

Темная сторона изолятов в Dart



Станислав Чернышев, к.т.н., доцент СПбГУАП и СПбГЭУ



Обо мне

- *Более 10 лет работы в IT (ВПК, МО, проекты на заказ)*
- *Основной вид деятельности – наука и преподавание*
- *Ученики работают в различных компаниях (VK, Paragon Software, Astra, СПб ИАЦ, Радар ММС и т.д.)*
- *Автор учебных книг*

Обо мне

Станислав Чернышев

ОСНОВЫ DART

2-е издание, переработанное и
дополненное



Использование данного файла означает, что вы согласны с условиями
лицензии, расположенной на следующей странице

2024

Что такое изолят?

Способы создания изолята

run

```
Future<R> run<R>(
    FutureOr<R> computation(),
    {String? debugName}
)
```

Способы создания изолята

run

```
Future<R> run<R>(
    FutureOr<R> computation(),
    {String? debugName}
)
```

spawn

```
Future<Isolate> spawn<T>(
    void entryPoint(T message),
    T message,
    {bool paused = false,
    bool errorsAreFatal = true,
    SendPort? onExit,
    SendPort? onError,
    @Since("2.3") String? debugName}
)
```

Создание новой изоляционной группы

spawnUri

```
Future<Isolate> spawnUri(  
    Uri uri,  
    List<String> args,  
    dynamic message,  
    {bool paused = false,  
    SendPort? onExit,  
    SendPort? onError,  
    bool errorsAreFatal = true,  
    bool? checked,  
    Map<String, String>? environment,  
    bool automaticPackageResolution = false,  
    @Since("2.3") String? debugName}  
)
```

spawn vs spawnUri



Kandinsky 3.0

spawn vs spawnUri

Флаг компиляции	Результат	Описание
<i>exe</i>	Автономный исполняемый файл	Автономный исполняемый файл , компилируемый под целевую платформу и содержащий урезанную среду выполнения Dart.
<i>aot-snapshot</i>	Модуль AOT	Файл, компилируемый под целевую платформу, без среды выполнения Dart.
<i>jit-snapshot</i>	Модуль JIT	Файл, компилируемый под целевую платформу, с промежуточным оптимизированным представлением исходного кода , который был получен в ходе учебного запуска программы.
<i>kernel</i>	Kernel модуль	Файл, содержащий промежуточное представление исходного кода в виде абстрактного синтаксического дерева (Kernel AST) . Может запускаться на любых платформах.

exe и *aot-snapshot* не поддерживают dart:mirrors и dart:developer

spawn vs spawnUri

```
Isolate.spawnUri(Uri.parse('new_group.aot'), [], null)
```

spawn vs spawnUri

```
C:\code\dart\deep_isolate\bin>dart compile exe main_isolate.dart
```

```
C:\code\dart\deep_isolate\bin>dart compile aot-snapshot new_group.dart
```

