

Лекция 6. Разработка на Flutter

Полезные ссылки:

- Официальный сайт Dart;
- Программа Visual Studio Code.

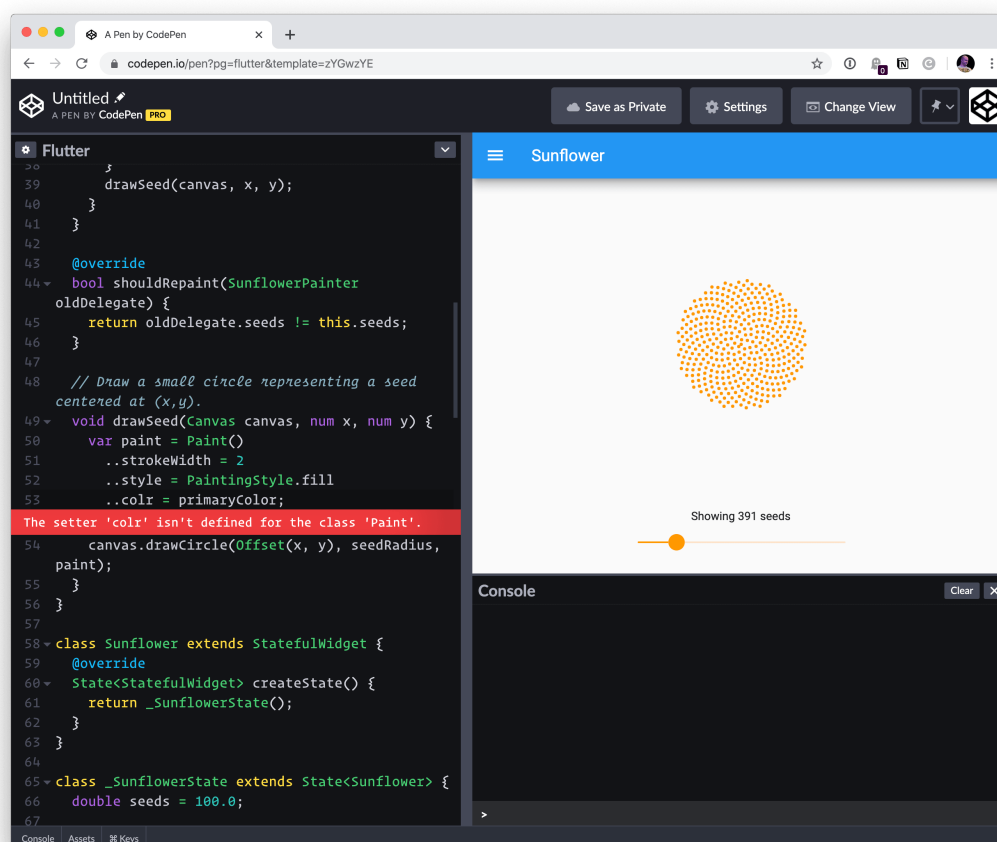
Flutter – что это такое?

Flutter – относительно новый фреймворк, что был разработан компанией Google. Его первое появление датируется 2015 годом. На основе Flutter любой разработчик может создать мобильное приложение как под Андроид, так и под iOS платформу. Помимо этого еще есть возможность создания веб приложений, но это уже совсем другая история.

В будущем планируется сделать поддержку разработки под компьютерные операционные системы: Windows, Mac, Linux и Google Fuchsia.

Что такое Dart?

Flutter – это лишь фреймворк, что в свою очередь написан на основе языков программирования C++ и Dart. Про язык Dart, вы, скорее всего, мало что знаете, так как язык появился лишь в 2011 году и был создан той же компанией Google. Основная цель данного языка – замена языка JavaScript. По этой причине Dart делает множество схожих действий с языком JavaScript, но делает это в более современном и верном стиле.



При разработке на Flutter весь код пишется на основе языка Dart. Если язык вам не знаком, то не переживайте, ведь мы будем его изучать в ходе курса и лишь после его изучения приступим к Flutter.

Удобство использования Dart в связке с Flutter также прослеживается при разработке проектов. Дело в том, что вы можете видеть все изменения по вашему проекту сразу же при сохранении проекта. Если говорить про стандартный способ разработки мобильных приложений, то чтобы увидеть изменения сперва нужно компилировать проект и запускать его на виртуальном устройстве.

Dart – язык программирования, что появился относительно недавно в 2011 году. Он был разработан компанией Google. Его предназначение стать альтернативой языку JavaScript. Dart выполняет множество действий, что в некоторой степени схожи с действиями из языка ДжаваСкрипт.

Основы языка

Любые программы на **Dart** состоят из нескольких компонентов:

- Подключение библиотек (по необходимости);
- Указание функции `main`, что ничего не возвращает.

Каждая строка должна закрываться точкой с запятой (;). В языке можно использовать все стандартные конструкции: переменные, списки, условия, циклы, функции и возможности ООП.

Переменные

Типы переменных в языке Дарт могут указываться перед названием переменной. Если это не сделать, то нужно прописывать ключевое слово `var`.

От конкретного значения будет зависеть тип данных переменной. Если мы укажем что переменная со значением 5, то тип данных будет указан для целых чисел. И наоборот, если указать тип данных для целых чисел `int`, то в неё мы не сможем записать строку или число с точкой.

В языке есть возможность создания динамических переменных, что смогут менять свой тип данных по ходу выполнения программы. Пример такой переменной:

```
dynamic some = 'text';  
some = 5;
```

Для создания констант необходимо прописывать ключевое слово **const** в начале переменной.

Прочие конструкции

Условия, списки, циклы и функции были рассмотрены в ходе уроки. Можно заметить, что по большей части они не сильно отличаются от аналогичных конструкций из других языков программирования.

Если у вас уже есть навыки работы с другими языками программирования, то освоить Дарт вам не составит никакого труда.

Почему Flutter крут?

Предположим, вы решили разработать некий мобильный проект. Тут же возникает много вопросов. Какой язык изучить: Джава или же Swift, а может лучше и вовсе попробовать C++? А какой язык изучить, если нужно под несколько платформ разрабатывать?

И таких вопросов получается достаточно много. На все из них нужно ответить, прежде чем разработать мобильное приложение. Более того, ваше приложение, скорее всего, будет работать лишь под одну ОС.

Разработчики это понимают и стараются избавить других молодых разработчиков от этой головной боли. В качестве альтернативного решения они придумали технологию Flutter.

Flutter является полноценным современным решением для разработчика, что решил создать мобильное приложение.

- Во-первых, никаких сложных языков программирования изучать не придется. Язык Dart хоть и новый язык, но его синтаксис очень прост.
- Во-вторых, разработка будет происходить под каждую платформу сразу же. Теперь вы просто пишете один код и получаете приложение как под iOS, так и под Андроид.
- В-третьих, ваши мобильные проекты ничем не будут отличаться от стандартного подхода к разработке приложений.

