Использование выражений и пользовательских функций

Николаичев Александр

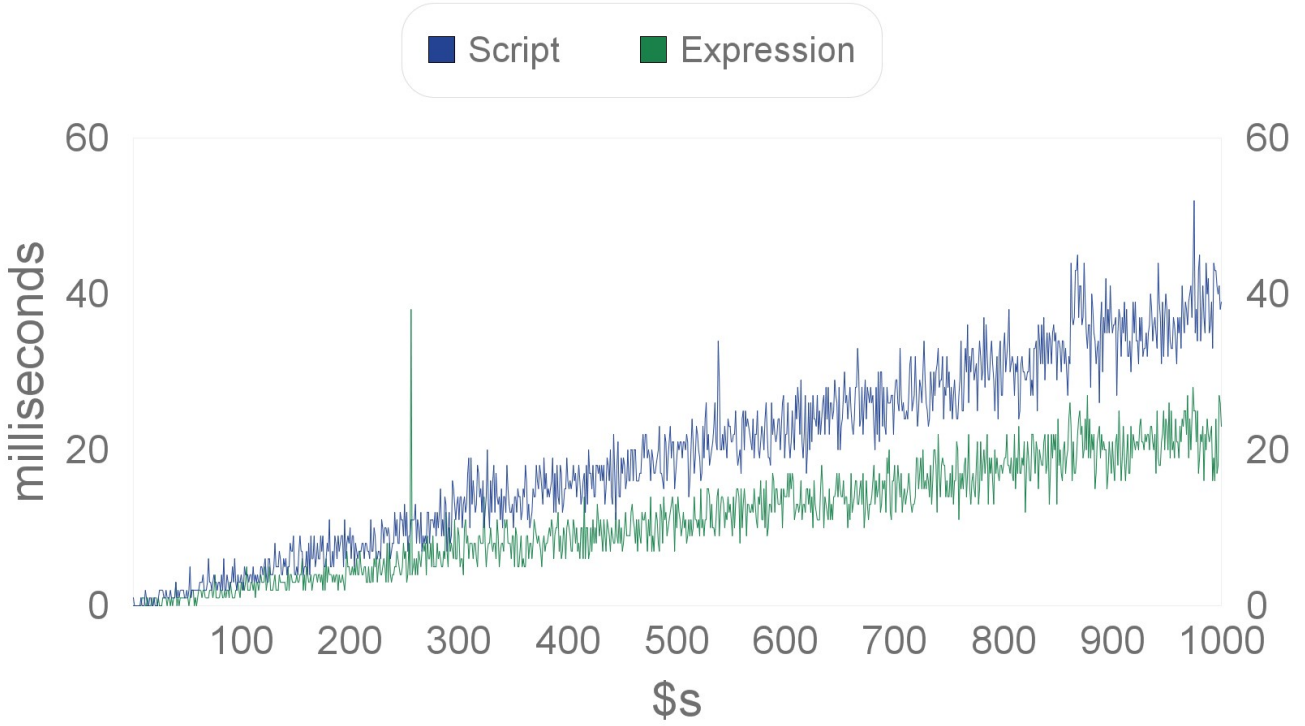
Скрипты против выражений

- Выражения быстрее, но в них сложнее ловить ошибки
- Все ветвления(и циклы до определённого предела) можно перенести в выражения
- Любой шаг скрипта, поведение которого можно задать переменной, можно вызвать в конце, предварительно вычислив переменную

```
1  Go to Layout [ “speed_test” (speed_test) ]
2  Delete All Records [ No dialog ]
3  Go to Layout [ original layout ]
4
5  Set Variable [ $m; Value:1000 ]
6  Set Variable [ $s; Value:1 ]
7  Loop
8    Set Variable [ $i; Value:Let($t=Get(CurrentTimeUTCMilliseconds);0) ]
9    Loop
10     Set Variable [ $i; Value:$i + 1 ]
11     Exit Loop If [ $i > $s ]
12   End Loop
13   Set Variable [ $script_time[$s]; Value:Get(CurrentTimeUTCMilliseconds) - $t ]
14
15   Set Variable [ $i; Value:Let($t=Get(CurrentTimeUTCMilliseconds);0) ]
16   Loop
17     Exit Loop If [ Let( $i = $i + 1; $i > $s ) ]
18   End Loop
19   Set Variable [ $expression_time[$s]; Value:Get(CurrentTimeUTCMilliseconds) - $t ]
20
21   Exit Loop If [ Let($s = $s + 1; $s > $m) ]
22 End Loop
23 Go to Layout [ “speed_test” (speed_test) ]
24 Delete All Records [ No dialog ]
25 Set Variable [ $s; Value:1 ]
26 Loop
27   New Record/Request
28   Set Field [ speed_test::sript; $script_time[$s] ]
29   Set Field [ speed_test::expression; $expression_time[$s] ]
30   Exit Loop If [ Let($s=$s+1;$s > $m) ]
31 End Loop
32 Go to Layout [ original layout ]
```