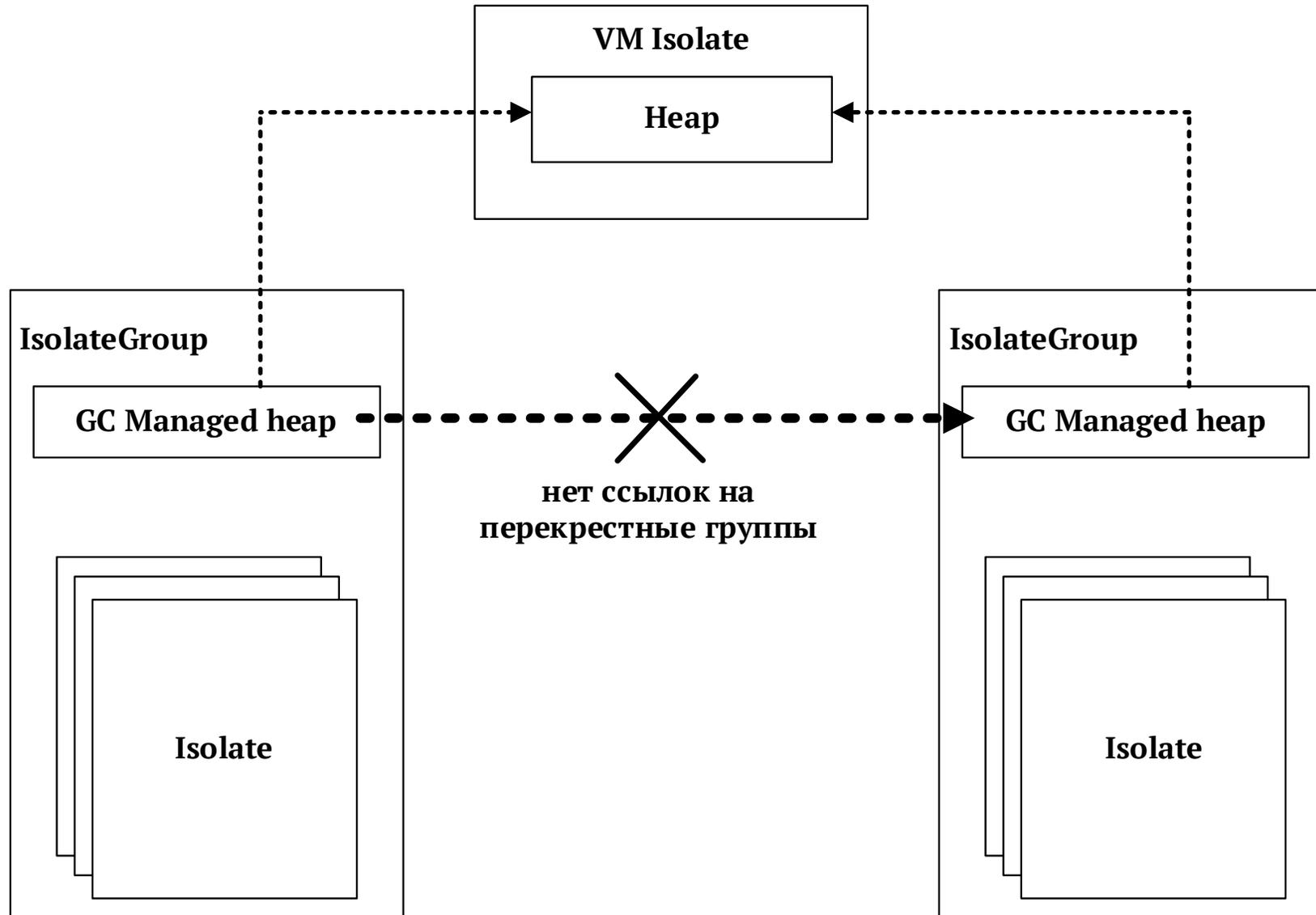
```
C:\code\dart\deep_isolate\bin>main_isolate.exe
```

Ограничения на использование SendPort

spawn vs spawnUri



Условное представление Dart VM





Свойство hashCode класса Object

```
1 class Cat {
2     final name;
3     const Cat(this.name);
4 }
5
6 void main(List<String> arguments) async {
7     var cat = Cat('Max');
8     print('Cat hashCode: ${cat.hashCode}');
9
10    var value = 10;
11    print('Value hashCode: ${value.hashCode}');
12 }
13 // Cat hashCode: 206932202
14 // Value hashCode: 116010
```

Свойство hashCode класса Object

```
14 void main(List<String> arguments) async {
15     var cat = Cat('Max');
16     print('${cat.hashCode}'); // 344563804
17     print('${identityHashCode(cat)}'); // 1013285941
18
19     var value = 10;
20     print('${value.hashCode}'); // 116010
21     print('${identityHashCode(value)}'); // 116010
22 }
```

Как объекту присваивается hashCode?

Как объекту присваивается hashCode?

