation providing JVM and C backends
- ➡ Basic analysis tools available

Conclusion

Fuzion is a new functional language

- ➡ Java maps very well to Fuzion
- ➡ effects encapsulate non-functional aspects
  - mutability
  - i/o
  - exceptions
- ➡ have a look, get involved!

@fuzion@types.pl
@FuzionLang
https://flang.dev
github.com/tokiwa-software/fuzion