

Fuzion Language Update

The marathon run from
a language prototype to
a full implementation and toolchain.



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

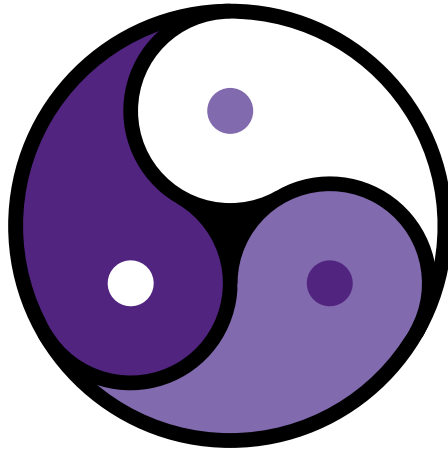
→ we need to ensure correctness

Fuzion Summary

Fuzion

- uses the **feature** as its main concept
- is **statically typed**
- has **inheritance** and **redefinition**
- uses **value types** and **dynamic (ref) types**
- encourages **immutability**
- offloads tasks and decisions from developers to **tools**

Fuzion Logo



Backing Company



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

Using White Space





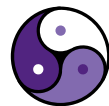
Using White Space

Avoiding bloat caused by

; { } () ,

and

- use whitespace instead
- increase clarity and readability
- avoid errors



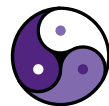
Using White Space: Semicolons

Flat line feeds: same indentation

```
stmt1  
stmt2  
stmt3
```

Equivalent to

```
stmt1;  
stmt2;  
stmt3;
```



Using White Space: Semicolons

Flat line feeds: same indentation

```
stmt1  
stmt2  
stmt3
```

Equivalent to

```
stmt1;  
stmt2;  
stmt3;
```

Statement separation in single line using ;

```
stmt1; stmt2; stmt3
```



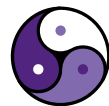
Using White Space: Blocks

Indenting line feeds $\Leftrightarrow$ {

Equivalent to

```
if cond
    stmt1
else
    stmt2
```

```
if cond {
    stmt1
} else {
    stmt2
}
```



Using White Space: Blocks

Indenting line feeds $\Leftrightarrow$ {

Equivalent to

```
if cond
    stmt1
else
    stmt2
```

```
if cond {
    stmt1
} else {
    stmt2
}
```

Blocks in a single line:

```
if cond then stmt1 else stmt2
```



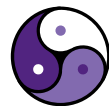
Using White Space: Calls

Calls do not require (,)

`f a b c`

Equivalent to

`f(a, b, c)`



Using White Space: Calls

Calls do not require (,)

Equivalent to

`f a b c`

`f(a, b, c)`

nesting

`f (g x y) (h z) c`

`f(g(x, y), h(z), c)`

Using White Space: Calls & Tuples



Calls with two arguments

f a b

Equivalent to

f(a, b)

Using White Space: Calls & Tuples



Calls with two arguments

`f a b`

Equivalent to

`f(a, b)`

Call with one tuple as argument

`g (a, b)`

`g((a, b))`

Using White Space: Calls & Tuples



Calls with two arguments

`f a b`

Call with one tuple as argument

`g (a, b)`

Equivalent to

`f(a, b)`

`g((a, b))`



Using White Space: Arrays

Accessing array **a** at indices **i**, **j**

a[i, j]



Using White Space: Arrays

Accessing array **a** at indices **i**, **j**

a[i, j]

Creating array with elements **x**, **y**, **z**

[x, y, z]



Using White Space: Arrays

Accessing array **a** at indices **i**, **j**

a[i, j]

Creating array with elements **x**, **y**, **z**

[x, y, z]

Passing this array as argument in call to **f**

f [x, y, z]



Using White Space: Arrays

Accessing array **a** at indices **i**, **j**

a[**i**, **j**]

Creating array with elements **x**, **y**, **z**

[**x**, **y**, **z**]

Passing this array as argument in call to **f**

f [**x**, **y**, **z**]



Using White Space: Operators

Using prefix operator on argument

f -x



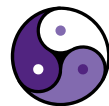
Using White Space: Operators

Using prefix operator on argument

f -x

Equivalent to

f (-x)



Using White Space: Operators

Using prefix operator on argument

f -x

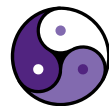
Equivalent to

f (-x)

while

f - x

f-x



Using White Space: Operators

Using prefix operator on argument

Equivalent to

f -x

f (-x)

while

f - x

f-x

and

f- x

(f-) x



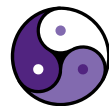
Using White Space: Operators

Using infix operators in calls

f a-b

Equivalent to

f (a-b)



Using White Space: Operators

Using infix operators in calls

f a-b

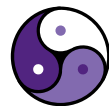
while

f a -b

Equivalent to

f (a-b)

f (a) (-b)



Using White Space: Operators

Using infix operators in calls

f a-b

while

f a -b

and

f a - b

Equivalent to

f (a-b)

f (a) (-b)

(f a) -b



Using White Space: In Java

White space has important semantics in other languages as well, e.g., Java code

```
if (cc) x(); elsewhere();
```

is very different to

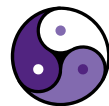
```
if (cc) x(); else where();
```

Semantic white space is basically a matter of practice and habit.

Type Inference



Avoid the need for explicit types.



Type Inference

Avoid the need for explicit types.

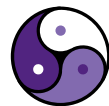
Occurs at assignments:

x := expr explicit assignment

f(a T) => g(a) implicit assignment to feature result

h 3.14e3 assignment of value to feature argument

Types propagates in both direction: forward and backward!