Сравнение в скорости вычислений

Цикл обёртка также оптимизирован



```
Show Custom Dialog [ Title: "Введите a, b, c"; Message: "ax^2 + bx + c = 0"; Default Button: “OK”, Commit: “Yes”; Input #1: $a, "a"; Input #2: $b, "b"; Input #3: $c, "c" ]
If [ $a = 0 ]
  If [ $b = 0 ]
    If [ $c = 0 ]
      Show Custom Dialog [ Message: "Бесконечно много решений"; Default Button: “OK”, Commit: “Yes” ]
    Else
      Show Custom Dialog [ Message: "Решений нет"; Default Button: “OK”, Commit: “Yes” ]
    End If
  Else
    Show Custom Dialog [ Message: "x = " & ( - $c / $b); Default Button: “OK”, Commit: “Yes” ]
  End If
Else
  Set Variable [ $d; Value:$b * $b - 4 * $a * $c ]
  If [ $d < 0 ]
    Show Custom Dialog [ Message: "Нет решений в поле действительных чисел"; Default Button: “OK”, Commit: “Yes” ]
  Else If [ $d = 0 ]
    Show Custom Dialog [ Message: "x = " & ( - $b / (2 * $a)); Default Button: “OK”, Commit: “Yes” ]
  Else
    Show Custom Dialog [ Message: "x = " & ( - $b + Sqrt($d) / (2 * $a)) & "¶" & "x = " & ( - $b - Sqrt($d) / (2 * $a)); Default Button: “OK”, Commit: “Yes” ]
  End If
End If
```

```
Show Custom Dialog [ Title: "Введите a, b, c"; Message: "ax^2 + bx + c = 0"; Default Button: “OK”, Commit: “Yes”; Input #1: $a, "a"; Input #2: $b, "b"; Input #3: $c, "c" ]
If [ $a = 0 ]
  If [ $b = 0 ]
    If [ $c = 0 ]
      Set Variable [ $t; Value:"Бесконечно много решений" ]
    Else
      Set Variable [ $t; Value:"Решений нет" ]
    End If
  Else
    Set Variable [ $t; Value:"x = " & ( - $c / $b) ]
  End If
Else
  Set Variable [ $d; Value:$b * $b - 4 * $a * $c ]
  If [ $d < 0 ]
    Set Variable [ $t; Value:"Нет решений в поле действительных чисел" ]
  Else If [ $d = 0 ]
    Set Variable [ $t; Value:"x = " & ( - $b / (2 * $a)) ]
  Else
    Set Variable [ $t; Value:"x = " & ( - $b + Sqrt($d) / (2 * $a)) & "¶" & "x = " & ( - $b - Sqrt($d) / (2 * $a)) ]
  End If
End If
Show Custom Dialog [ Message: $t; Default Button: “OK”, Commit: “Yes” ]
```

Перенос ветвлений

- Все условные конструкции переносим внутрь вычисляемого выражения

```
1  Show Custom Dialog [ Title: "Введите a, b, c"; Message: "ax^2 + bx + c = 0"; Default Button: “OK”, Commit: “Yes”; Input #1: $a, "a"; Input #2: $b, "b"; Input #3: $c, "c" ]
2  Show Custom Dialog [ Message: If($a = 0; If($b = 0; If($c = 0; "Бесконечно много решений"; "Решений нет" ); "x = " & ( - $c / $b) ); Let(d = ($b * $b - 4 * $a * $c); If(d < 0; "Нет решений в поле действительных чисел"; If(d = 0; "x = " & ( - $b / (2 * $a)); "x = " & ( - $b + Sqrt(d) / (2 * $a)) & "¶" & "x = " & ( - $b - Sqrt(d) / (2 * $a)) ) ) ) ); Default Button: “OK”, Commit: “Yes” ]
```

