Swiss Knife

(JS Prototype alike tool for Go)

@pitakill
Leopoldo Caballero

About

From JS to Go (my path)

Static types

Generics

Static types

Valid JS code:

```
const paragraph = 'Hola Mundo!'

let regex = /[A-Z]/g
let found = paragraph.match(regex)

console.log(found)
// ['H', 'M']

regex = /\d/g
found = paragraph.match(regex)

console.log(found)
// [] or null
```

Static types

Valid JS code:

```
const matchs = (string, regex) => string.match(regex)

const paragraph = 'Hola Mundo! 8'
let regex = /[A-Z]/g

const lowercased = matchs(paragraph, regex).map(s => s.toLowerCase())
console.log(lowercased)
// ['h', 'm']

regex = /\d/g
const squares = matchs(paragraph, regex).map(d => d * d)
console.log(squares)
// [] or NaN or null or Breaks the execution
```

Static types

Valid Go code:

```go
// ...

func main() {
    p := "Hola Mundo!"

    r := regexp.MustCompile(`[A-Z]`)
    matched := r.FindAllString(p, -1)

    lowercased := make([]string, 0, len(matched))
    for _, e := range matched {
        lowercased = append(lowercased, strings.ToLower(e))
    }

    fmt.Println(lowercased)
    // [h, m]

    r = regexp.MustCompile(`\d`)
    matched = r.FindAllString(p, -1)

    squares := make([]int, 0, len(matched))
    for _, e := range matched {
        number, _ := strconv.Atoi(e)
        squares = append(squares, number*number)
    }

    fmt.Println(squares)
    // [] slice of int empty
}
```

Static Types

Trade-offs

Verbose code

Generics (mainly)

Generics

Valid JS code:

```
// ...

// matchs is an array of strings (hopefully)

const lowercased = matchs(paragraph, regex).map(s => s.toLowerCase())
// lowercased is an array of strings (hopefully)

// ...

const squares = matchs(paragraph, regex).map(d => d * d)
// squares is an array of numbers (hopefully)

// Array.map returns an array of any type (Generics)

// ...
```

Generics

Valid Go code:

```go
// ...

// matched is an Slice of strings

lowercased := make([]string, 0, len(matched))
for _, e := range matched {
    lowercased = append(lowercased, strings.ToLower(e))
}

// ...

squares := make([]int, 0, len(matched))
for _, e := range matched {
    number, _ := strconv.Atoi(e)
    squares = append(squares, number*number)
}

// We don't have a full generics solution in Go
// append is a generic function because works with any type

// ...
```

Generics

Trade-offs

Verbose code

Swiss knife (slice) (JS Prototype alike on Go)

```go
// ...

// matched is an Slice of strings

lowercased := make([]string, 0, len(matched))
for _, e := range matched {
    lowercased = append(lowercased, strings.ToLower(e))
}

// ...

squares := make([]int, 0, len(matched))
for _, e := range matched {
    number, _ := strconv.Atoi(e)
    squares = append(squares, number*number)
}

// We don't have a full generics solution in Go
// append is a generic function because works with any type

// ...
```

```go
// ...

// matched is an Slice of strings

lowercased, err := slice.StringMap(matched, strings.ToLower)
if err != nil {
  // Handle error accordingly
}

// http://bit.ly/swiss-knife-example

// ...

squares, err := slice.Map(matched, func(e string) int {
    number, _ := strconv.Atoi(e)
    return number*number
})
if err != nil {
  // Handle error accordingly
}
// Currently not working
// PR's always welcome :-)

// ...
```

Thank you

https://github.com/pitakill/swiss-knife

@pitakill
Leopoldo Caballero