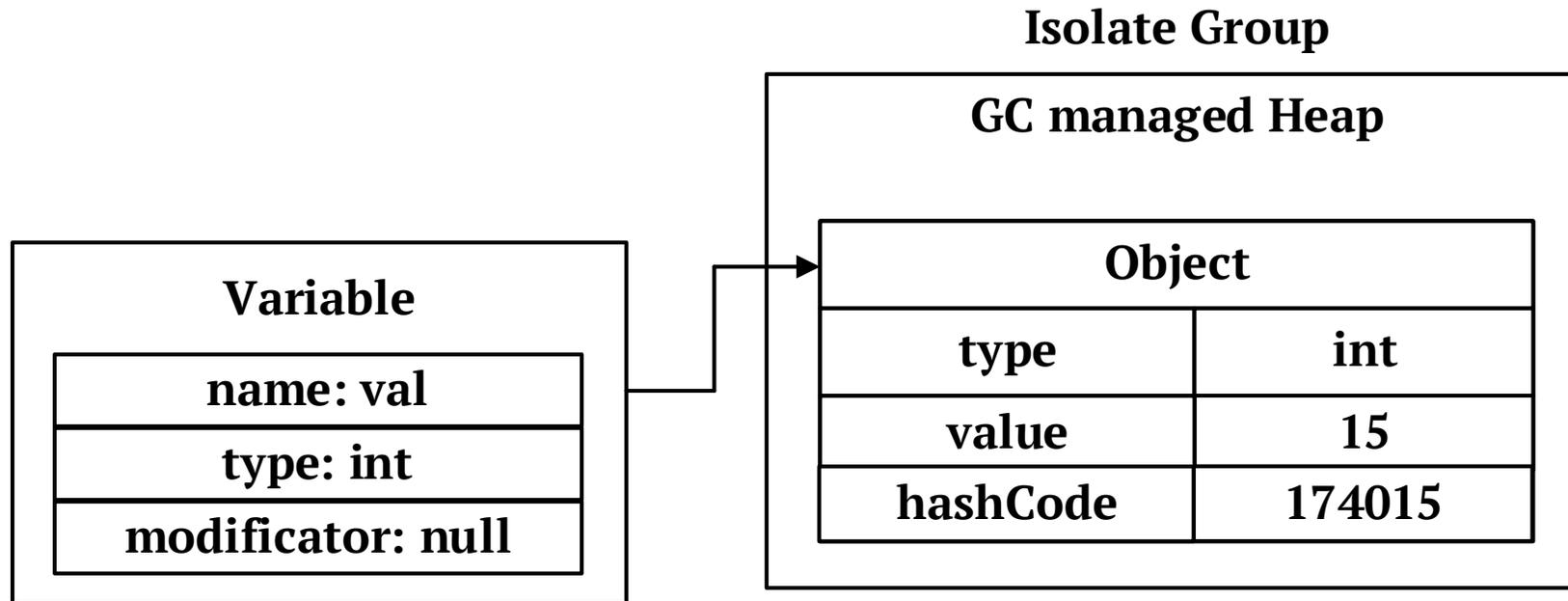
```
// sdk/runtime/vm/object_graph.cc
uint32_t HeapSnapshotWriter::GetHashHelper(Thread* thread, ObjectPtr obj) {
    uint32_t hash;
    #if defined(HASH_IN_OBJECT_HEADER)
        hash = Object::GetCachedHash(obj);
        if (hash == 0) {
            ASSERT(
                !thread->heap()->old_space()->IsObjectFromImagePages(obj)
            );
            hash = GenerateHash(thread->random());
            Object::SetCachedHashIfNotSet(obj, hash);
        }
    #else
        Heap* heap = thread->heap();
        hash = heap->GetHash(obj);
        if (hash == 0) {
            ASSERT(!heap->old_space()->IsObjectFromImagePages(obj));
            hash = GenerateHash(thread->random());
            heap->SetHashIfNotSet(obj, hash);
        }
    #endif
    return hash;
}
```

Как объекту присваивается hashCode?