Получается, что используя Flutter вы облегчаете себе процесс разработки, получаете сразу готовое решение под разные операционные системы, да и ко всему этому получаете массу готовых дополнительных библиотек, что можно использовать для быстрого добавления дополнительного функционала.

На что способен Flutter?

Flutter способен на многое. Начиная от создания простого проекта с одной страницей и до создания полноценного приложения, где у вас будет множество функций, красивый дизайн, безопасность и подключение к базе данных.

На Flutter уже было создано большое количество проектов. Можно выделить такие мобильные проекты:

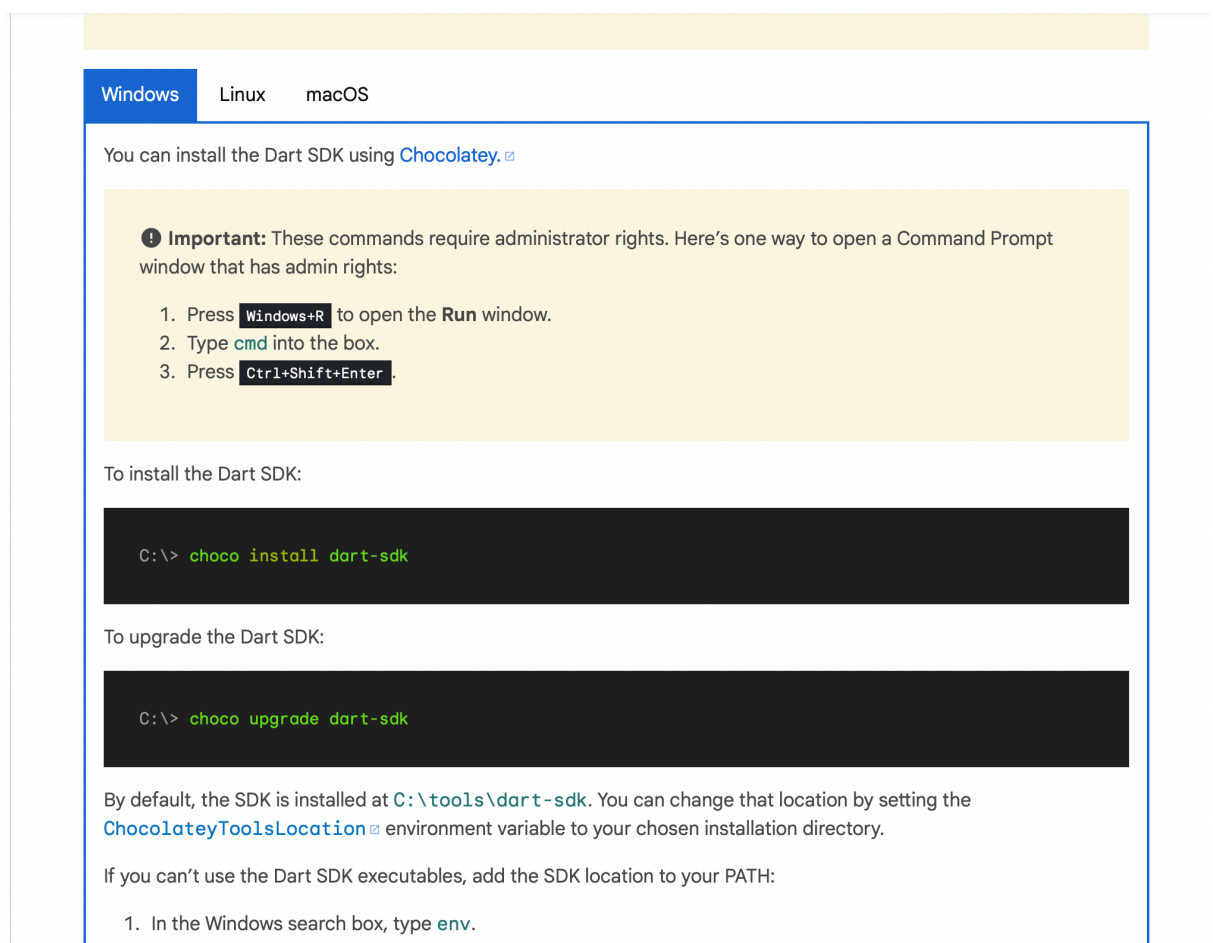
- Google Ads – сервис рекламы от Google;
- Alibaba – мобильное приложение всемирноизвестного магазина Alibaba;
- Reflectly – фитнес приложение;
- Birch Finance – финансовое приложение;
- Hamilton Musical – музыкальное приложение.

Flutter действительно является полноценным фреймворком для создания мобильных приложений, поэтому если вам нужно разработать проект, то смело можете выбирать данный фреймворк.

Сравнение с React Native

Сравнение платформ – дело сложное, ведь у каждой платформы есть свои фанаты и разработчики. Сказать что является более хорошей платформой очень сложно, ведь обе делают свое дело очень хорошо. **React Native** делает все те же функции, вот только он разработан компанией Facebook, а Flutter компанией Google. Сказать что из них лучше или хуже – очень сложно.

1) Install - <https://dart.dev/get-dart>



The screenshot shows the 'Windows' tab of the Dart SDK installation guide. It includes instructions on how to install the SDK using Chocolatey, with a note about requiring administrator rights. It provides a list of steps to open a Command Prompt with admin rights, followed by the command to install the SDK: `C:\> choco install dart-sdk`. It also shows the command to upgrade the SDK: `C:\> choco upgrade dart-sdk`. Finally, it mentions the default installation path and how to change it by setting the `ChocolateyToolsLocation` environment variable, and provides instructions on how to add the SDK location to the PATH.

Windows Linux macOS

You can install the Dart SDK using [Chocolatey](#).

Important: These commands require administrator rights. Here's one way to open a Command Prompt window that has admin rights:

1. Press **Windows+R** to open the **Run** window.
2. Type `cmd` into the box.
3. Press **Ctrl+Shift+Enter**.

To install the Dart SDK:

```
C:\> choco install dart-sdk
```

To upgrade the Dart SDK:

```
C:\> choco upgrade dart-sdk
```

By default, the SDK is installed at `C:\tools\dart-sdk`. You can change that location by setting the [ChocolateyToolsLocation](#) environment variable to your chosen installation directory.