Type Inference: Feature Results

Expression type propagated forward:

`v := 123`

type `i32`, default for
integer literal

`s := "hello"`

type `string`

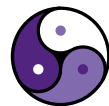
`h(v u8) => v >> 4`

type `u8`

`origin => point 0 0`

type `point<i32>`

Type Inference: Type Arguments



Type of actual arguments in call propagated to call's type arguments:

```
square<T: integer<T>> (a T) => a * a
```

```
s2 := square<i32> 123    using explicit type parameter
```

```
s2 := square 123        using type inference
```



Type Inference: Lambdas

Backward propagation: Lambda type defined by what it is assigned to:

`si(f (string, i32) -> string) => say (f "hello" 3)`

`ii(f (i32, i32) -> i32) => say (f 10 3)`

`si (s,i -> s * i)`

lambda type `(string, i32) -> string`

`ii (s,i -> s * i)`

lambda type `(i32, i32) -> i32`

Type Inference: Numeric Literals



Backward propagation: Target type defines type of literal

Type Inference: Numeric Literals



Backward propagation: Target type defines type of literal

u u16 := 54321

literal of type **u16**

Type Inference: Numeric Literals



Backward propagation: Target type defines type of literal

u **u16** := **54321**

literal of type **u16**

v **u16** := **54.321E3**

literal of type **u16**

Type Inference: Numeric Literals



Backward propagation: Target type defines type of literal

u **u16** := 54321

literal of type **u16**

v **u16** := 54.321E3

literal of type **u16**

w := **u16** 54321

literal of type **u16**



Type Inference: Numeric Literals

Backward propagation: Target type defines type of literal

u **u16** := 54321 literal of type **u16**

v **u16** := 54.321E3 literal of type **u16**

w := **u16** 54321 literal of type **u16**

x **f64** := 54321 literal of type **f64**

Type Inference: Numeric Literals



Backward propagation: Target type defines type of literal

u **u16** := 54321

literal of type **u16**

v **u16** := 54.321E3

literal of type **u16**

w := **u16** 54321

literal of type **u16**

x **f64** := 54321

literal of type **f64**

y **f64** := 54.321E3

literal of type **f64**



Type Inference: Numeric Literals

Backward propagation: Target type defines type of literal

u **u16** := 54321 literal of type **u16**

v **u16** := 54.321E3 literal of type **u16**

w := **u16** 54321 literal of type **u16**

x **f64** := 54321 literal of type **f64**

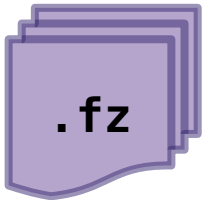
y **f64** := 54.321E3 literal of type **f64**