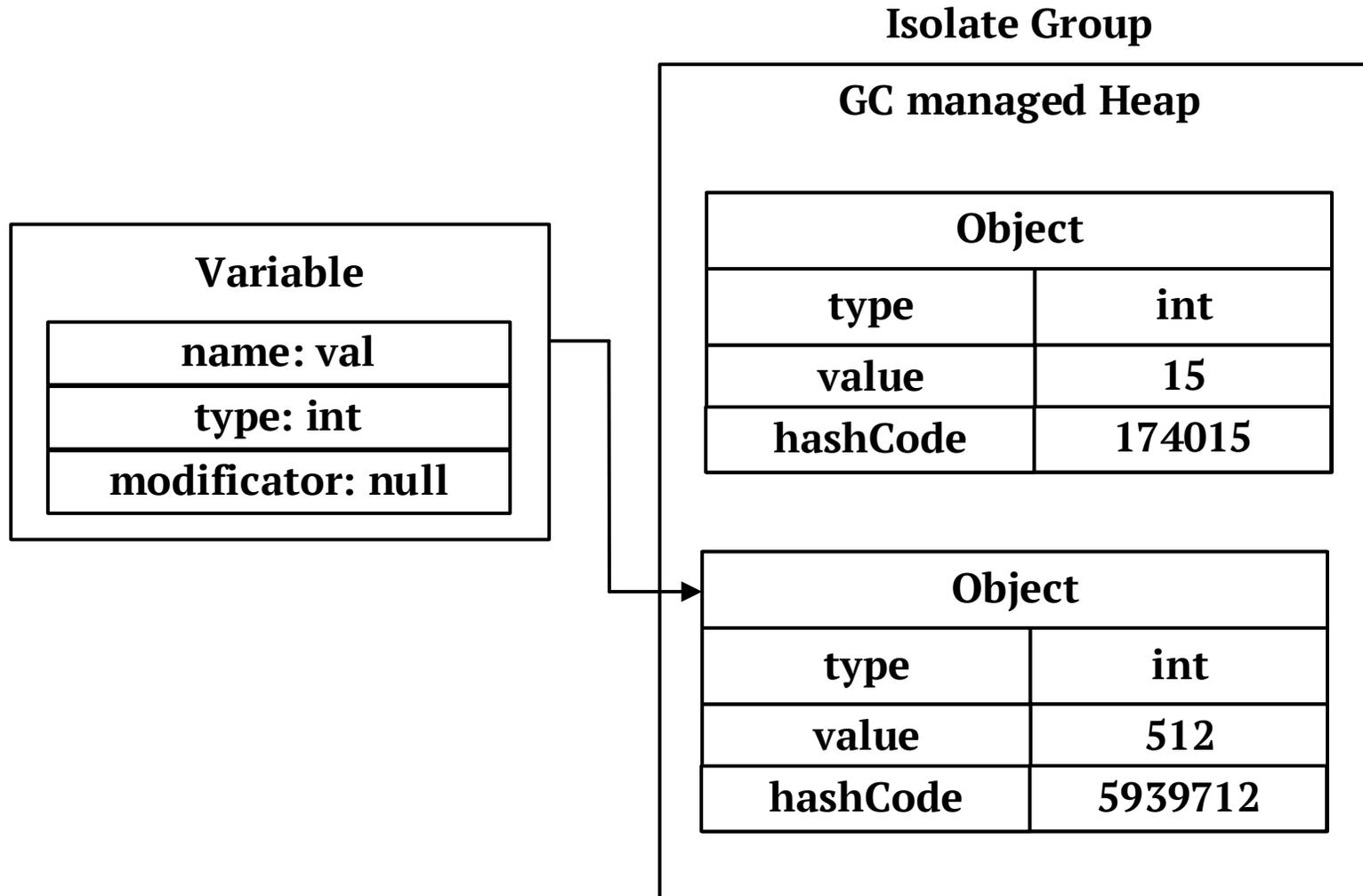
```
// sdk/runtime/vm/object_graph.cc
static uint32_t GenerateHash(Random* random) {
    uint32_t hash;
    do {
        hash = random->NextUInt32();
    } while (hash == 0 || (kSmiBits < 32 && !Smi::IsValid(hash)));
    return hash;
}
```

Объявление переменной или имени?

