Dependent Types

and other ideas for guaranteeing correctness with types

Radek Pietruszewski

@radexp • radex.io

Static typing is nice

Expressing guarantees

Partial functions

```
let xs = [1, 2, 3]
let xs = [1, 2, 3]

xs[0]  // => 1
```

```
let xs = [1, 2, 3]

xs[0]  // => 1
xs[4]  // => crash
xs[-1] // => crash
```

```
let xs = [1, 2, 3]
```

$$xs[n] = \begin{cases} \texttt{some value}; & n \in \langle 0,2 \rangle \\ \texttt{¯\textbackslash\_(ツ)\_/¯}; & n<0 \lor n>2 \end{cases}$$

```
subscript (index: Int) -> Element
```

```
-a lot, ..., -1, 0, 1, 2, 3, 4, a lot
```

```
subscript (index: Int) -> Element
```

```
-a lot, ..., -1, 0, 1, 2, 3, 4, a lot
```

```
subscript (index: Int) -> Element
```

Partial function!

No compile time check

No compile time check

😕

Prefer
total functions
to partial functions

Optionals

Foo* vs **Foo** vs **Foo?**

Enums

```swift
func move(direction: String)
func move(direction: String)


move("up")
move("down")
move("wat") // undefined
```

```swift
enum Direction {
    case Up, Down, Left, Right
}

func move(direction: Direction)
```

```
enum SuggestionViewModel {
  case Header(String)
  case Suggestion(Suggestion)
}
```

Dependent Types

```swift
struct User {
    var loggedIn: Bool
    ...
}
```

```swift
    struct User {
        var loggedIn: Bool
        ...
    }


/// `user` must be logged in!
func doSomethingImportant(user: User)
```

```swift
/// `user` must be logged in!
func doSomethingImportant(user: User)
```

```swift
User(loggedIn: false)
```

```swift
/// `user` must be logged in!
func doSomethingImportant(user: User)
```

```swift
User(loggedIn: false)
```

```
User<loggedIn=true>
```

Validated

github.com/Ben-G/Validated

```swift
struct LoggedInValidator: Validator {
  static func validate(value: User) -> Bool {
    return value.loggedIn
  }
}
```

```swift
struct LoggedInValidator: Validator {
  static func validate(value: User) -> Bool {
    return value.loggedIn
  }
}
```

```
Validated<User, LoggedInValidator>
```

```
Validated<User, LoggedInValidator>
```

```
User<loggedIn=true>
```

```
typealias LoggedInUser =
    Validated<User, LoggedInValidator>
```

```
let rawUser = User(loggedIn: false)
let rawUser = User(loggedIn: false)

let loggedInUser = LoggedInUser(rawUser)
```

```swift
func doSomethingImportant(user: LoggedInUser)
func doSomethingImportant(user: LoggedInUser)
```

Keep your functions total

@radexp • radex.io

bit.do/partial-functions

github.com/Ben-G/Validated