z := **f64** 54321 literal of type **f64**

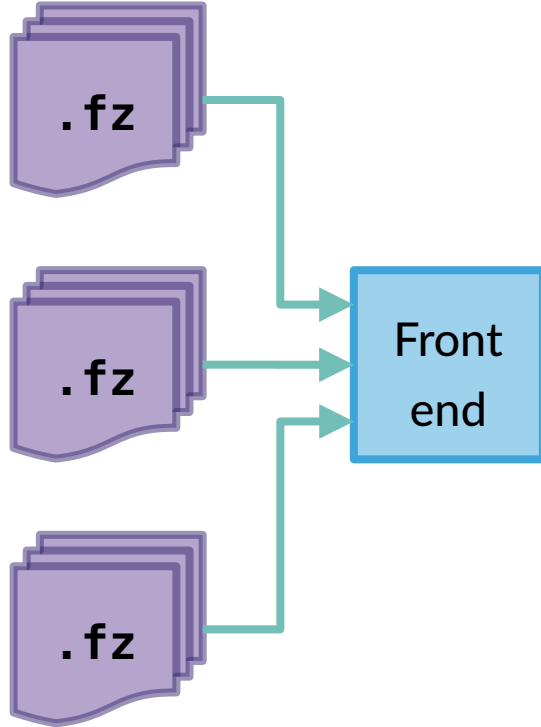
Fuzion Toolchain: Planned Design



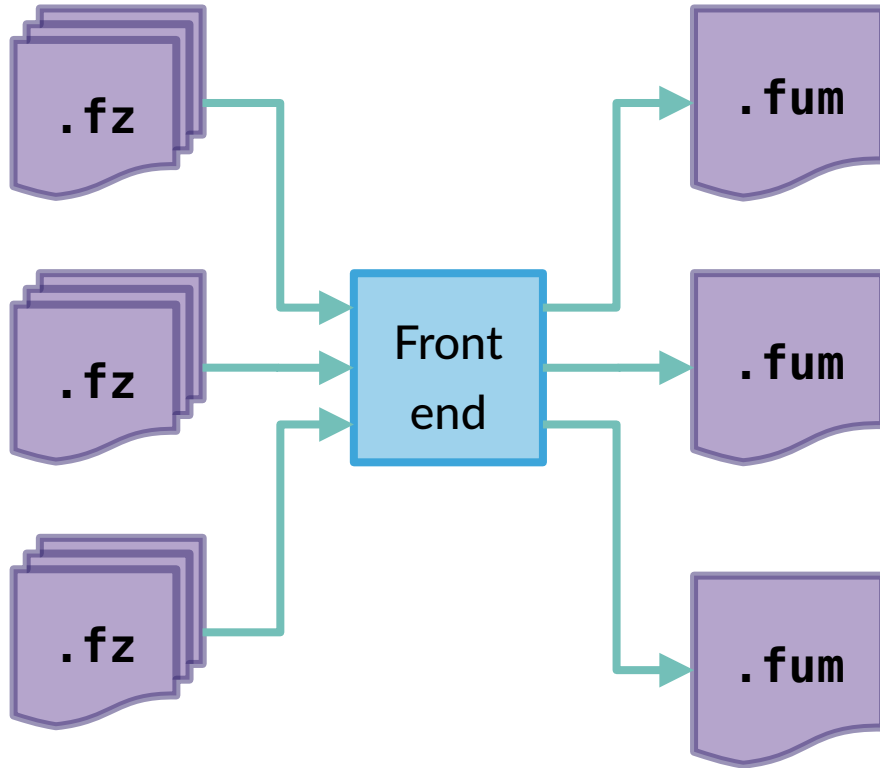
Fuzion Toolchain: Planned Design