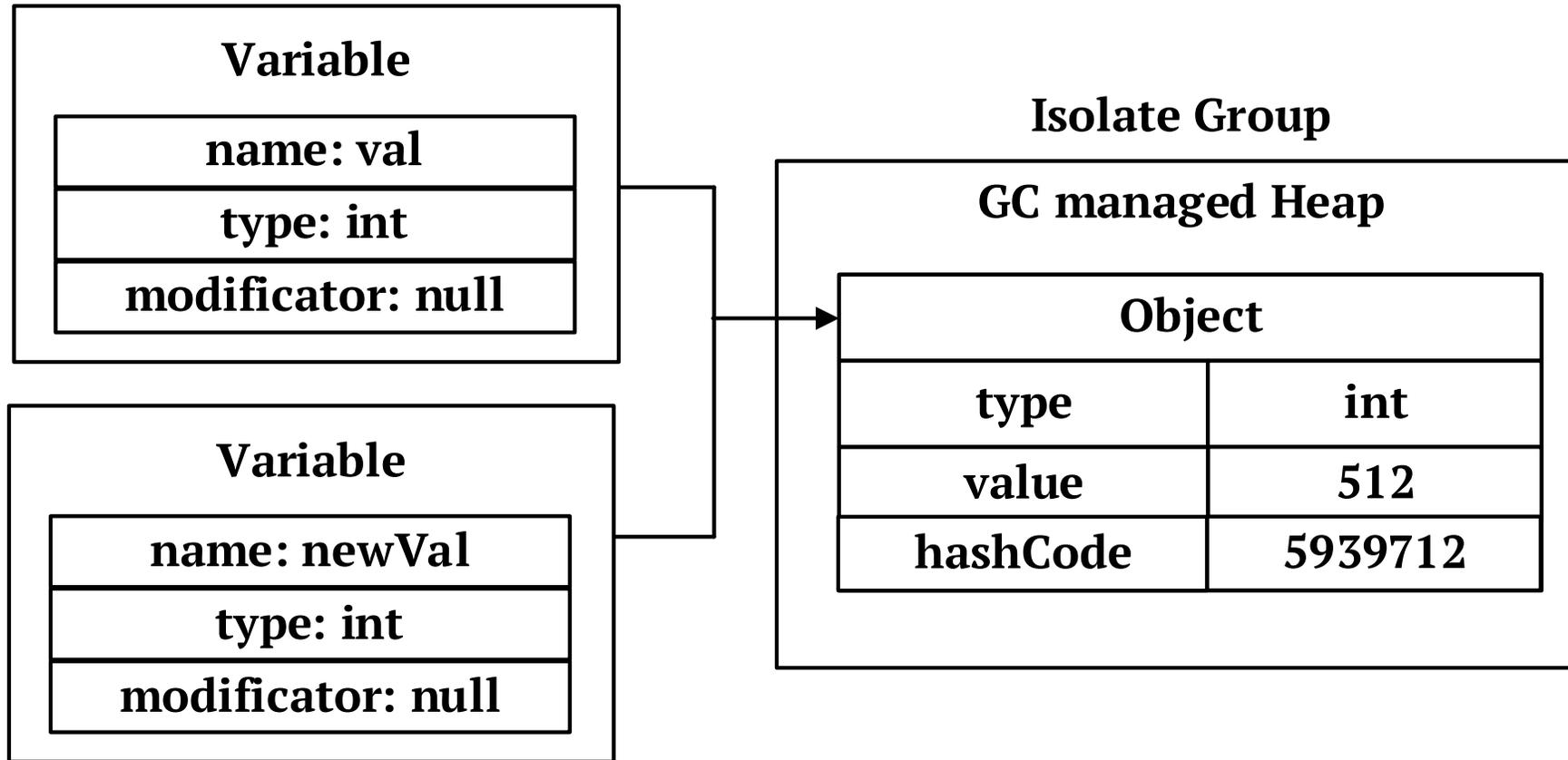
```
var val = 15;
```



`val = 512;`



`var newVal = val;`



```
1 import 'dart:convert';
2
3 void main(List<String> arguments) async {
4   var val = 'Hello';
5   var newVal = 'Hello';
6   print('${val.hashCode}'); // 828172268
7   print('${newVal.hashCode}'); // 828172268
8   print('${identical(val, newVal)}'); // true
9
10  var list = [1, 2, 3];
11  var newList = [1, 2, 3];
12  print('${list.hashCode}'); // 664960072
13  print('${newList.hashCode}'); // 464510092
14  print('${identical(list, newList)}'); // false
15
16  var value = jsonDecode('{"a": 1, "b": 1}');
17  print(identical(value['a'], value['b'])); // true
18  print(identityHashCode(value['a'])); // 11601
19  print(identityHashCode(value['b'])); // 11601
20 }
```

```
1 class Cat {
2     final name;
3     const Cat(this.name);
4 }
5
6 void main(List<String> arguments) async {
7     var cat = const Cat('Max');
8     var newCat = const Cat('Max');
9     print('${cat.hashCode}'); // 940337853
10    print('${newCat.hashCode}'); // 940337853
11    print('${identical(cat, newCat)}'); // true
12
13    var list = const [1, 2, 3];
14    var newList = const [1, 2, 3];
15    print('${list.hashCode}'); // 286116521
16    print('${newList.hashCode}'); // 286116521
17    print('${identical(list, newList)}'); // true
18 }
```



```
1 void main(List<String> arguments) {
2     var str1 = 'Mobius';
3     var str2 = 'Mobius';
4     print(identical(str1, str2)); // true
5
6     var str3 = 'Mob' + 'ius';
7     print(identical(str1, str3)); // false
8
9     var value = jsonDecode('{ "a": "f", "b": "f" }');
10    print(identical(value['a'], value['b'])); // true
11
12    value = jsonDecode('{ "a": "wtf", "b": "wtf" }');
13    print(identical(value['a'], value['b'])); // false
14    print(identityHashCode(value['a']) == identityHashCode(value['b'])); // true
15 }
```



Жизненный цикл объекта



1-поколение



2-поколение



Алгоритм сборки мусора в 1-м поколении



Алгоритм сборки мусора во 2-м поколении

1. живые объекты помечаются меткой;

Алгоритм сборки мусора во 2-м поколении

1. живые объекты помечаются меткой;