If you can't use the Dart SDK executables, add the SDK location to your PATH:

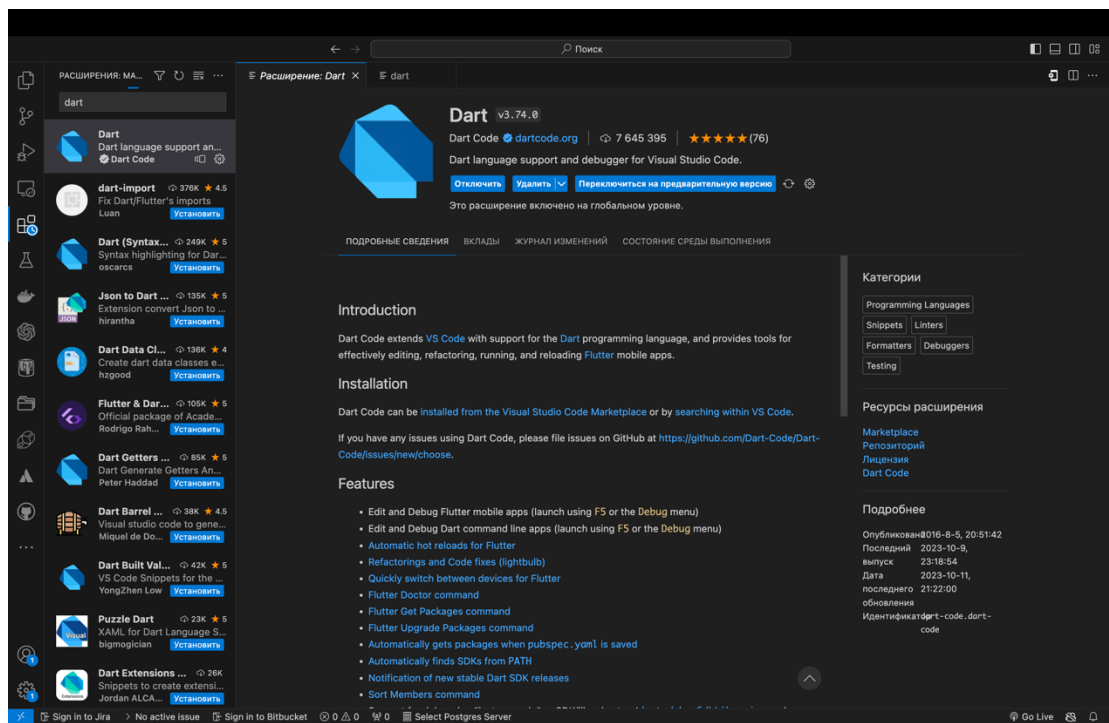
1. In the Windows search box, type `env`.

Mac OS Install:

```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ  КОММЕНТАРИИ
$ brew install dart
zsh: command not found: $
zsh: command not found: $
$ brew tap dart-lang/dart
Running 'brew update --auto-update'...
==> Auto-updated Homebrew!
Updated 4 taps (hashicorp/tap, homebrew/services, homebrew/core and homebrew/cask).
==> New Formulae
bashunit          incus                  jupyter-r          netlistsvg         python-psutil
==> New Casks
kuaitie                               low-profile
You have 37 outdated formulae installed.

==> Tapping dart-lang/dart
Cloning into '/opt/homebrew/Library/Taps/dart-lang/homebrew-dart'...
remote: Enumerating objects: 3784, done.
remote: Counting objects: 100% (1538/1538), done.
remote: Compressing objects: 100% (415/415), done.
remote: Total 3784 (delta 1282), reused 1326 (delta 1117), pack-reused 2246
Receiving objects: 100% (3784/3784), 682.42 KiB | 1.45 MiB/s, done.
Resolving deltas: 100% (2558/2558), done.
Tapped 22 formulae (54 files, 886.0KB).
$ brew install dart
==> Fetching dart-lang/dart/dart
==> Downloading https://storage.googleapis.com/dart-archive/channels/stable/release/3.1.3/sdk/dart-sdk-macos-arm64-release.zip
###
2.9%
```

Установка плагинов Dart и Flutter в VSCode



РАСШИРЕНИЯ: МА... flutter

Flutter

Flutter support and debug...

Dart C...

Идет установка

6.9K

5

Flutter Widget...

A set of helpful widget sni...

Alexis Villa...

Установить

1M

5

[F] Flutter F...

Quickly scaffold flutter blo...

Igor Kravche...

Установить

391K

5

Flutter Tree

Extension for Flutter to bul...

Marcelo Vela...

Установить

326K

5

Awesome Fl...

Awesome Flutter Snippets...

Neevash R...

Установить

368K

5

Flutter Color

This plugin help you to ea...

Nilesh Chavan

Установить

274K

4.5

Flutter Intl

Flutter localization binding...

Localizely

Установить

267K

5

Flutter Helpers

Helper utilities for flutter p...

Akshar Patel

Установить

214K

5

flutter-styli...

Flutter Stylizer organizes y...

gmilewis-vsc...

Установить

158K

4.5

Flutter & Dar...

Official package of Acade...

Rodrigo Rah...

Установить

105K

5

Flutter Snipp...

Supercharge your Flutter ...

Maruf Hassan

Установить

88K

5

Расширение: Flutter

Flutter

v3.74.0

Dart Code

dartcode.org

6 981 619

★★★★★(76)

Flutter support and debugger for Visual Studio Code.

Идет установка

ПОДРОБНЫЕ СВЕДЕНИЯ

ВКЛАДЫ

ЖУРНАЛ ИЗМЕНЕНИЙ

ЗАВИСИМОСТИ

Introduction

This **VS Code** extension adds support for effectively editing, refactoring, running, and reloading **Flutter** mobile apps. It depends on (and will automatically install) the **Dart extension** for support for the **Dart** programming language.

Note: Projects should be run using **F5** or the **Debug** menu for full debugging functionality. Running from the built-in terminal will not provide all features.

Installation

Install from the Visual Studio Code Marketplace or by searching within VS Code. The Dart extension will be installed automatically, if not already installed.

Documentation

Please see the [Flutter documentation](#) for using VS Code.

Reporting Issues

Issues for both Dart and Flutter extensions should be reported in the [Dart-Code issue tracker](#).

