

# Progress report in the Pen programming language

@raviqqe

# Progress report

# keys and values built-in functions

- The `keys` function returns keys of a map.
- The `values` function returns values of a map.
- An order of keys and values are consistent among different instances of equivalent maps.
  - i.e. `xs == ys` where `xs` and `ys` are maps
- Note that key orders depends on their implementations because maps in Pen are hash maps.

```
xs = {string: number "foo": 1, "bar": 2}
```

```
keys(xs) == [string "bar", "foo"]
```

```
values(xs) == [number 2, 1]
```

# Parallel list comprehension

- Natural extension to list comprehension for `zip`-ish computation
- Not related to parallel computation

```
[number x() + y() for x, y in xs, ys]
```

## In Haskell

- With [the ParallelListComp](#) language extension in GHC

```
[x + y | x <- xs | y <- ys]
```

# Use cases

## Zipping

- Example: JSON serialization
- Removal of list comprehension with map iteratees

```
String'Join(  
  [string  
    serializeString(key()) + ":" + serializeValue(value())  
    for key, value in keys(map), values(map)  
  ],  
  ",", "  
)
```

# Use cases

## Enumeration of list elements

- Example: SQL query build with placeholders
- Lists in Pen are lazy.

```
" where "  
+ String'Join(  
  [string  
    field() + " = $" + Number'String(index())  
    for field, index in whereFields, Number'Sequence(Number'Infinity())  
  ],  
  " and ",  
)
```

# Future work

- More little language features
  - `sort` built-in function
- Code generator
  - For meta-programming
- Language server

# Summary

- Now, Pen has parallel list comprehension for `zip`-ish composition of lists.