Обо мне

- Семёнов Дмитрий, ведущий разработчик, Модульбанк.
- Профессионально занимаюсь разработкой на .NET с 2012 года.
- Увлёкся функциональным программированием в 2016 году.

F# и предсказуемый код

Нужен ли вам F#? Стоит ли прилагать усилия к изучению нового языка?

- Зачем, если есть C#?
- F# имеет свои преимущества, но они несущественные и и оно того не стоит,
- Я не использую F# в работе, но его изучение помогло мне программировать лучше!

> A language that doesn't affect the way you think about programing, is not worth knowing - Alan Perlis

Кратко об языке F#

- Разработан Microsoft Research, 2010
- Открытый исходный код на GitHub
- Возможность разработки на разных платформах (Windows, Linix, MacOS)
    - Работает с VS Code (и др. редакторы)
- Очень активное и дружелюбное сообщество
    - Начни с fsharp.com
    - Или fsharpforfunandprofit.com

Кратко об языке F#

- Различия с C#
  - более краткий синтаксис,
  - разное поведение по-умолчанию,
  - **другая философия**;
- Только в F#
  - автоматический вывод типов,
  - алгебраические типы данных,
  - функциональная парадигма;

Другая философия

- C# исторически вышел из языка C,
- F# вышел из ML, языка *метапрограммирования*, созданного для доказательства *корректности* программ.

Цель: предсказуемый код

- Можете ли вы понять код, просто посмотрев на него?
- Вам не надо копаться в других частях кодовой базы!

Пример 1

```
function DoSomething (arg) { arg = "world"; }

var x = 1;
DoSomething(x);

var y = "hello " + x;
```

Такое никогда не произойдет в C#!

Пример 1 (F#)

```fsharp
let mutable x = 1

let doSomething (arg : byref<_>) = arg <- "world"

doSomething x |> ignore

"hello " + x
// error FS0001: This expression was expected to have type
// 'byref<string>' but here has type 'int'
// error FS0001: The type 'int' does not match the type
// 'string'
```

Пример 1 F#

```
let mutable x = 1

let doSomething (arg : byref<_>) = arg <- "world"

doSomething x |> ignore

"hello " + x
// error FS0001: This expression was expected to have type
// 'byref<string>' but here has type 'int'
// error FS0001: The type 'int' does not match the type
// 'string'
```

У переменных не должно быть возможности **поменять тип**

Пример 2

```
var cust1 = new Customer(99, "J Smith");
var cust2 = new Customer(99, "J Smith");

// true или false? Непонятно. Зависит от реализации.
cust1 == cust2;
```

Пример 2 F#

```
let cust1 = Customer(99, "J Smith")
let cust2 = Customer(99, "J Smith")

// true
cust1 == cust2;
```

Объекты с одинаковыми значениями должны быть **равными по-умолчанию**

Пример 3

```
var cust = new Customer(99, "J Smith");
var order = new Order(99, "J Smith");

// true или false? Опять таки зависит от реализации.
// Это потенциальный баг!
cust.Equals(order);
```

Пример 3 F#

```
let cust = Customer(99, "J Smith");
let order = Order(99, "J Smith");

// true или false? Опять таки зависит от реализации.
// Это потенциальный баг!
cust.Equals(order);
```

Сравнение объектов с **разными типами** приведет к ошибке компиляции

Пример 4

```
var cust = new Customer();

// что ожидается на выходе?
// Может быть `NullReferenceException`?
Console.WriteLine(cust.Address.Country);
```

Пример 4 F#

```
let cust = Customer()

match cust.Address with
| Some address ->
    Console.WriteLine address.Country // Some 'Russia или None

| None ->
    Console.WriteLine "Нет адреса"
```

Пример 4 F# продолжение

```
let cust = Customer()

match cust.Address with
| Some address ->
    Console.WriteLine address.Country // Some 'Russia или None

| None ->
    Console.WriteLine "Нет адреса"
```

Если адрес **необязательный**, то пусть это будет **явно**!

Объекты всегда должны быть инициированы **валидными** значениями. Если нет - ошибка компиляции

Пример 5

```
var cust = new Customer(99, "J Smith");

var processedCustomers = new HashSet<Customer>();
processedCustomers.Add(cust);

ProcessCustomer(cust);

// пользователь должен содержаться в множестве...
// ...но это неточно.
processedCustomers.Contains(cust);
```

Пример 5

```
var cust = new ImmutableCustomer(99, "J Smith");

var processedCustomers = new HashSet<ImmutableCustomer>();
processedCustomers.Add(cust);

// делаем что-то с ним и возвращаем измененного пользователя
var changedCustomer = ProcessCustomer(cust);

// теперь содержиться во множестве?
processedCustomers.Contains(cust);
```

Неизменяемость - очень полезное свойство, позволяющее больше доверять своему коду.

Как сделать язык предсказуемым:

- У переменных не должно быть возможности поменять их тип,
- Объекты с одинаковыми значениями должны быть равными по-умолчанию,
- Сравнение объектов с разными типами приведет к ошибке компиляции,
- Объекты всегда должны быть инициированы валидными значениями. Если нет - ошибка компиляции,
- Однажды созданные, объекты и коллекции должны быть **неизменяемы**.

Пример 6

```
var repo = new CustomerRepository();

var customer = repo.GetById(345);

// что ожидается на выходе?
Console.WriteLine(customer.Id);
```

Что если пользователь не найден? Вернется `null`? Или исключение?

Пример 6

```
var repo = new CustomerRepository();

var customerOrError = repo.GetByIdOrError(345);

if (customerOrError.IsCustomer)
{
    Console.WriteLine(customerOrError.Customer.Id);
}
else
{
    Console.WriteLine(customerOrError.ErrorMessage);
}
```

Как сделать язык предсказуемым:

- У переменных не должно быть возможности поменять их тип,
- Объекты с одинаковыми значениями должны быть равными по-умолчанию,
- Сравнение объектов с разными типами приведет к ошибке компиляции,
- Объекты всегда должны быть инициированы валидными значениями. Если нет - ошибка компиляции,
- Однажды созданные, объекты и коллекции должны быть неизменяемы,
- отсутствие данных или ошибки должны быть **явными**. Никаких null-ов.

F# стремиться к тому, чтобы быть предсказуемым языком:

- переменные не могут менять тип,
- объекты с одинаковыми значениями равны по-умолчанию,
- сравнение объектов с разными типами влечет ошибку компиляции,
- объекты должны быть инициированы валидными значениями. Если нет - ошибка компиляции,
- однажды созданные, объекты и коллекции **почти всегда** неизменяемы,
- отсутствие данных или ошибки **почти всегда** делаются явными. Null-ы (и исключения) - плохой тон.

F# неидеален, кое-где вместо компилятора в дело вступают **соглашения**

Спасибо за внимание!