2. прочесывание памяти для удаления мертвых объектов;

Алгоритм сборки мусора во 2-м поколении

1. живые объекты помечаются меткой;
2. прочесывание памяти для удаления мертвых объектов;
3. уплотнение объектов в памяти, что позволяет уменьшить ее фрагментацию.

Общая память изоляционных групп



```

9 void main(List<String> arguments) async {
10   await Future.wait([
11     Future.delayed(Duration(seconds: 2), () {
12       var a = const ['22'];
13       var b = 44;
14       var cat = const Cat('Max');
15       var c = ['45'];
16       print('Main cat: ${identityHashCode(cat)}'); // 693067717
17       print('Main a: ${identityHashCode(a)}'); // 405082921
18       print('Main b: ${identityHashCode(b)}'); // 510444
19       print('Main c: ${identityHashCode(c)}'); // 959463167
20     })),
21     Isolate.run(() => isolateFoo()),
22   ]);
23 }
24
25 void isolateFoo() async {
26   var cat = const Cat('Max');
27   final a = const ['22'];
28   var b = 44;
29   var c = ['45'];
30   print('Isolate cat: ${identityHashCode(cat)}'); // 693067717
31   print('Isolate a: ${identityHashCode(a)}'); // 405082921
32   print('Isolate b: ${identityHashCode(b)}'); // 510444
33   print('Isolate c: ${identityHashCode(c)}'); // 243495788
34
35   b += 3;
36   print('Isolate b: ${identityHashCode(b)}'); // 545247
37   await Future.delayed(Duration(seconds: 2));
38 }

```

```

10 void main(List<String> arguments) async {
11   await Future.wait([
12     Future.delayed(Duration(seconds: 2), () {
13       var a = const ['22'];
14       var cat = const Cat('Max');
15       var b = ['45'];
16       var c = 44;
17       var d = 'hi!';
18       var e = 23.1;
19       print('Main cat: ${identityHashCode(cat)}'); // 640354238
20       print('Main a: ${identityHashCode(a)}'); // 896670331
21       print('Main b: ${identityHashCode(b)}'); // 913878150
22       print('Main c: ${identityHashCode(c)}'); // 510444
23       print('Main d: ${identityHashCode(d)}'); // 1009204041
24       print('Main e: ${identityHashCode(e)}'); // 15509272291868675
25     }),
26     Isolate.spawnUri(Uri.parse('new_group.dart'), [], null),
27   ]);
28 }