Категории

Programming Languages

Snippets

Linters

Debuggers

Formatters

Ресурсы расширения

Marketplace

Репозиторий

Лицензия

Dart Code

Подробнее

Опубликован

#018-4-18,

22:49:03

Последний

2023-10-2, 21:25:41

выпуск

Идентификатор

dart-code.Flutter

Sign in to Jira

No active issue

Sign in to Bitbucket

0 0

0 0

Select Postgres Server

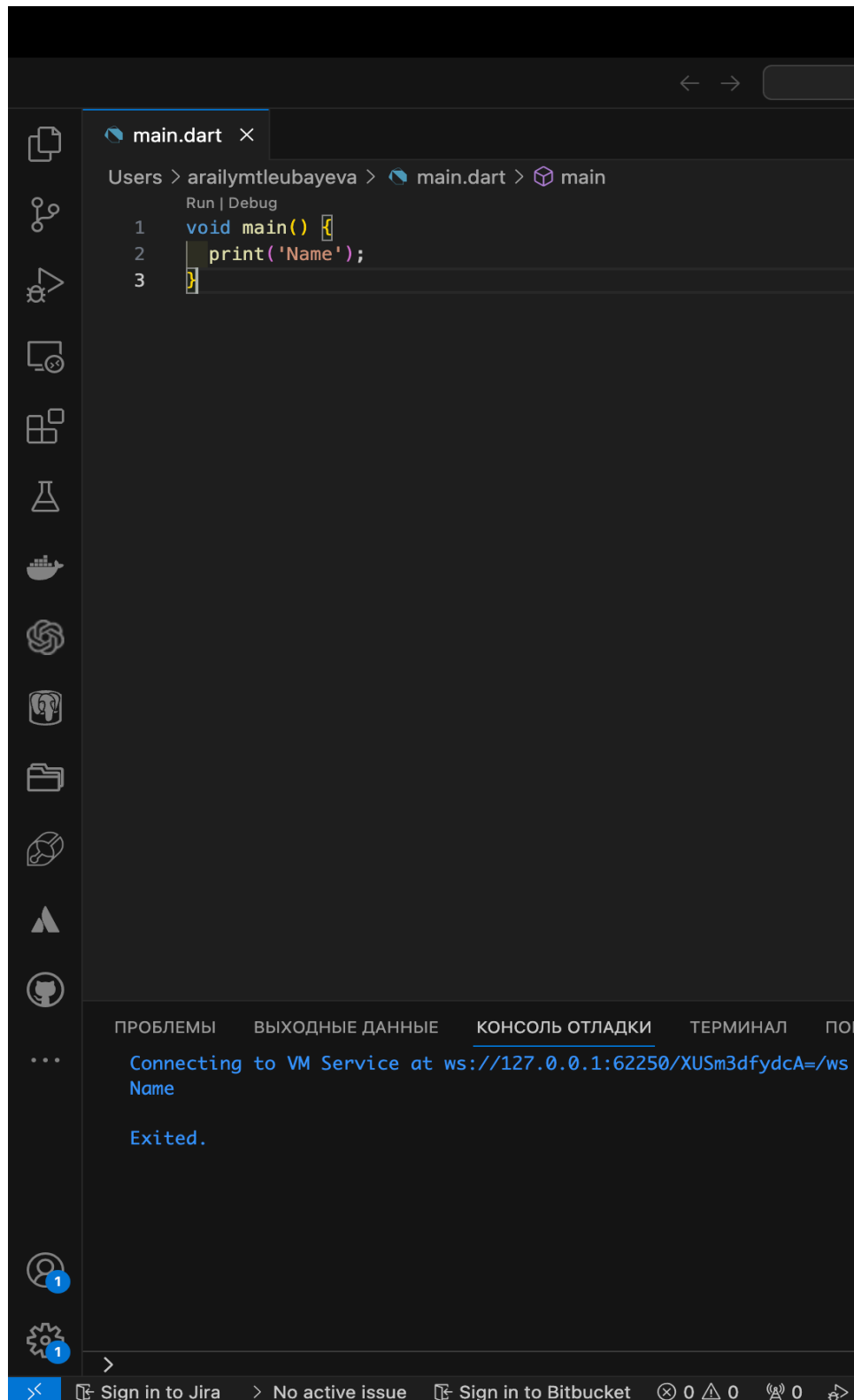
Go Live

Изучение языка Dart. Основные концепции

Примеры кода Dart:

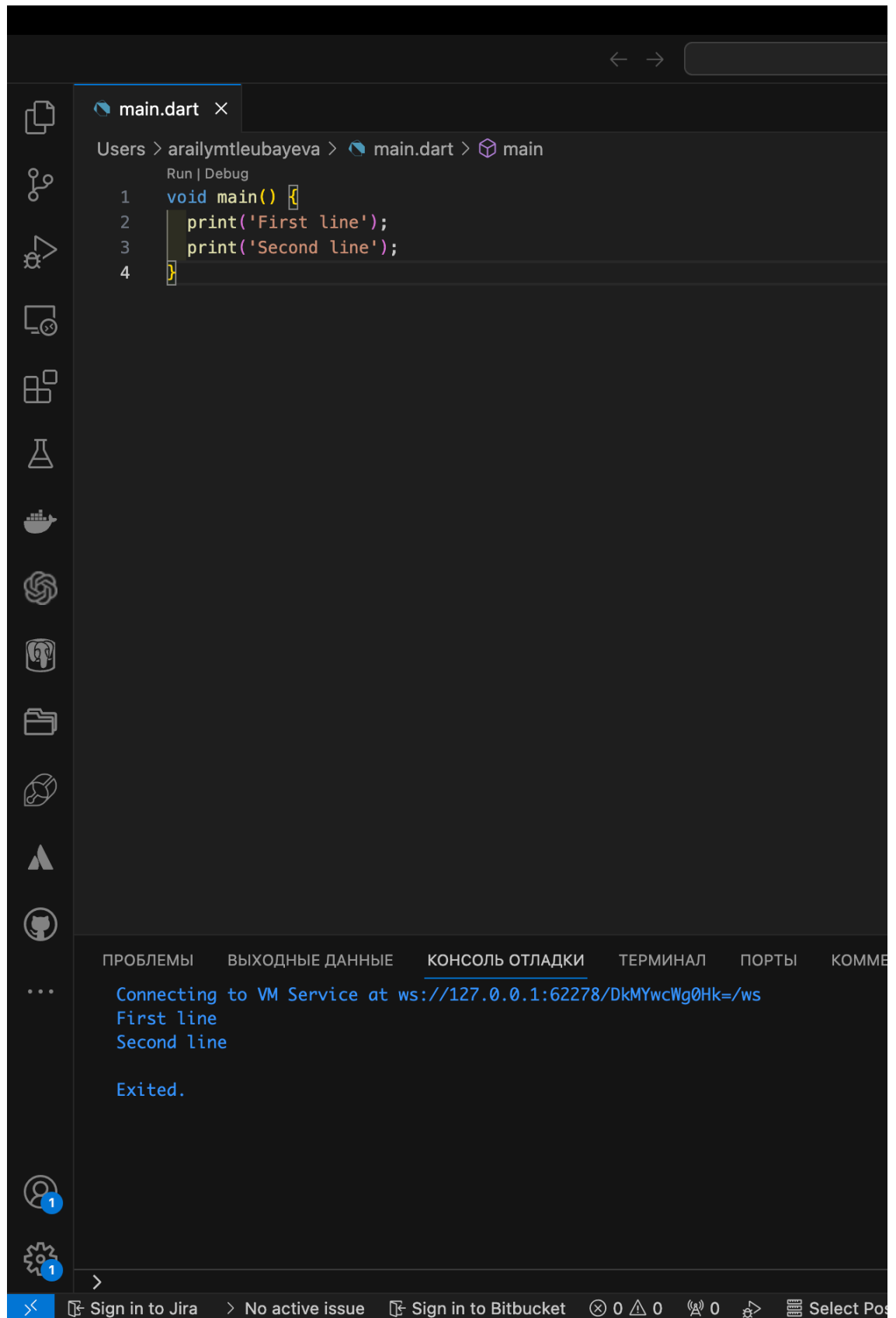
1) Вывод информации

Напишите программу, что выведет ваше имя в консоль.