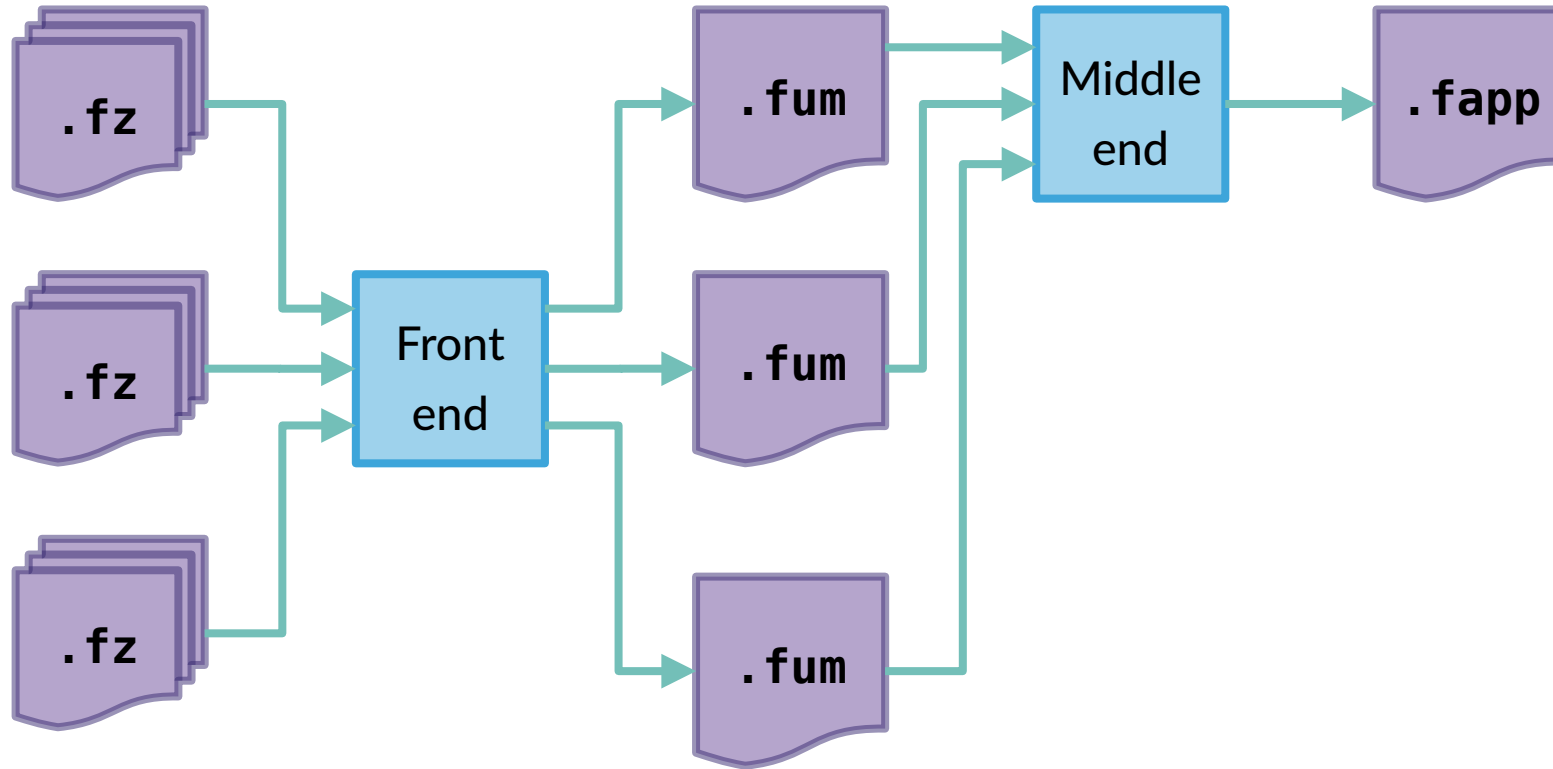
Fuzion Toolchain: Planned Design



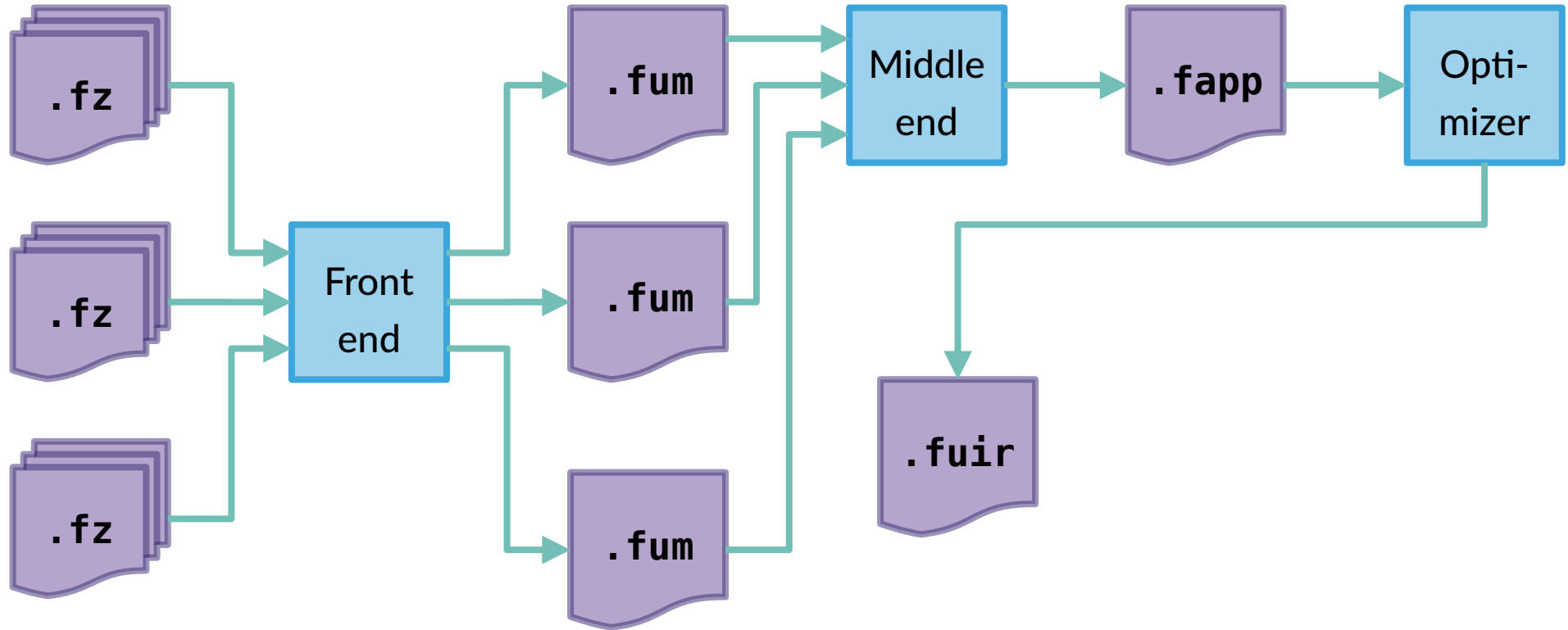
Fuzion Toolchain: Planned Design



