# Visi: Cultured & Distributed

@dpp at Strange Loop Emerging Languages

*September 23rd, 2012*

# @dpp

- Wrote spreadsheets
  - Mesa -- First real-time spreadsheet
  - Integer -- First distributed, browser-based spreadsheet
- Founded Lift
- Crazy passionate, lawyer trained tech dude

# Overview

- Tension: Compatibility vs. Innovation
- Practical
- Philosophical

# Visi

- Spreadsheet language, re-imagined
- An exploration of culture, technology, and PLT
- Something in my veins

# But Why?

- Spreadsheet is the most common programming language in the world
- It's massively broken (50%-80% of all spreadsheets have bugs)
- Economics of Software Broken

# Language Demo

- It's a spreadsheet
- No, it's Haskell
- Wait... maybe it's something else

# More Language Stuff

- Structural typing like OCaml
- Single level subtyping
- All (almost) immutable data structures
- Open data structures and classes
- No modification of Type Constructors

# Data Structures

- `struct Bool2 = True | False`

# Data Structures

- `struct Dog(String)`

# Data Structures

- `struct Cat(name: String)`

# Data Structures

- ```
  struct Thing = This(String) |
                 That(when: Date)
  ```
  ```

# Data Structures

- ```
  struct Person(String, age: Int) =
    Kid() |
    Parent(kids: [Person])
  ```
  ```

# Pattern Matching

- Type Constructor/Extractor
- `dogsName Dog(name) = name`
- `kidsName Kid(name, _) = name`

# Pattern Matching

- Nominal (anything with the property `name`)
- `name (name => theName) = theName`
  `name2 (name =>) = name`

# Pattern Matching

- Positional (anything in the first position of a product type with a single param constructor)

- ```
  something (thing) = thing
  // require a String
  someString (str: String) = str
  ```
  ```

# Pattern Matching

- Tests `name` is “fred”
- Non-exhaustive means `Box`
- `fredsAge (name == “fred”,`
  `age: Int =>) = age // Box Int`

# Pattern Matching

- Tests `name` is “fred” for a `Person`
- Non-exhaustive means `Box`
- `fredsAge2 Person(name == “fred”,`
  `age =>) = age // Box Int`

# Functions

- Structural Typing
- ```
  anyAge n = n.age
  anyAge2 = #age // curried
  ```
  ```

# Define Methods

- Methods on a type
- 
```
struct Foo(age: Int)
  methods
    old? = self.age > 85
    addToAge n = self.age + n

testOld n = n.old?
testOld2 = #old?
```

# Updators

- How to create a new instance?

- ```
  kid = Kid “Daniel” 7
  birthday = kid.=age 8
  nextYear n =
    curAge = n.age
    n.=age (curAge + 1)
  makeOld = #=age 86
  ```
  ```

# Updators

- Via function
- 
```
kid = Kid “Daniel” 7

nextYear kid = kid.>age (+ 1)
nextYear2 = #>age (+ 1)

nextYear2 kid // Kid “Daniel” 8
```

# Precursors

- Mixins with attitude
- precursor TestAge

```
  data
    old? = olderThan 85
  methods
    olderThan2 n = self.age > n

enhance Person with TestAge
```

# Sources & Sinks

- Accumulation
- ?age // input the age