2) Несколько строк

Выполните вывод информации в нескольких строках.



The screenshot shows an IDE interface with a dark theme. The top bar has navigation arrows and a search input. The left sidebar contains icons for Explorer, Search, Run and Debug, Source Control, Extensions, Testing, Remote Explorer, Docker, AI Assistant, Accounts, and Settings. The main editor area displays a file named `main.dart` with the following Dart code:

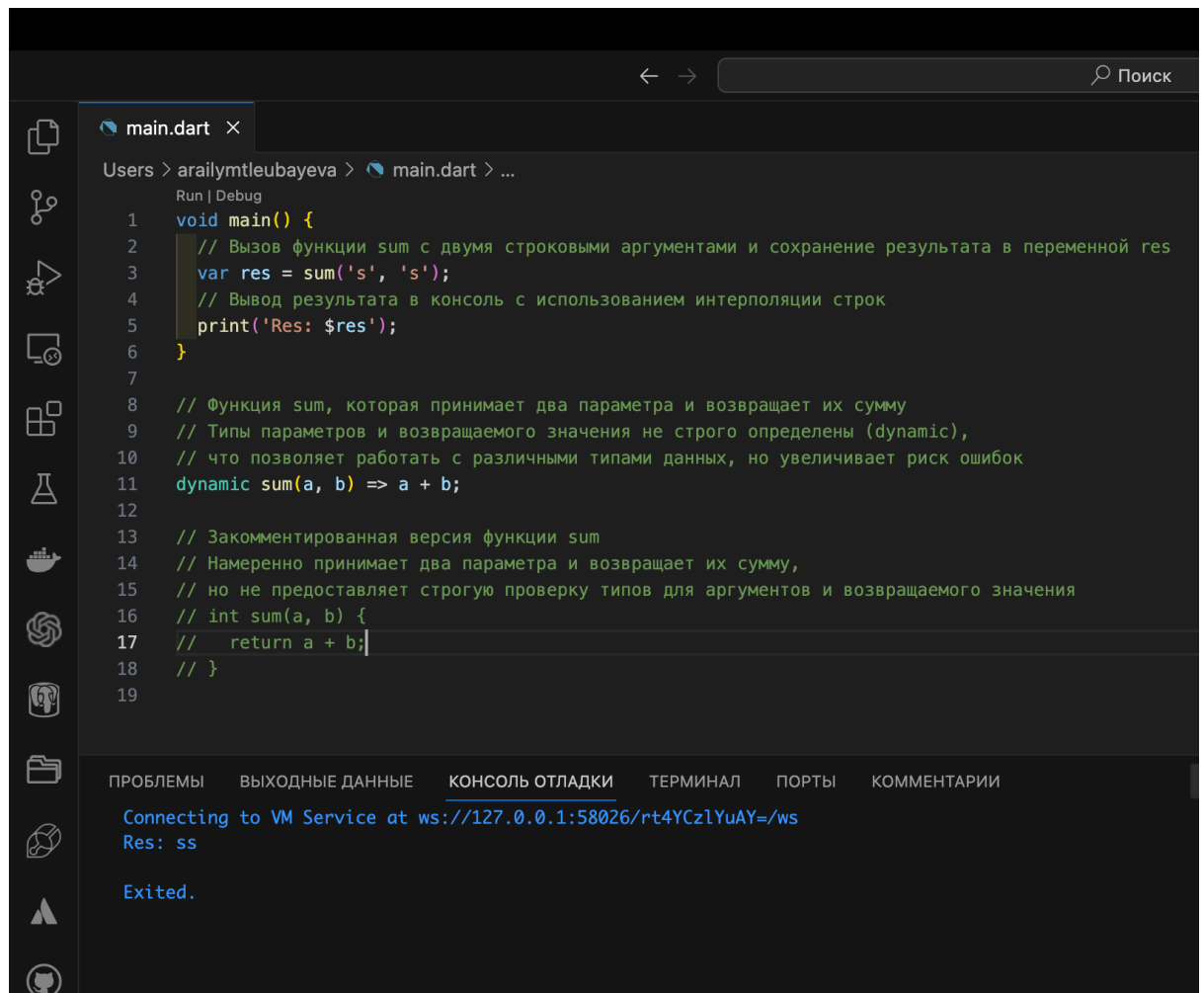
```
1 void main() {  
2   print('First line');  
3   print('Second line');  
4 }
```

Below the editor, the 'Run and Debug' panel is active, showing the execution output in the 'CONSOLE ОТЛАДКИ' (Debug Console) tab. The output text is:

```
Connecting to VM Service at ws://127.0.0.1:62278/DkMYwcWg0Hk=/ws  
First line  
Second line  
  
Exited.
```