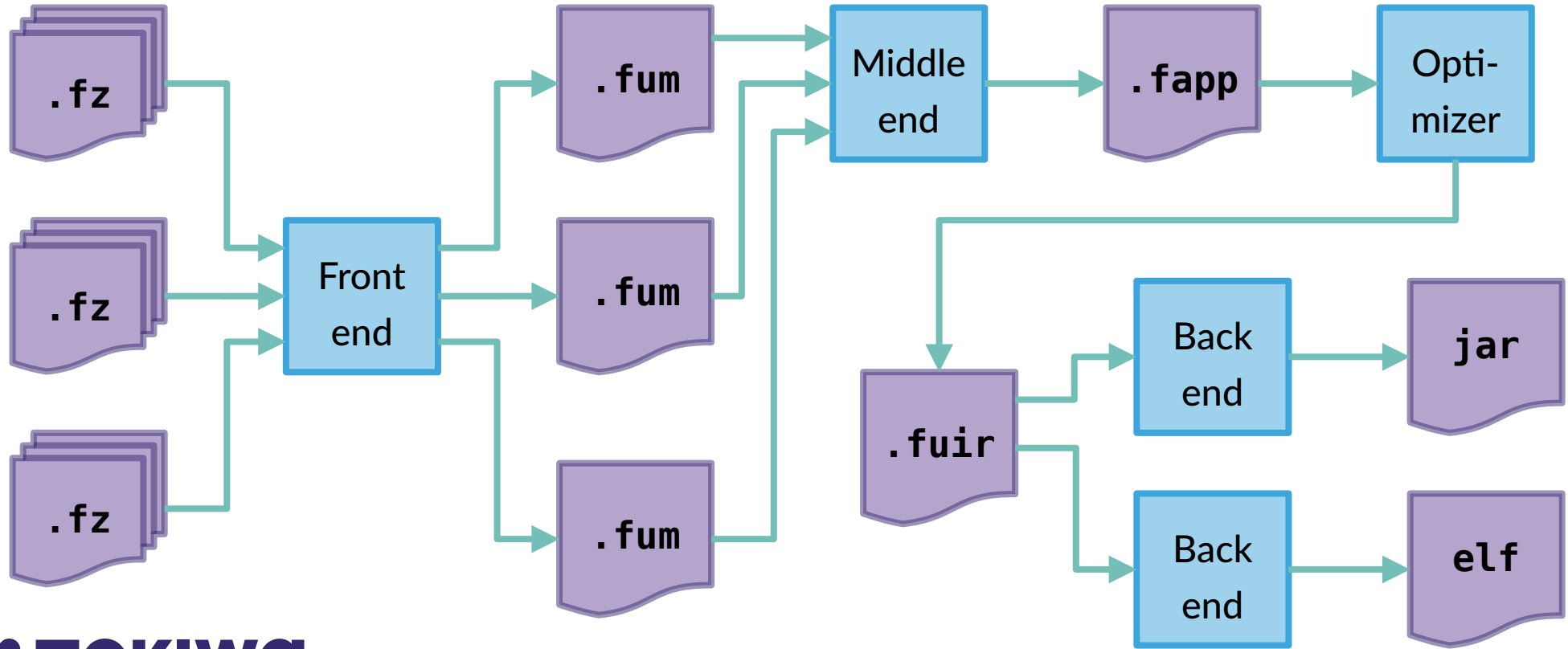
Fuzion Toolchain: Planned Design



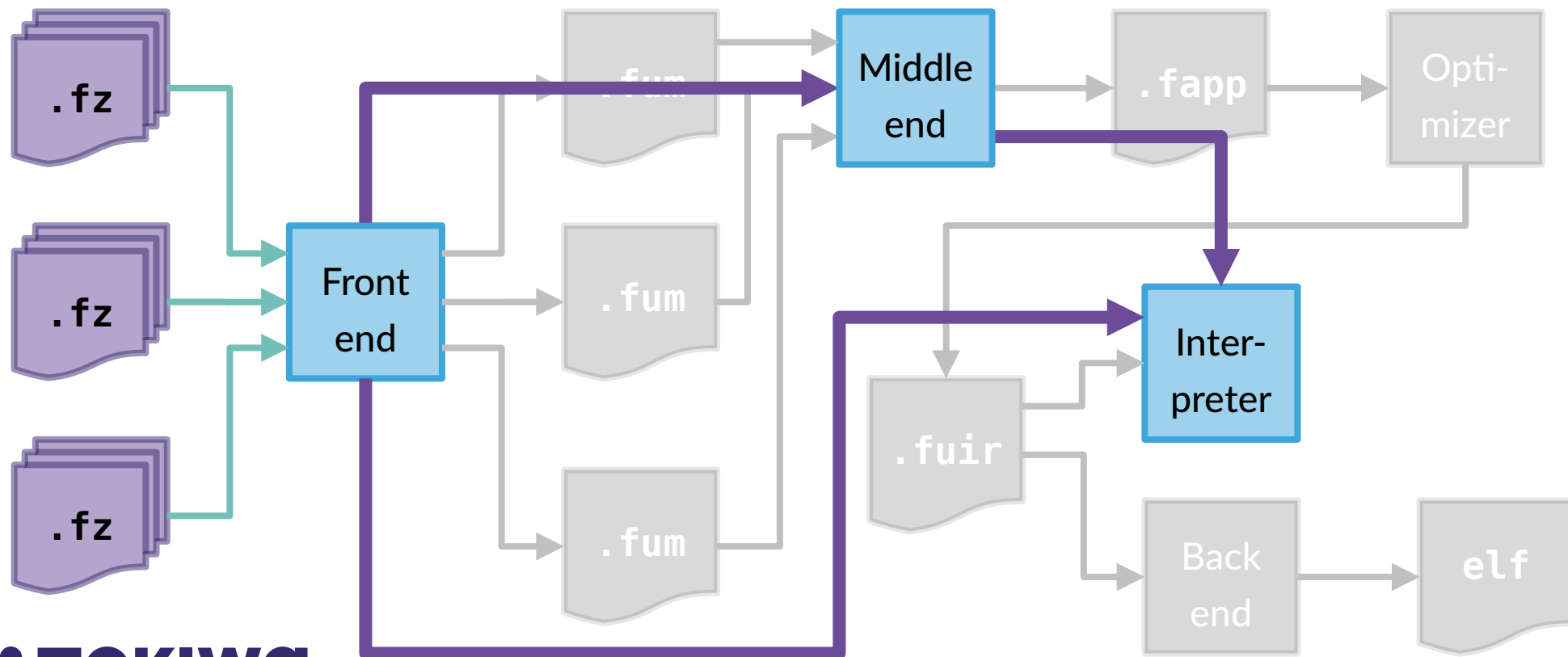
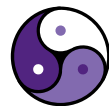
Fuzion Toolchain: Planned Design