```
allAges = age:allAges // collect
ageCnt = length allAges

"age count" = ageCnt
"average" = (sum allAges)/ageCnt
```

# References

- Clojure-like
- Computation delineation points
- No syntax or semantics, yet (waves hands)

# More unfinished stuff

- Modules/packages/dependency mgt
- Visibility
- Code signing/execution rights
- Library mode (access to types and mutability and stuff)
- Unit Types/Type Algebra

# Social Dynamic

- GitHub: where the code and libraries live
- Smalltalk: hack the platform & share your hacks
- HyperCard/Excel: never start with a blank slate

# Technology ⇔ Human

- Interactive: try it, you'll like it (or you can undo)
- Separation between model, computation, data, execution
- Core IDE/Interactivity Support & Incremental Compilation

# Right Thing == Easy Thing

- Docs
  - JavaDocs: yeah, we can document that
  - Visi: Prose around Logic, models are Markdown documents
- Tests
  - Rails & Tests: a core social dynamic
  - Visi: language docs are the language tests
- Types: Looks and Feels dynamic, problems caught early

# The Bridge

- Events flow and Visi programs handle them
- The locus of the computation & transportation is invisible
  - Data BSON serializable and immutable: homage to Erlang
  - Delineated side effects (sinks, references): retries/migration simple
- Like query optimization, it can only get better

# Logic Primacy

- Logic trumps plumbing
- App author distinct from Library author
  - Immutable vs. locally mutable
  - Execution locus hints/constraints
  - Access to typed lambda calculus/macros