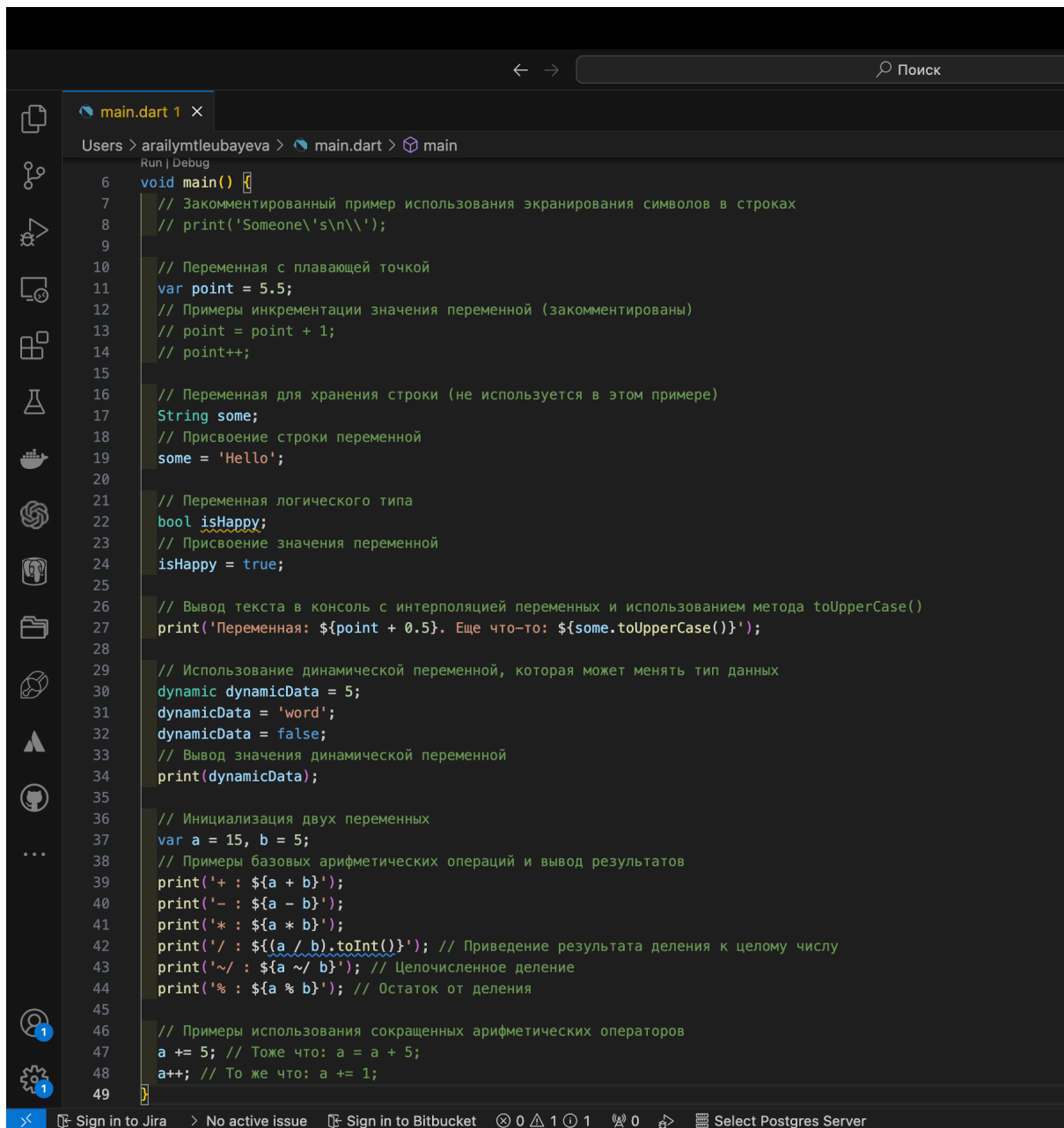
The bottom status bar includes links for 'Sign in to Jira', 'No active issue', 'Sign in to Bitbucket', and resource usage indicators (0 errors, 0 warnings, 0 info), along with a 'Select Pos' button.

3) Функции



```
main.dart x
Users > arailymtleubayeva > main.dart > ...
Run | Debug
1 void main() {
2     // Вызов функции sum с двумя строковыми аргументами и сохранение результата в переменной res
3     var res = sum('s', 's');
4     // Вывод результата в консоль с использованием интерполяции строк
5     print('Res: $res');
6 }
7
8 // Функция sum, которая принимает два параметра и возвращает их сумму
9 // Типы параметров и возвращаемого значения не строго определены (dynamic),
10 // что позволяет работать с различными типами данных, но увеличивает риск ошибок
11 dynamic sum(a, b) => a + b;
12
13 // Закомментированная версия функции sum
14 // Намеренно принимает два параметра и возвращает их сумму,
15 // но не предоставляет строгую проверку типов для аргументов и возвращаемого значения
16 // int sum(a, b) {
17 //     return a + b;
18 // }
19
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ  КОММЕНТАРИИ
Connecting to VM Service at ws://127.0.0.1:58026/rt4YCz1YuAY=/ws
Res: ss
Exited.
```

4) Переменные



The screenshot shows an IDE window with a file named `main.dart`. The code is in Dart and demonstrates various variable types and operations. The interface includes a sidebar with icons for file explorer, search, and other IDE features. The top bar shows the user's name and a search bar. The bottom bar has status information and a button to select a PostgreSQL server.

```
6 void main() {  
7     // Закомментированный пример использования экранирования символов в строках  
8     // print('Someone\'s\n\\');  
9  
10    // Переменная с плавающей точкой  
11    var point = 5.5;  
12    // Примеры инкрементации значения переменной (закомментированы)  
13    // point = point + 1;  
14    // point++;  
15  
16    // Переменная для хранения строки (не используется в этом примере)  
17    String some;  
18    // Присвоение строки переменной  
19    some = 'Hello';  
20  
21    // Переменная логического типа  
22    bool isHappy;  
23    // Присвоение значения переменной  
24    isHappy = true;  
25  
26    // Вывод текста в консоль с интерполяцией переменных и использованием метода toUpperCase()  
27    print('Переменная: ${point + 0.5}. Еще что-то: ${some.toUpperCase()}');  
28  
29    // Использование динамической переменной, которая может менять тип данных  
30    dynamic dynamicData = 5;  
31    dynamicData = 'word';  
32    dynamicData = false;  
33    // Вывод значения динамической переменной  
34    print(dynamicData);  
35  
36    // Инициализация двух переменных  
37    var a = 15, b = 5;  
38    // Примеры базовых арифметических операций и вывод результатов  
39    print('+ : ${a + b}');  
40    print('- : ${a - b}');  
41    print('* : ${a * b}');  
42    print('/ : ${(a / b).toInt()}'); // Приведение результата деления к целому числу  
43    print('~ / : ${a ~/ b}'); // Целочисленное деление  
44    print('% : ${a % b}'); // Остаток от деления  
45  
46    // Примеры использования сокращенных арифметических операторов  
47    a += 5; // То же что: a = a + 5;  
48    a++; // То же что: a += 1;  
49 }
```

ПРОБЛЕМЫ 2

ВЫХОДНЫЕ ДАННЫЕ

КОНСОЛЬ ОТЛАДКИ

ТЕРМИНАЛ

ПОРТЫ

КОММ

Connecting to VM Service at ws://127.0.0.1:58038/esV5u9V5MNM=/ws

Переменная: 6.0. Еще что-то: HELLO

false

+ : 20

- : 10

* : 75

/ : 3

~/ : 3

% : 0

Exited.