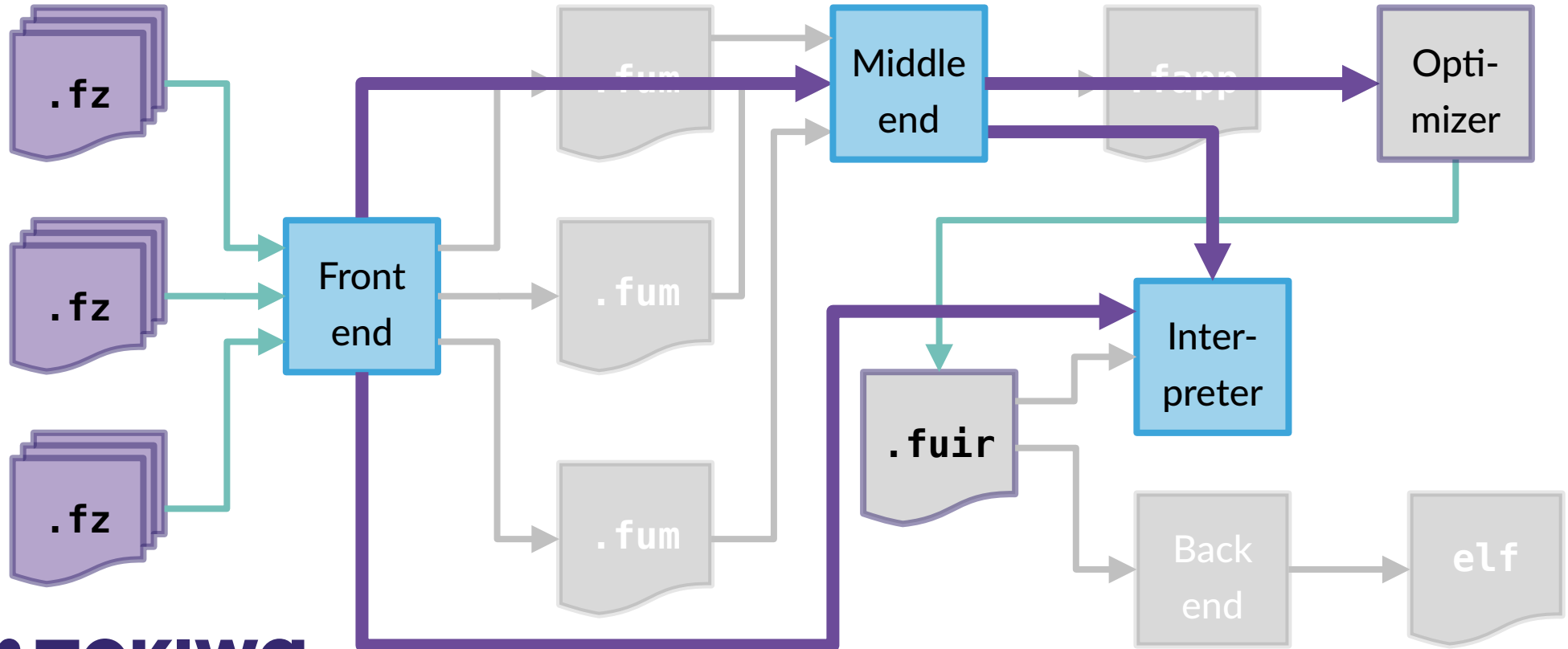
Fuzion Toolchain: Planned Design



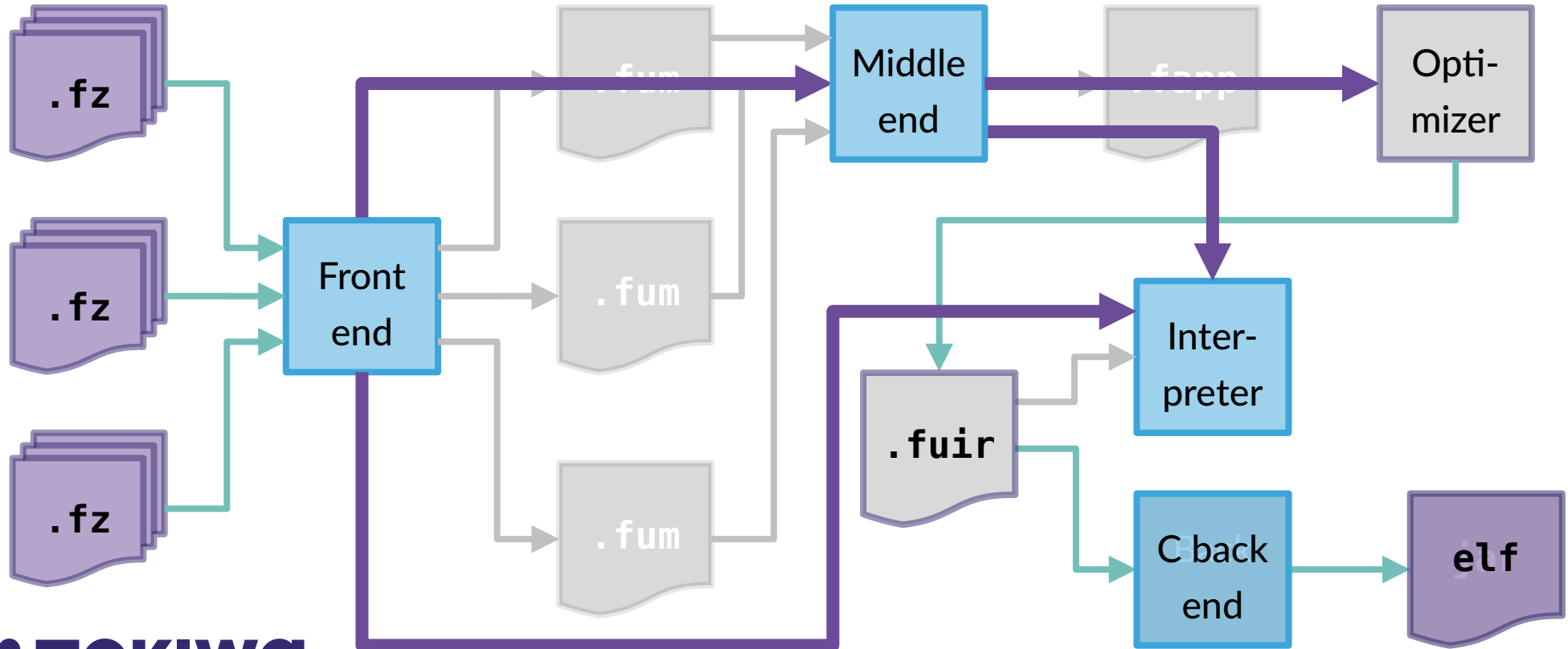
Fuzion Toolchain: State Jan'21



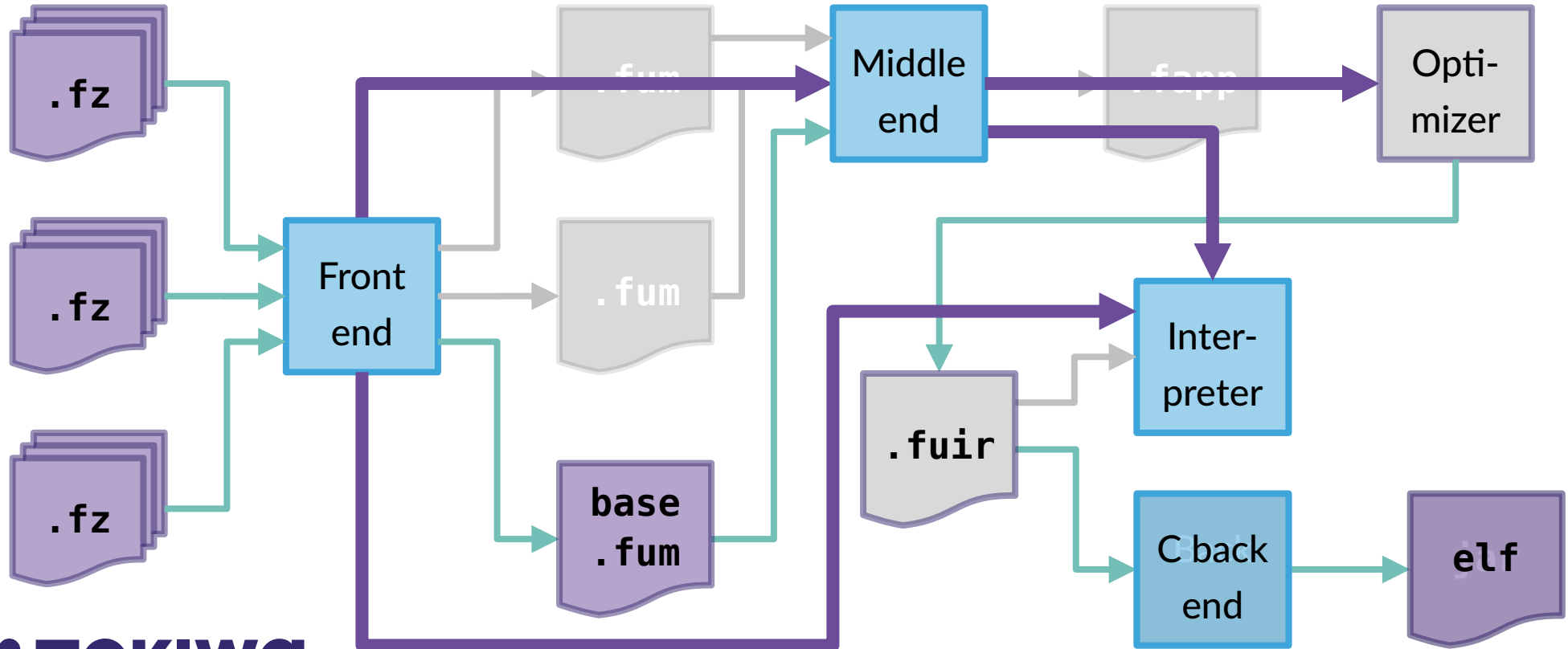
Fuzion Toolchain: Changes 2021



Fuzion Toolchain: Changes 2021



Fuzion Toolchain: State Jan'22



Fuzion Module Files



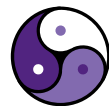
Binary file format

Fuzion Module Files



```
fridi@zen: ~/fuzion/work

001550: 00 00 01 6e 00 00 00 00 00 00 00 00 0d 01 f6 .....n.....
001560: 00 00 12 99 ff ff ff fe 00 00 08 57 00 00 00 00 .....W....
001570: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
001580: 00 00 00 00 0e 00 00 00 06 4d 6f 6e 6f 69 64 00 .....Monoid.
001590: 00 00 00 00 00 00 00 00 0d 07 65 00 00 00 00 00 .....e.....
0015a0: 00 00 01 00 00 00 00 00 01 54 ff ff ff fe 00 00 00 .....T.....
0015b0: 2b 00 00 00 01 00 00 00 11 84 00 0d 07 65 00 00 +.....e..
0015c0: 1d d5 ff ff ff fe 00 00 00 00 2b 00 00 00 00 00 .....+.....
0015d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0015e0: 07 f3 01 00 00 00 08 23 5e 4d 6f 6e 6f 69 64 00 .....#^Monoid.
0015f0: 00 00 00 00 00 00 0d 07 65 00 00 15 84 ff .....e.....
001600: ff ff fc 00 00 00 00 00 00 00 00 00 00 00 00 .....
001610: 00 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 .....
001620: 65 00 00 00 00 00 00 00 00 00 0d 0a 34 00 00 15 e.....4...
001630: 84 ff ff ff ff 00 00 15 a4 00 00 00 01 00 00 00 .....
001640: 11 84 00 0d 0a 34 00 00 1d d5 ff ff ff fe 00 00 .....4.....
001650: 00 2b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 +.....