- Inform but don't control the locus and order of logic application

# Lightweight, Excellent Types

- Unit Types w/Type Algebra
  - 4 inches * 6 feet = 288 (inches * inches)
  - $4 + 6 inches = unit error
  - $4 / 2 inches = $2/inch
  - $4 + €6 = maybe a result
- Generally Invisible
- Hardest Problem

# End

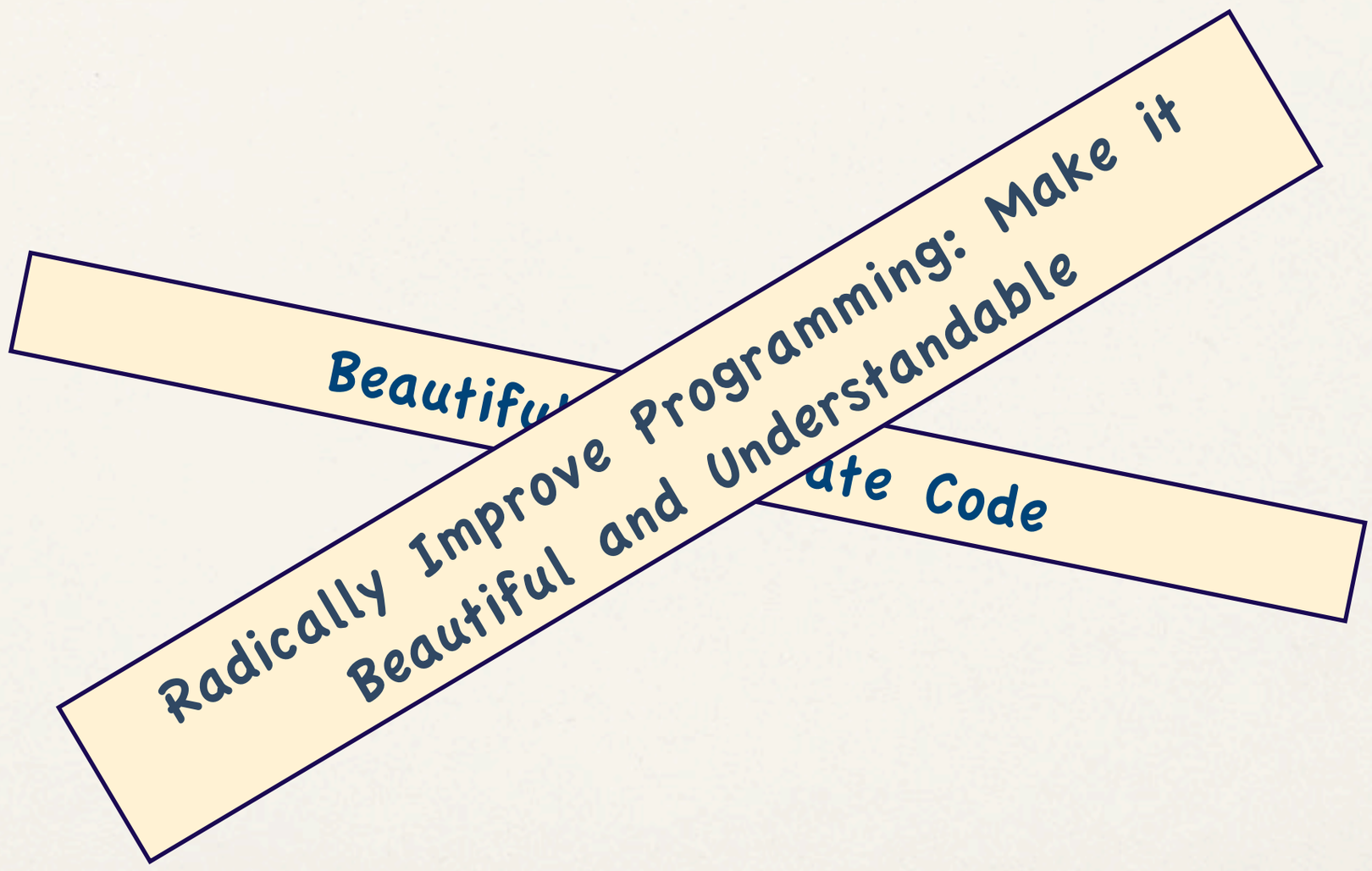
- http://www.visi.io/
- https://github.com/visi-lang/
- Beer

# Visi

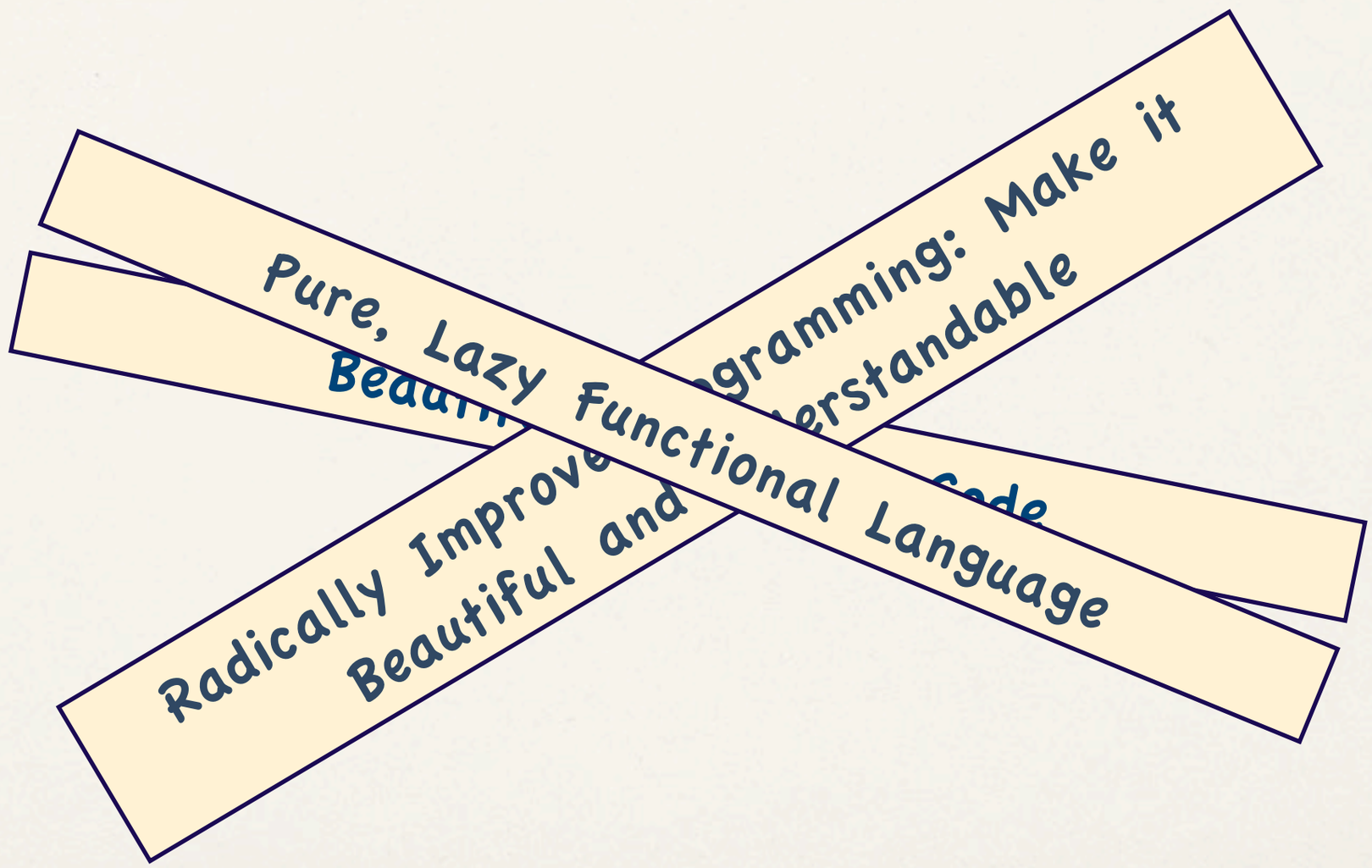
# Visi

Beautiful & Literate Code

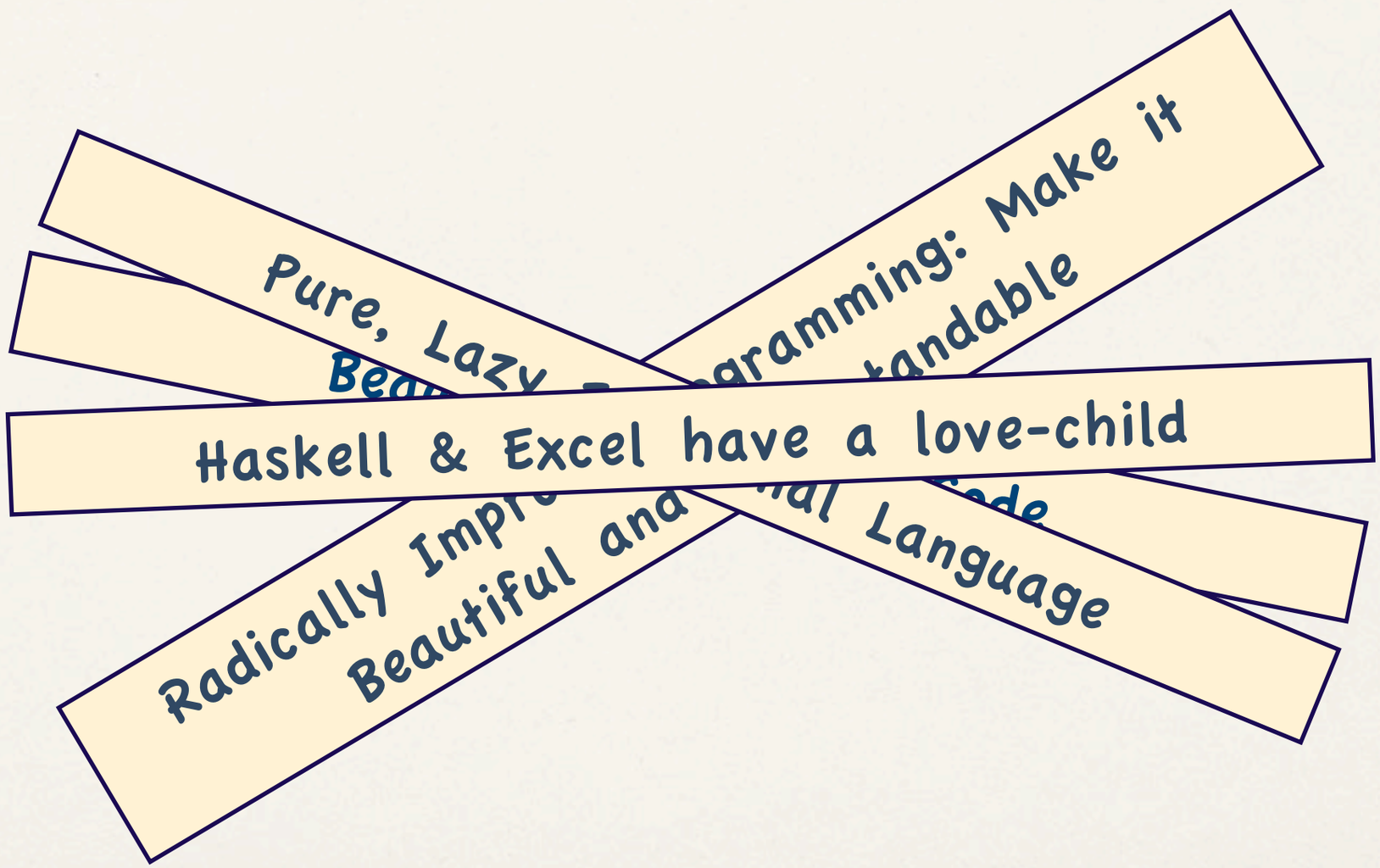