- Слева также неполный список шагов скрипта, которые можно вызвать лишь один раз, предварительно определив переменную

```
1  Set Field By Name [ $name; $value ]
2  Import Records [ Source: “$name” ] [ No dialog ]
3
4  Go to Layout [ $name ]
5  Go to Object [ Object Name: $name ]
6
7  Refresh Object [ Object Name: $name; Repetition: 1 ]
8  Refresh Portal [ Object Name: $name ]
9  Set Web Viewer [ Object Name: $name; URL: $url ]
10 Open URL [ $url ] [ No dialog ]
11
12 Close Window [ Name: $name; Current file ]
13 Select Window [ Name: $name; Current file ]
14 Set Window Title [ Of Window: $name; Current file; New Title: $title ]
15
16 Add Account [ Account Name: $name; Password: $password; Privilege Set: [Data Entry Only] ]
17 Delete Account [ Account Name: $name ]
18 Re-Login [ Account Name: $name; Password: $password ] [ No dialog ]
19
20 Perform AppleScript [ Calculated AppleScript: $script ]
21 Perform Script [ By name; $name; Parameter: $param ]
22
23 Show Custom Dialog [ Title: $title; Message: $message; Default Button: “OK”, Commit: “Yes”; Input #1: $input ]
```

Пользовательские функции

- Let([q=Char(34);s="Let([q=Char(34);s=];Left(s;18)&q&s&q&Right(s;Length(s)-18))"];Left(s;18)&q&s&q&Right(s;Length(s)-18))

Тьюринг-полный язык позволяет нам создавать любые абстракции. Однако в синтаксисе Filemaker они могут выглядет громоздко.

Особенности функций

- Вместо циклов — рекурсия
- Глубина стека вызовов = 50000(!)
- Могут менять локальные и глобальные переменные(Let)
- Не могут возвращать функции

Преобразования типов

- <type> & <type> ⇒ <string>
- <type> + <type> ⇒ <number> (-,*,/,^)
- <type> and <type> ⇒ <boolean> (or,xor)

- False ⇒ 0 ⇒ "0"
  True ⇒ 1 ⇒ "1"

Работа с псевдомассивами

- "1¶2¶3¶4¶5" - псевдомассив
- Любое выражение с некой переменной можно преобразовать так, что в явном виде переменная входит в него 1 раз(Let).
  Evaluate("Let(x=" & e & ";x^2 + 5*x - 3")

- start = "Let(x="; end = ";x^2 + 5*x - 3)"
  start & Substitute(arr;"¶";end & " & \"\¶\" & ¶" & start) & end

- "1¶2¶3¶4¶5"

Исходная строка

- "Let(x=1;x^2 + 5*x - 3) & \"\¶\" &
  Let(x=2;x^2 + 5*x - 3) & \"\¶\" &
  Let(x=3;x^2 + 5*x - 3) & \"\¶\" &
  Let(x=4;x^2 + 5*x - 3) & \"\¶\" &
  Let(x=5;x^2 + 5*x - 3)"

Строка для функции Evaluate(помним про экранирование)

- Let(x=1;x^2 + 5*x - 3) & "¶" &
  Let(x=2;x^2 + 5*x - 3) & "¶" &
  Let(x=3;x^2 + 5*x - 3) & "¶" &
  Let(x=4;x^2 + 5*x - 3) & "¶" &
  Let(x=5;x^2 + 5*x - 3)

Выражение в явном виде

- "3¶11¶21¶33¶47"