001660: 00 00 00 00 00 4c 01 00 00 00 0a 23 5e 4d 6f 6e .....L.....#^Mon
001670: 6f 69 64 2e 65 00 00 00 00 00 00 00 00 0d 0a oid.e.....
001680: 34 00 00 16 1b 00 00 00 01 00 00 15 84 00 ff ff 4.....
001690: ff fe 00 00 16 31 ff ff ff fd 00 00 00 00 00 00 .....1.....
0016a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0016b0: 00 00 00 00 00 02 65 71 00 00 00 02 00 00 00 00 .....eq.....
0016c0: 00 00 0d 0a d1 00 00 15 84 ff ff ff fe 00 00 07 .....
0016d0: dc 00 00 00 01 00 00 00 11 84 00 0d 0a d1 00 00 .....
0016e0: 1d d5 ff ff ff fe 00 00 00 2b 00 00 00 00 00 00 .....+.....
0016f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 4f 85 00 .....0...
001700: 0d 0a df 04 00 00 18 75 ff ff ff fe 00 00 16 85 .....u.....
001710: 85 00 0d 0a e8 04 00 00 17 51 ff ff ff fe 00 00 .....Q.....
001720: 16 31 02 85 00 0d 0a eb 04 00 00 17 c5 ff ff ff .1.....
001730: fe 00 00 16 31 02 84 00 0d 0a df 00 00 18 b5 ff ....1.....
001740: ff ff fe 00 00 07 dc 05 01 00 00 18 39 00 00 01 .....9...
001750: 64 01 00 00 00 01 61 00 00 00 00 00 00 00 00 00 d.....a.....
001760: 0d 0a d4 00 00 16 b2 ff ff ff fe 00 00 16 31 00 .....1.....
001770: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
001780: 00 00 00 00 00 00 3e 01 00 00 00 0d 23 5e 4d 6f .....>.....#^Mo
001790: 6e 6f 69 64 2e 65 71 2e 61 00 00 00 00 00 00 00 noid.eq.a.....
0017a0: 00 00 0d 0a d4 00 00 17 51 ff ff ff fc 00 00 00 .....Q.....
0017b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0017c0: 00 00 00 00 00 01 00 00 00 01 62 00 00 00 00 00 .....b.....
Field, Monoid.eq.b
```