# Visi


Beautifu
ate Code
Radically Improve Programming: Make it
Beautiful and Understandable

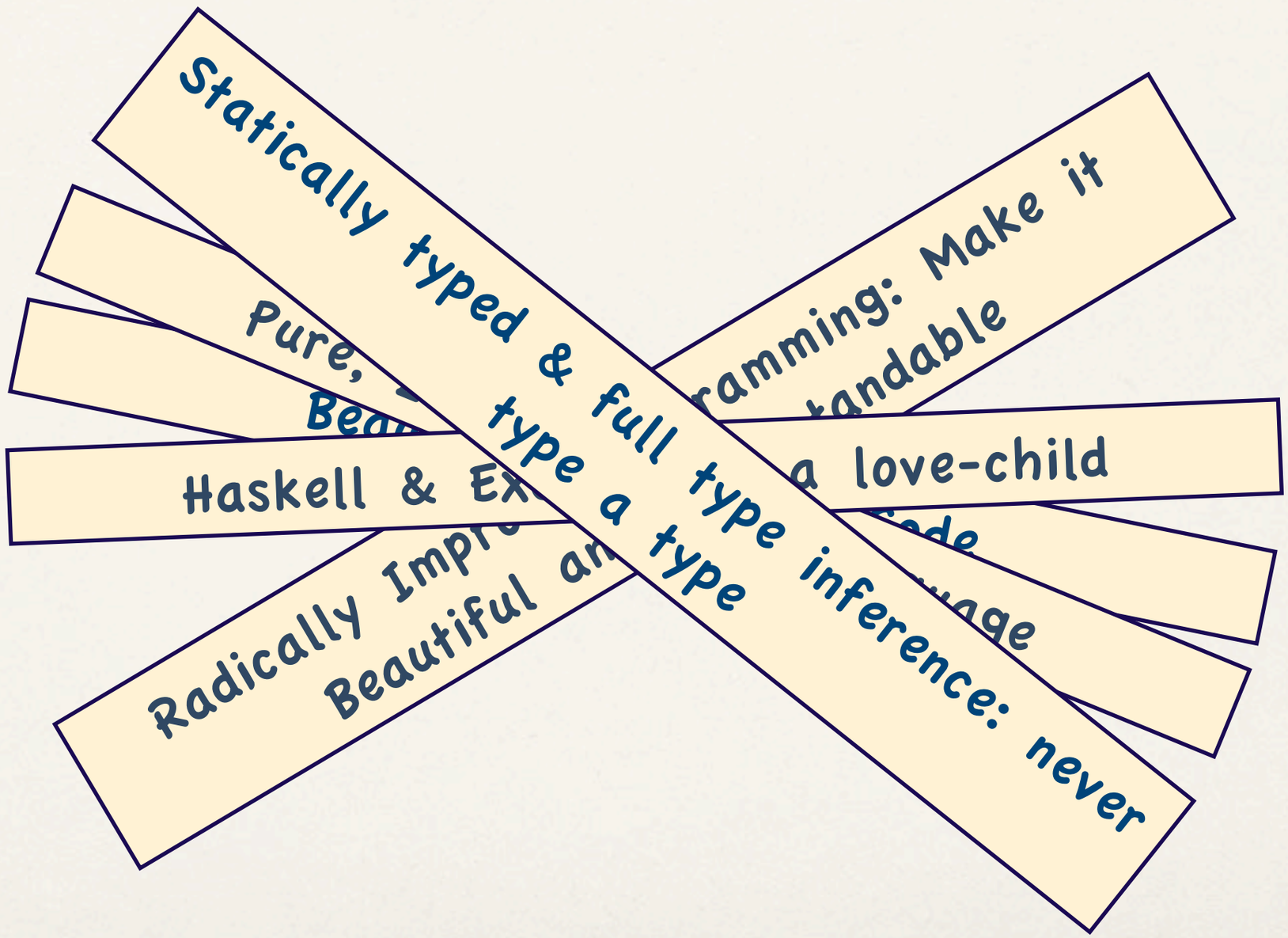
# Visi


Pure, Lazy Functional Language
Radically Improve
Beautiful and
Make it
Understandable

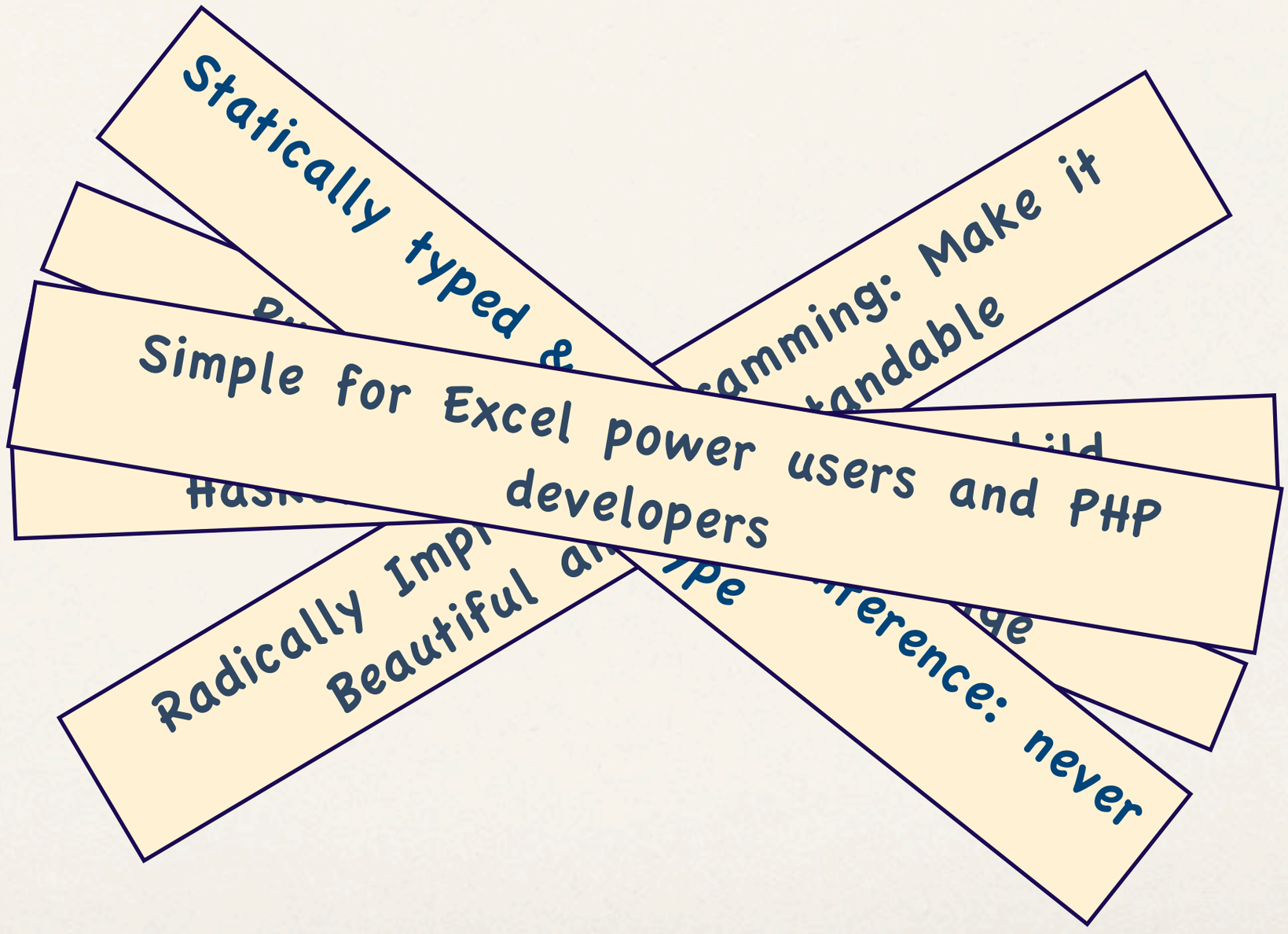
# Visi


Pure, Lazy
Make it
Haskell & Excel have a love-child
Radically
Beautiful and
Language

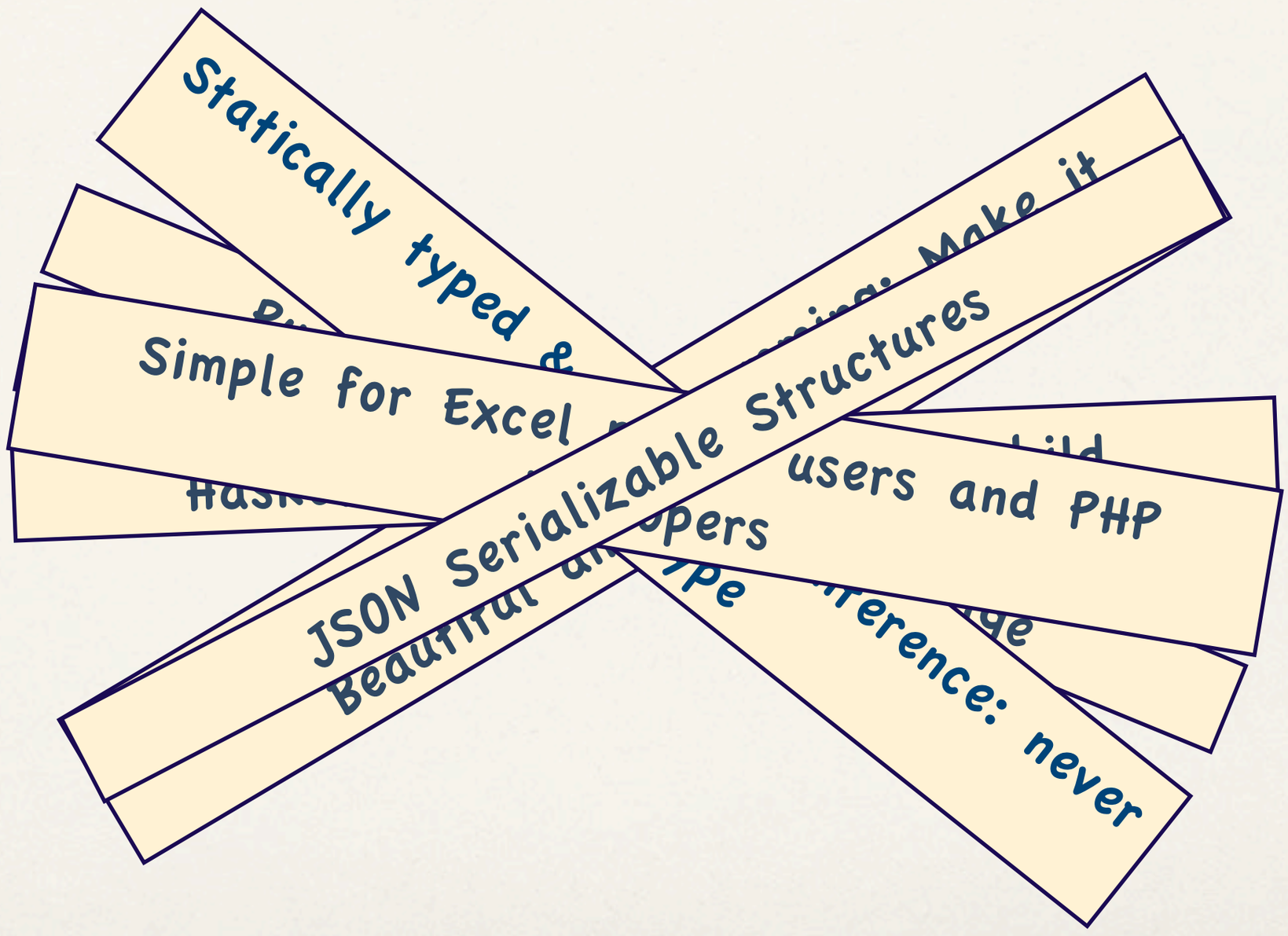
# Visi


Statically typed & full type inference: never type a type
Pure,
Bea
Haskell & Ex
Radically Impr
Beautiful an