```

```

1 void main() {
2   var a = const ['22'];
3   var cat = const Cat('Max');
4   var b = ['45'];
5   var c = 44;
6   var d = 'hi!';
7   var e = 23.1;
8   print('IG cat: ${identityHashCode(cat)}'); // 478627355
9   print('IG a : ${identityHashCode(a)}'); // 174622686
10  print('IG b : ${identityHashCode(b)}'); // 167233800
11  print('IG c : ${identityHashCode(c)}'); // 510444
12  print('IG d: ${identityHashCode(d)}'); // 1009204041
13  print('IG e: ${identityHashCode(e)}'); // 15509272291868675
14 }

```

Разница в передаче обычных и константных объектов между изолятами группы

```
1 void main(List<String> arguments) async {
2   var cat1 = const Cat('Max');
3   print('Main cat1: ${identityHashCode(cat1)}');
4
5   var cat2 = Cat('Max');
6   print('Main cat2: ${identityHashCode(cat2)}');
7
8   await Future.wait([
9     Isolate.run(() => isolateFoo(cat1, 1)),
10    Isolate.run(() => isolateFoo(cat2, 2)),
11  ]);
12  await Future.delayed(Duration(seconds: 3));
13 }
14
15 void isolateFoo(Cat cat, int number) async {
16   print('Isolate cat$number: ${identityHashCode(cat)}');
17   await Future.delayed(Duration(seconds: 1));
18 }
19 // Main cat1: 528416660 !!!
20 // Main cat2: 9390701
21 // Isolate cat1: 528416660 !!!
22 // Isolate cat2: 683512425
```

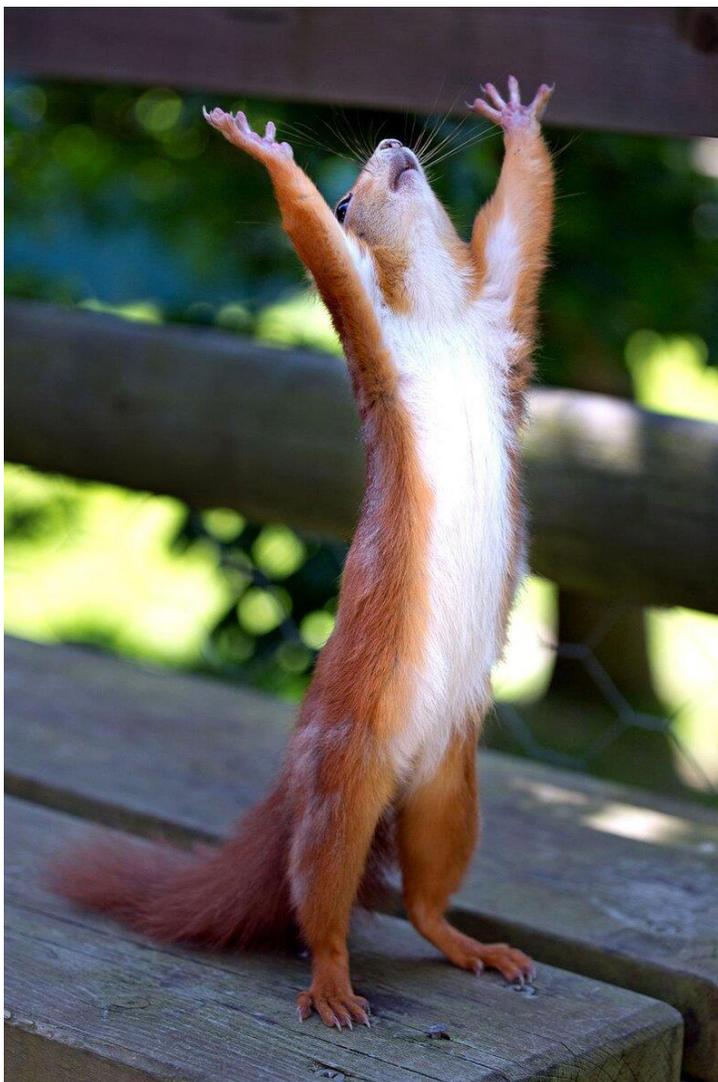
```
C:\code\dart\deep_isolate>bin\deep_isolate.exe 1000000 1  
0:00:01.621376 count: 0
```

```
C:\code\dart\deep_isolate>bin\deep_isolate.exe 1000000 0  
0:00:01.446942 count: 0
```