Fuzion Module Files

Binary file format

- **fmap()** ped to memory
- references using offsets, not names
- contains **features, types, expressions, source code**
- features are **routine, field, intrinsic, abstract** or **choice**
- types are **feature types** or **type parameters**
- 10 kinds of expressions: **call, match, const, assign, pop, ...**



Fuzion Module Files

Impact:

- faster compilation
- module-based correctness analysis
- define feature visibility
- container for exchange of library code



Other Tools: FZJava

Create Fuzion interface to Java module



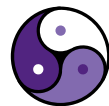
```
java.lang.System.out.println "Hello Java    !"
```



Other Tools: Language Server

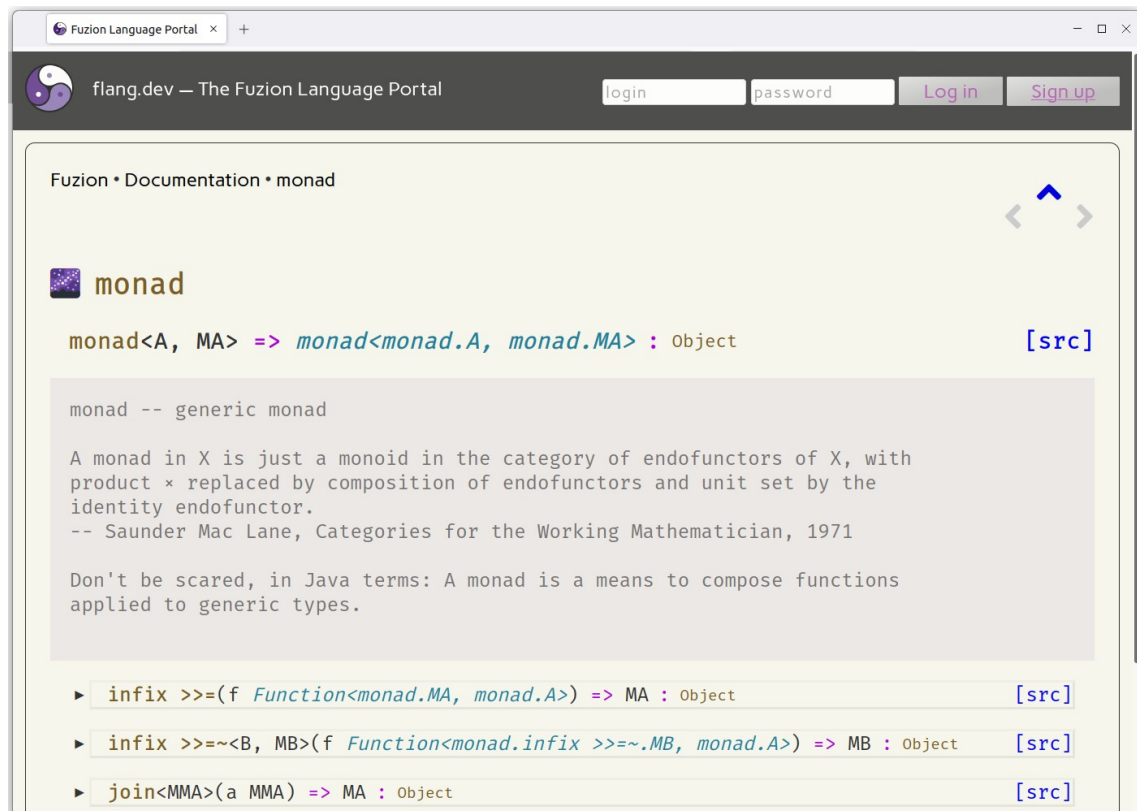
Support for IDEs and editors (vim, emacs)

- completion
- signature help
- documentation
- ...



Other Tools: FuzionDoc tool

Extract documentation from Fuzion source code

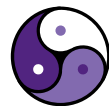




Fuzion: Next Steps

Development Plan

- intermediate files: .fum, .fapp, .fuir
- simple analysis tools: field init, immutability
- C back-end: GC, floats, etc.
 - interfacing C library code
- Standard Library
- Modeling I/O, thread communication and immutability
 - using automatic monadic lifting?



Conclusion

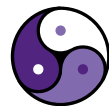
We are running a Marathon here

- we have not made the first 10km yet
- 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



Conclusion

We are running a Marathon here

- we have not made the first 10km yet
- 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