5) Условные конструкции

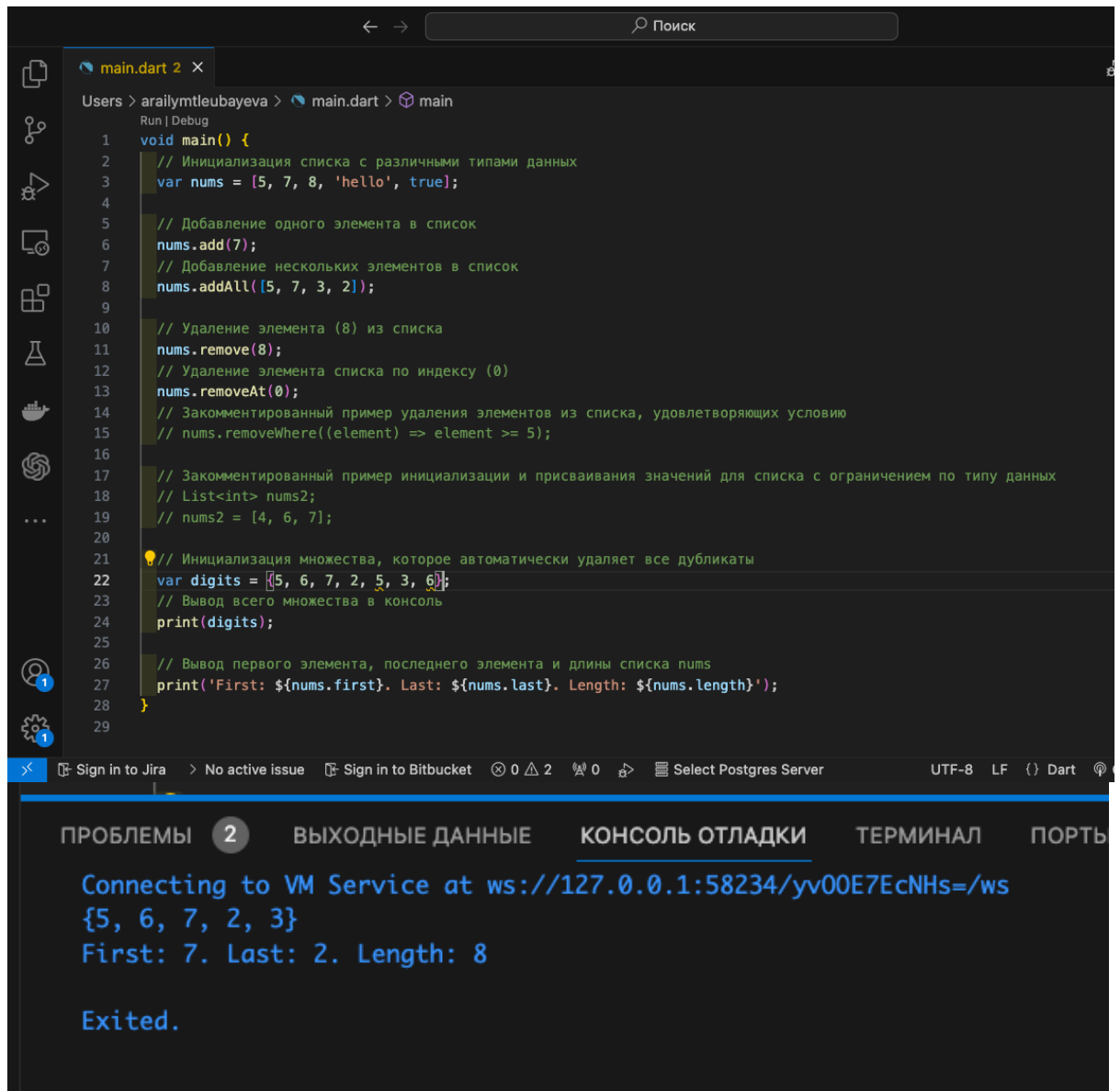
The image shows a screenshot of an IDE with a Dart file named `main.dart`. The code demonstrates various conditional constructs in Dart, including `if-else`, the ternary operator, and a `switch` statement. The code is as follows:

```
1 void main() {  
2     // Инициализация переменных a и b  
3     var a = 15, b = 5;  
4  
5     // Блок условного оператора if-else  
6     // Проверяет условия и выводит соответствующие сообщения  
7     if (a == b || b > 3) { // Если a равно b или b больше 3  
8         print('hello');  
9     } else if (a > b) { // Если предыдущие условия ложные и a больше b  
10        print('a > b');  
11    } else if (a < b) { // Если все предыдущие условия ложные и a меньше b  
12        print('a < b');  
13    } else { // Если все вышестоящие условия ложные  
14        print('else');  
15    }  
16  
17    // Использование тернарного оператора для сокращения блока if-else  
18    // Если a равно 5, переменной res присваивается значение 10, иначе - 20  
19    var res = a == 5 ? 10 : 20;  
20  
21    // Задаем переменную digit  
22    var digit = 5;  
23  
24    // Блок оператора switch, который проверяет значение переменной digit  
25    // и выводит сообщение, соответствующее значению  
26    switch (digit) {  
27        case 4: // Если digit равно 4  
28            print('Equal 4');  
29            break; // Завершаем выполнение блока case  
30        case 5: // Если digit равно 5  
31            print('Equal 5');  
32            break; // Завершаем выполнение блока case  
33        case 7: // Если digit равно 7  
34            print('Equal 7');  
35            break; // Завершаем выполнение блока case  
36        default: // Если ни одно из условий не выполнено  
37            print('Number is unknown');  
38    }  
39 }  
40
```

The output in the console at the bottom shows the execution results:

```
Connecting to VM Service at ws://127.0.0.1:64060/kNuacWW3K6U=/ws  
hello  
Equal 5  
  
Exited.
```

6) Списки данных



The screenshot shows an IDE with a Dart file named `main.dart`. The code defines a `main` function that demonstrates various list operations. Comments in Russian explain each step: initializing a list with mixed types, adding elements, removing elements by value and index, and using `removeWhere` and `removeAt`. It also shows how to initialize a list with a type constraint (`List<int>`) and how to use a set (`Set`) to remove duplicates. The console output at the bottom shows the execution results: the initial list `{5, 6, 7, 2, 3}`, the state after operations `First: 7. Last: 2. Length: 8`, and the final `Exited.` message.