ramming: Make it
tandable
a love-child
ode
uage

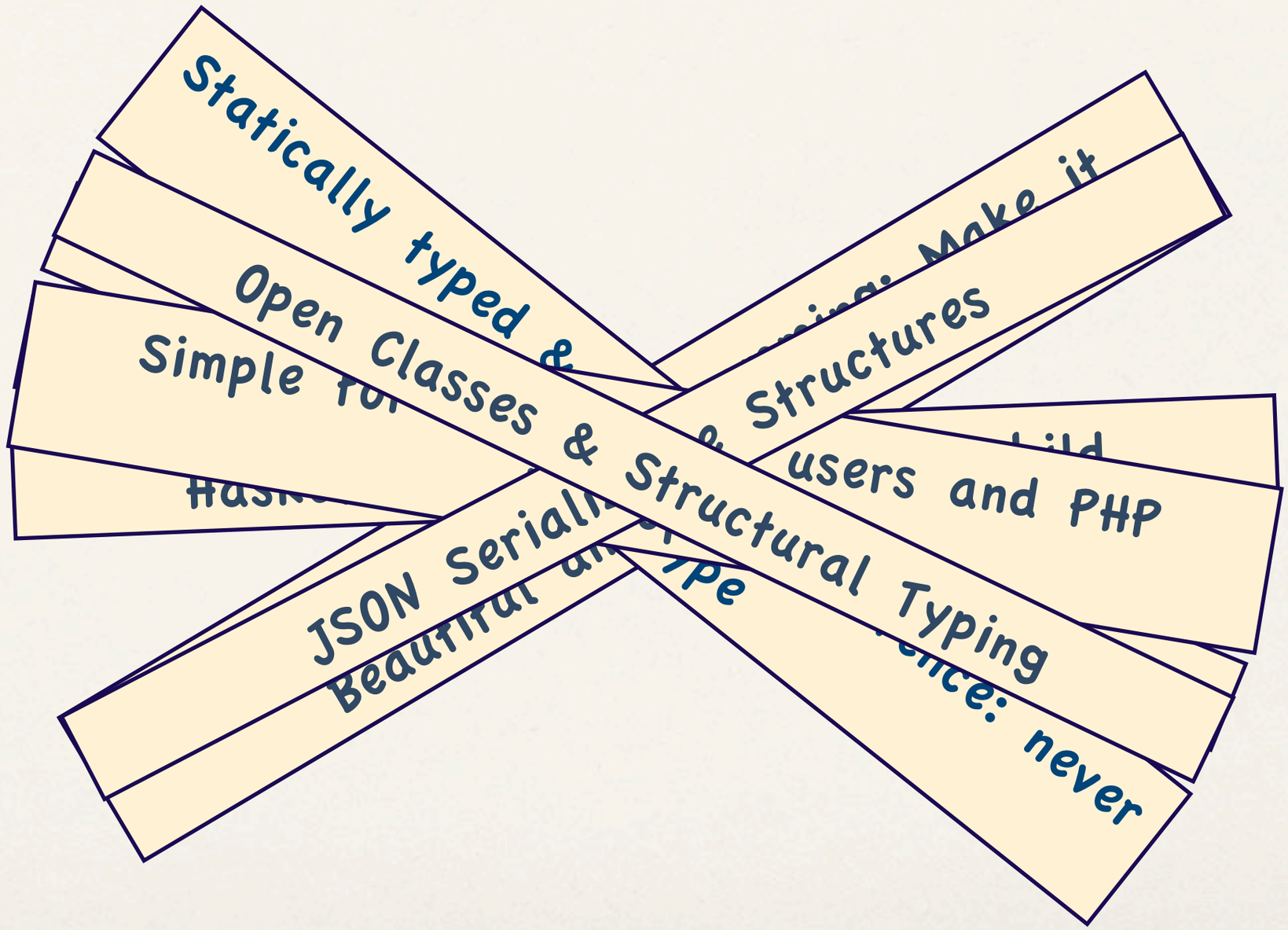
# Visi


Statically typed &
Make it
Simple for Excel power users and PHP developers
Radically Impl
Beautiful a
never

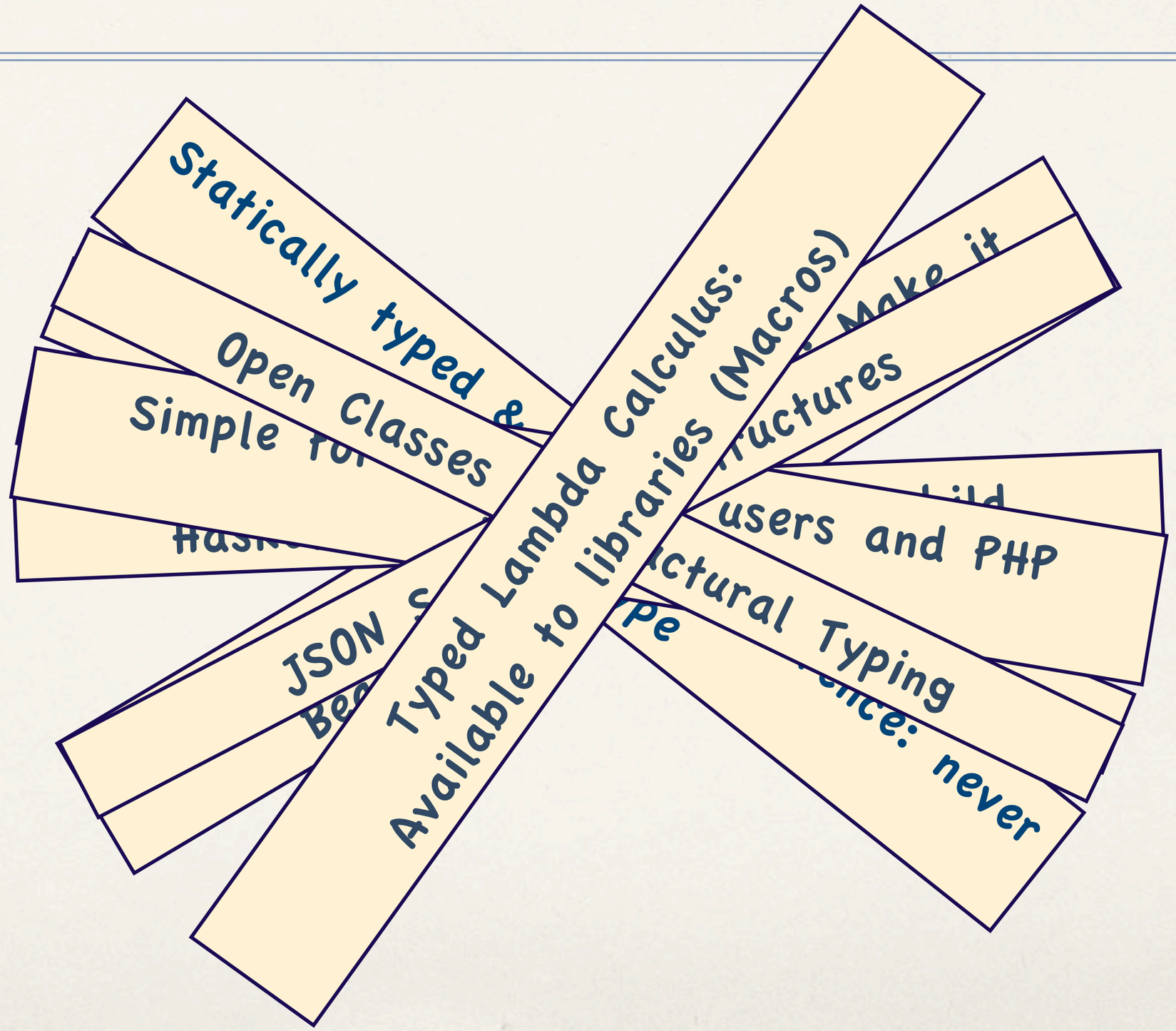
# Visi


Statically typed &
Simple for Excel
JSON Serializable Structures
Beautiful
users and PHP
never

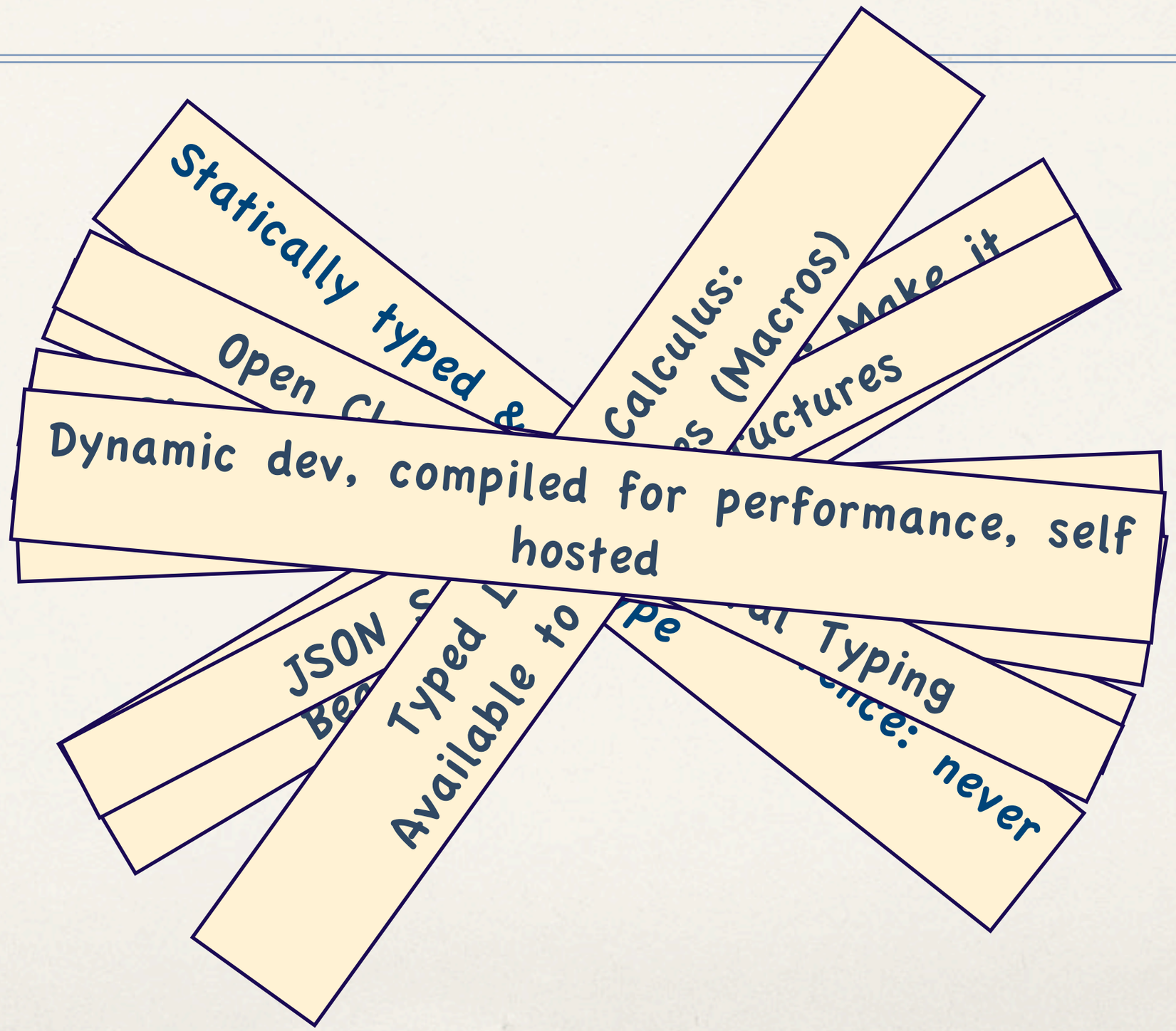
# Visi


Statically typed &
Open Classes & Structural Typing
Structures
users and PHP
JSON Serializ
Beautiful
never