Создание изолята из другого изолята

```
1 import 'dart:isolate';
2 import 'dart:async';
3
4 void main() async {
5   print('Main isolate started');
6   await Future.wait([
7     Isolate.run(firstIsolate),
8     Future.delayed(Duration(seconds: 1)),
9   ]);
10 }
11
12 Future<void> firstIsolate() async {
13   print('First isolate started');
14   await Future.wait([
15     Isolate.run(secondIsolate),
16     Future.delayed(Duration(seconds: 1)),
17   ]);
18 }
19
20 Future<void> secondIsolate() async {
21   print('Second isolate started');
22 }
23 // Main isolate started
24 // First isolate started
25 // Second isolate started
```

Будущее изолятов



Shared Memory Multithreading #3531

Open mraleph wants to merge 7 commits into `dart-lang:main` from `mraleph:shared-memory`

Conversation 168 Commits 7 Checks 3 Files changed 1



mraleph commented on Dec 21, 2023

Member ...

Initial proposal attempting to introduce shared memory multithreading.

/cc @leafpetersen @lrhn @eerstg @munificent

/cc @a-siva @alexmarkov @mkustermann @dcharkes @liamappelbe @chinmaygarde

/fyi @syg @kevmoo

👍 34 🗑️ 2 🍷 9 ❤️ 9 🚀 14 👁️ 4

Initial proposal

✓ f47d

Выводы

- Объекты простых общих типов данных представлены одним экземпляром;



Выводы

- Объекты простых общих типов данных представлены одним экземпляром;
- Константа времени компиляции представлена одним объектом только в рамках изоляционной группы;



Выводы

- Объекты простых общих типов данных представлены одним экземпляром;
- Константа времени компиляции представлена одним объектом только в рамках изоляционной группы;
- Передача константы времени компиляции и объектов простых общих типов данных между изолятами группы требует меньше затрат;



Выводы



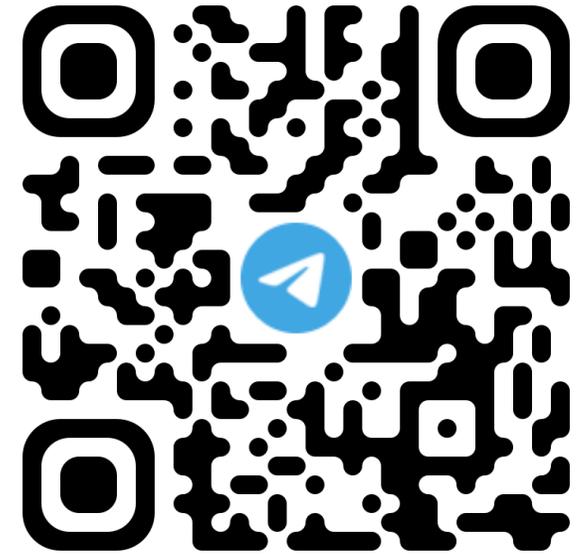
- Объекты простых общих типов данных представлены одним экземпляром;
- Константа времени компиляции представлена одним объектом только в рамках изоляционной группы;
- Передача константы времени компиляции и объектов простых общих типов данных между изолятами группы требует меньше затрат;
- От кого пошло, что изолят нельзя запускать из другого изолята?

Выводы



- Объекты простых общих типов данных представлены одним экземпляром;
- Константа времени компиляции представлена одним объектом только в рамках изоляционной группы;
- Передача константы времени компиляции и объектов простых общих типов данных между изолятами группы требует меньше затрат;
- От кого пошло, что изолят нельзя запускать из другого изолята?
- Изоляты – это не страшно!

Вопросы?



Станислав Чернышев, к.т.н., доцент СПбГУАП и СПбГЭУ