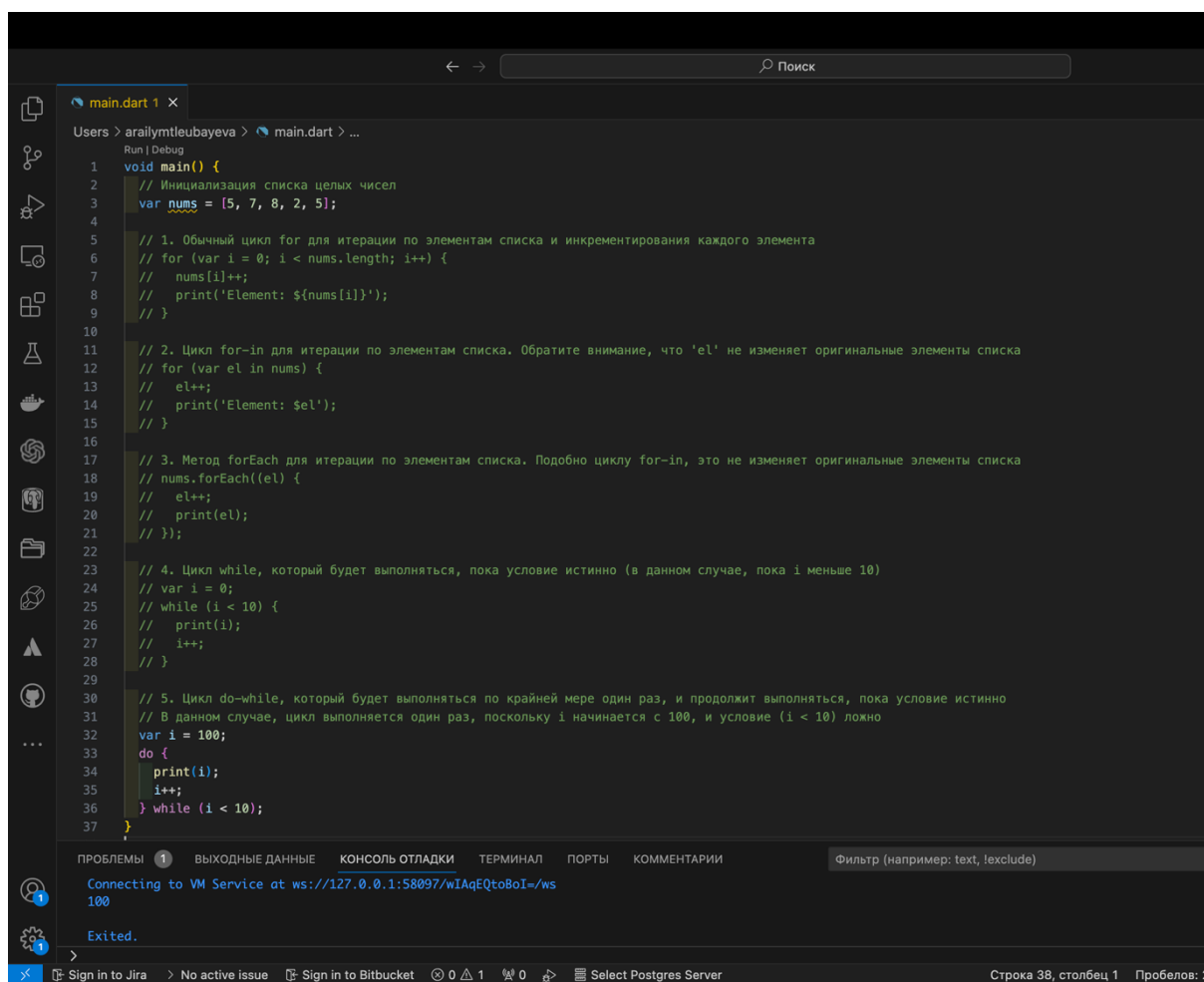
```
1 void main() {  
2   // Инициализация списка с различными типами данных  
3   var nums = [5, 7, 8, 'hello', true];  
4  
5   // Добавление одного элемента в список  
6   nums.add(7);  
7   // Добавление нескольких элементов в список  
8   nums.addAll([5, 7, 3, 2]);  
9  
10  // Удаление элемента (8) из списка  
11  nums.remove(8);  
12  // Удаление элемента списка по индексу (0)  
13  nums.removeAt(0);  
14  // Закомментированный пример удаления элементов из списка, удовлетворяющих условию  
15  // nums.removeWhere((element) => element >= 5);  
16  
17  // Закомментированный пример инициализации и присваивания значений для списка с ограничением по типу данных  
18  // List<int> nums2;  
19  // nums2 = [4, 6, 7];  
20  
21  // Инициализация множества, которое автоматически удаляет все дубликаты  
22  var digits = {5, 6, 7, 2, 5, 3, 6};  
23  // Вывод всего множества в консоль  
24  print(digits);  
25  
26  // Вывод первого элемента, последнего элемента и длины списка nums  
27  print('First: ${nums.first}. Last: ${nums.last}. Length: ${nums.length}');  
28 }  
29
```

PROБЛЕМЫ 2 Выходные данные КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

Connecting to VM Service at ws://127.0.0.1:58234/yv00E7EcNHs=ws
{5, 6, 7, 2, 3}
First: 7. Last: 2. Length: 8

Exited.

7) Циклы



```
1 void main() {  
2   // Инициализация списка целых чисел  
3   var nums = [5, 7, 8, 2, 5];  
4  
5   // 1. Обычный цикл for для итерации по элементам списка и инкрементирования каждого элемента  
6   // for (var i = 0; i < nums.length; i++) {  
7   //   nums[i]++;  
8   //   print('Element: ${nums[i]}');  
9   // }  
10  
11  // 2. Цикл for-in для итерации по элементам списка. Обратите внимание, что 'el' не изменяет оригинальные элементы списка  
12  // for (var el in nums) {  
13  //   el++;  
14  //   print('Element: $el');  
15  // }  
16  
17  // 3. Метод forEach для итерации по элементам списка. Подобно циклу for-in, это не изменяет оригинальные элементы списка  
18  // nums.forEach((el) {  
19  //   el++;  
20  //   print(el);  
21  // });  
22  
23  // 4. Цикл while, который будет выполняться, пока условие истинно (в данном случае, пока i меньше 10)  
24  // var i = 0;  
25  // while (i < 10) {  
26  //   print(i);  
27  //   i++;  
28  // }  
29  
30  // 5. Цикл do-while, который будет выполняться по крайней мере один раз, и продолжит выполняться, пока условие истинно  
31  // В данном случае, цикл выполняется один раз, поскольку i начинается с 100, и условие (i < 10) ложно  
32  var i = 100;  
33  do {  
34    print(i);  
35    i++;  
36  } while (i < 10);  
37 }
```

Самостоятельное задание 1

1) Создание переменных

Создайте несколько переменных для хранения таких значений, как: -34, 4, 'R', 23.093433, true. Продумайте типы данных для переменных, чтобы они максимально подходили под каждое значение.

Важно: не используйте тип данных var для создания переменных!

2) Вывести четное число

Из двух чисел с разной четностью вывести на экран четное число. a, b - данные числа.

3) Создание калькулятора

Создайте две переменные со значением 5 и 19. Создайте еще одну переменную со значением одного арифметического действия: +, -, *, /.

В зависимости от символа, который будет во второй переменной, выполните математические действия над числами.

Для проверки данных используйте оператор switch case

4) Сумма чисел

Найдите сумму $1+2+3+\dots+n$, где число n записано в отдельной переменной.

5) Сумма чисел

Найдите сумму отрицательных элементов массива.

Массив:

`[-24, 34, -4, 4, 5, -1]`

6) Сравнение чисел

У вас есть два числа: 5 и 8.

Создайте функцию, что сравнивает числа и возвращает результат в виде знаков $>$, $<$ или $=$.

[Посмотреть ответ](#)

- 1) Отработать все примеры , обязательно скриншот кода и результата с решением задач в комментариях!**
- 2) Подготовьте отчет(Word файл) по самостоятельной работе с скриншотами исходного кода и результата компиляции.**
- 3) На скриншоте пусть будет видно что это ваша работа (Создайте файл dart с вашим именем).**
- 4) Исходные файлы с кодом загрузите в репозиторий GitHub и оставьте ссылку в отчете. Создайте внутри репозитория папку с названием Практика1Flutter**
- 5) Дедлайн до 19 октября**