Результат

- **map_eval**(array;e;expression){
  Let(
   [
    start = "Let(" & e & " = \"";
    end = "\";" & expression & ")"
   ];
  Evaluate(start & Substitute(array;"¶";end & " & \"\¶\" & " & start) & end)
)
  }
- **reduce_eval**(array;a;e;expression;init){
  Let(
   [
    r = GetAsText(Random);
    r = Right(r;Length(r)-2);
    g = "acc_" & r;
    start = g & " = Let([" & acc & " = " & g & ";" & e & " = \"";
    end = "\"];" & expression & ")"
   ];
   Evaluate("Let([" & g & "=\"" & init & "\";" & start & Substitute(array;"¶";end & ";" & start) & end & ";a=" & g & ";" & g & "=\"\"];a)")
)
  }
- **filter_eval**(array;e;expression){
  Let(
   [
    r = GetAsText(Random);
    r = Right(r;Length(r)-2);
    g = "exp_" & r;
    start = g & "_f = " & g & "_f & Let([" & e & " = \"";
    end = "\";" & g & " = " & expression & "];If(" & g & ";" & e & " & \"\¶\";\"\"))";
    a = Evaluate("Let([" & g & "_f=\"\";" & start & Substitute(array;"¶";end & ";" & start) & end & ";" & g & "=\"\";a=" & g & "_f;" & g & "_f=\"\"];a)")
   ];
   Left(a;Length(a)-1)
)
  }

Используем JSON

- Передача параметров функции
  some_function("{\"var_1\":15,\"var_2\":101}")

  Let([
  var_1=JSONGetElement(arguments;"var_1");
  var_2=JSONGetElement(arguments;"var_2")
  ];var_1+var_2)

- Проверка существования ключа
  JSONGetElement($var;"key") = ""

- Разделение на типы значений
  JSONSetElement($var;"key";Pi;JSONNumber)

- Быстрая вставка в webviewer
  "<!DOCTYPE html><html><script>let a = " & $json_var & ";window.alert(a)</script></html>"

Не хватает возможностей

- Иногда реализация сложной логики или функций встроенными средствами утомительна.
- Webviewer — javascript (typescript, coffescript, ...)
- Perform AppleScript ["do shell script \"\""]

Javascript

- Регулярные выражения без плагинов:

```
1  If [ $x = "" ]
2     Set Variable [ $text; Value:"data:text/html,<!DOCTYPE html><html><script>location.href=
      `fmp://$/" & Get(FileName) & ".fmp12?script=" & Get(ScriptName) & "&$x=${"" &
      regex::string & "".search(/^" & regex::regex & "$/)}`</script></html>" ]
3     Set Web Viewer [ Object Name: "regex"; URL: $text ]
4  Else
5     Set Variable [ $$result; Value:If($x≠-1;"True";"False") ]
6     Refresh Window
7  End If
```

```
1  Set Field [ shell::result; "" ]
2  Set Variable [ $path; Value:Let( [ p = Substitute(Get(FilePath);"/";"¶"); p =
   MiddleValues(p;3;ValueCount(p)-3); p = Left(p;Length(p)-1) ]; "/" & Substitute(p;["¶";"/"];["
   ";"\ "]) ) ]
3  Set Variable [ $file; Value:Right(Random;10) & ".csv" ]
4
5  Set Variable [ $code_text; Value:shell::code ]
6  Set Variable [ $code; Value:Substitute(shell::code;["\"";"\\\""];["¶";";"]) ]
7
8  Perform AppleScript [ Calculated AppleScript: "do shell script " & Quote("cd " & $path & ";
   ruby -e \" $>=File.open('" & $file & "','w');" & $code &" \" ") ]
9  Set Variable [ $file_path; Value:Let( [ p = Substitute(Get(FilePath);"/";"¶"); p =
   MiddleValues(p;2;ValueCount(p)-2) ]; "filemac:/" & Substitute(p;"¶";"/") & $file ) ]
10
11 Go to Layout [ “shell” (shell) ]
12 Show All Records
13 Delete All Records [ No dialog ]
14 Import Records [ Source: “$file_path”; Target: “shell”; Method: Add; Character Set: “Mac
   Roman”; Field Mapping: Source field 1 import to shell::result ] [ No dialog ]
15 If [ Get(LastError) ≠ 0 ]
16    New Record/Request
17 End If
18 Show All Records
19 Go to Layout [ original layout ]
20
21 Set Field [ shell::code; $code_text ]
22 Go to Field [ shell::result ]
23
24 Perform AppleScript [ Calculated AppleScript: "do shell script " & Quote("cd " & $path & ";
   rm " & $file ) ]
```

Shell

- Почему ruby?
  Он предустановлен на MacOS
- Для тяжёлых вычислений намного удобнее писать код вне Filemaker, а изнутри вызывать только shell-скрипт, запускающий вычисления и отдающий результат

Спасибо за внимание!

Список функций
https://github.com/AlexandrNikolaichev/Filemaker_functions