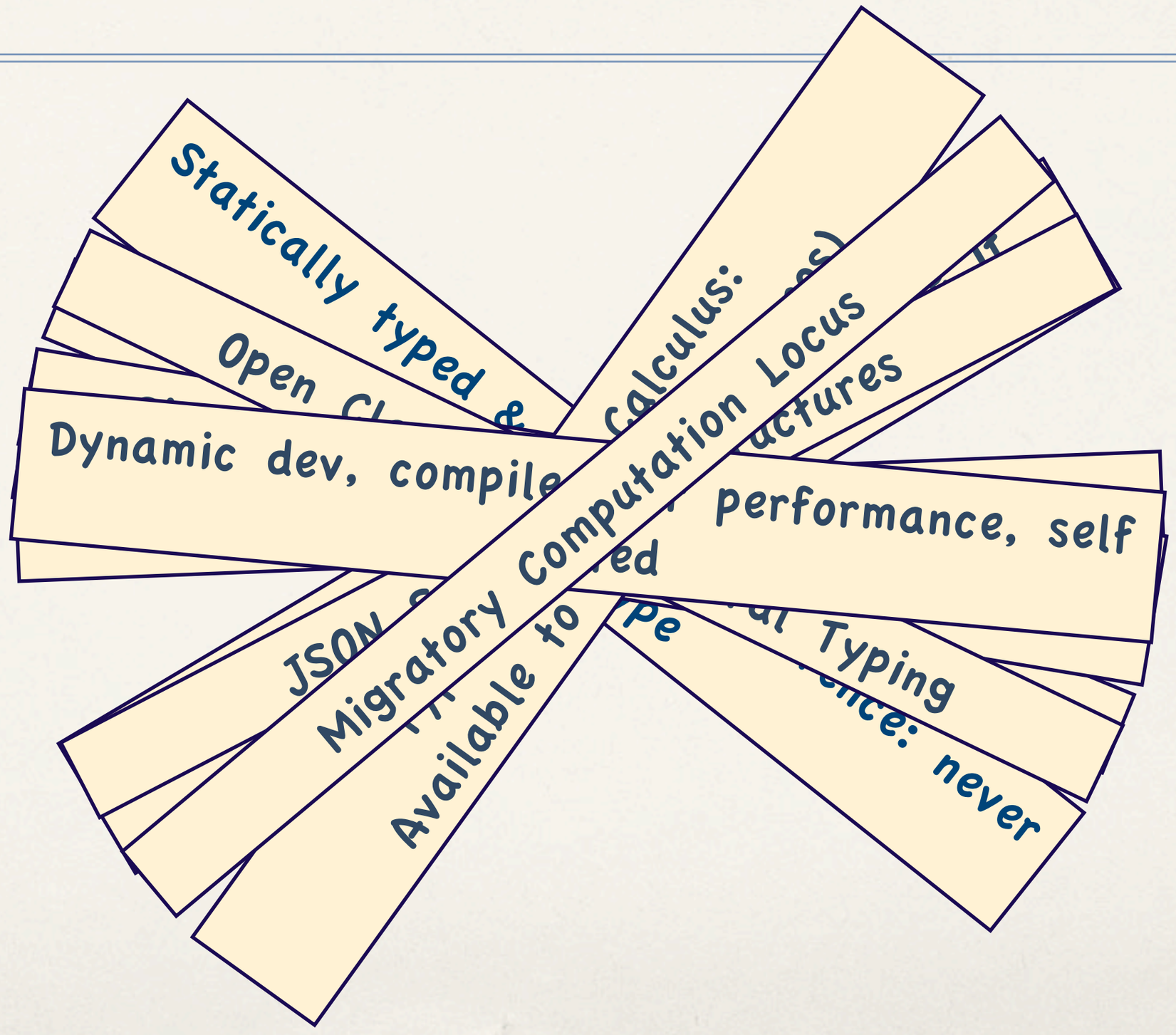
# Visi


Statically typed &
Open Classes
Simple for
Typed Lambda Calculus:
Available to libraries (Macros)
Make it
structures
users and PHP
ctural Typing
never

# Visi


Statically typed &
Open Cl
Calculus:
(Macros)
Make it
ructures
Dynamic dev, compiled for performance, self hosted
JSON S
Bea
Typed L
Available to
Pe
al Typing
ence: never

# Visi


Statically typed &
Open Cl
Dynamic dev, compile
JSON
Migratory Computation
Available to
calculus:
Locus
actures
Performance, self
Typing
never

# Visi