Dart, как возможность перенести нативное приложение в **WEB**


Can you help me debug this page?
I wrote it in Dart.

План доклада

- Попытки перенести нативные приложения в веб
- Почему мы выбрали Dart
- Что такое Dart?
- Особенности Dart
- Экосистема Dart
- Сложности перехода с C++ на веб разработку с Dart
- Выводы

Кто я

инфа о себе //TODO

1

Попытки перенести нативное приложение в веб

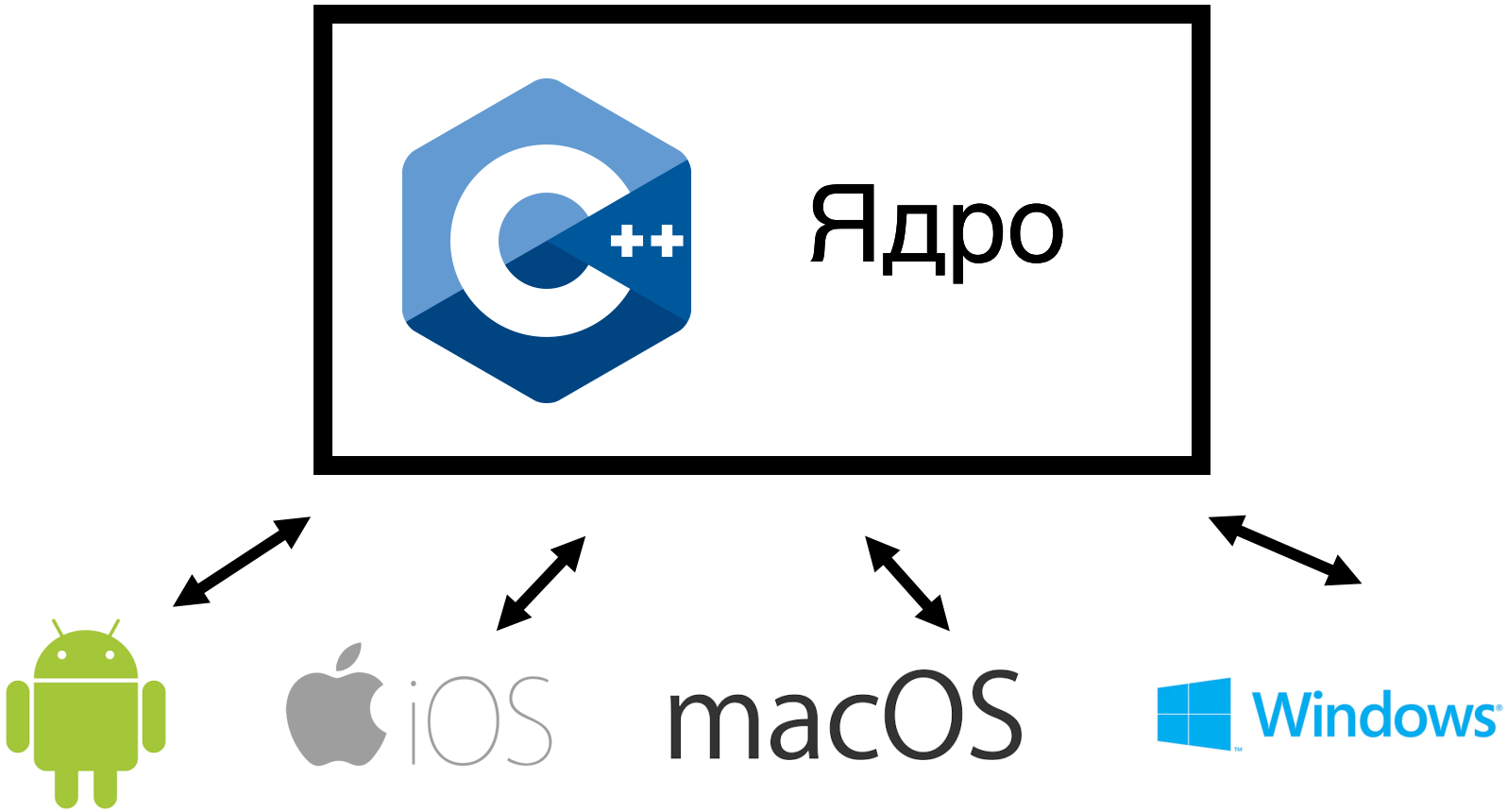
KeepSolid Sign

Архитектура

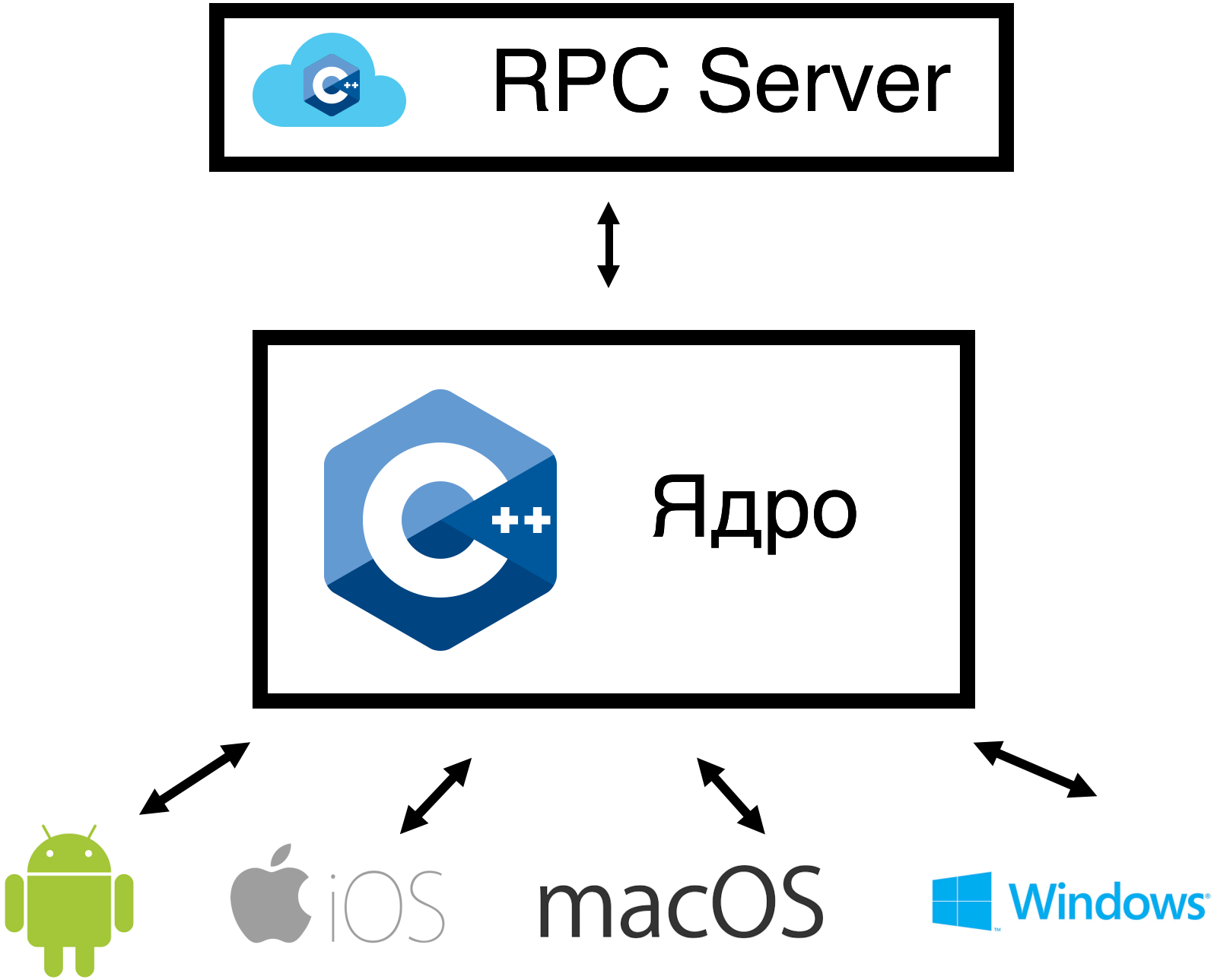
Архитектура

macOS

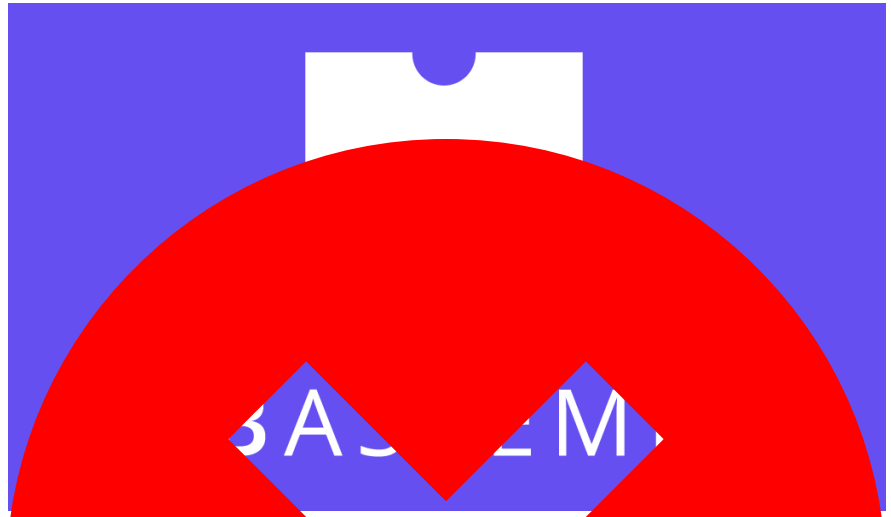
Архитектура



Архитектура



emscripten



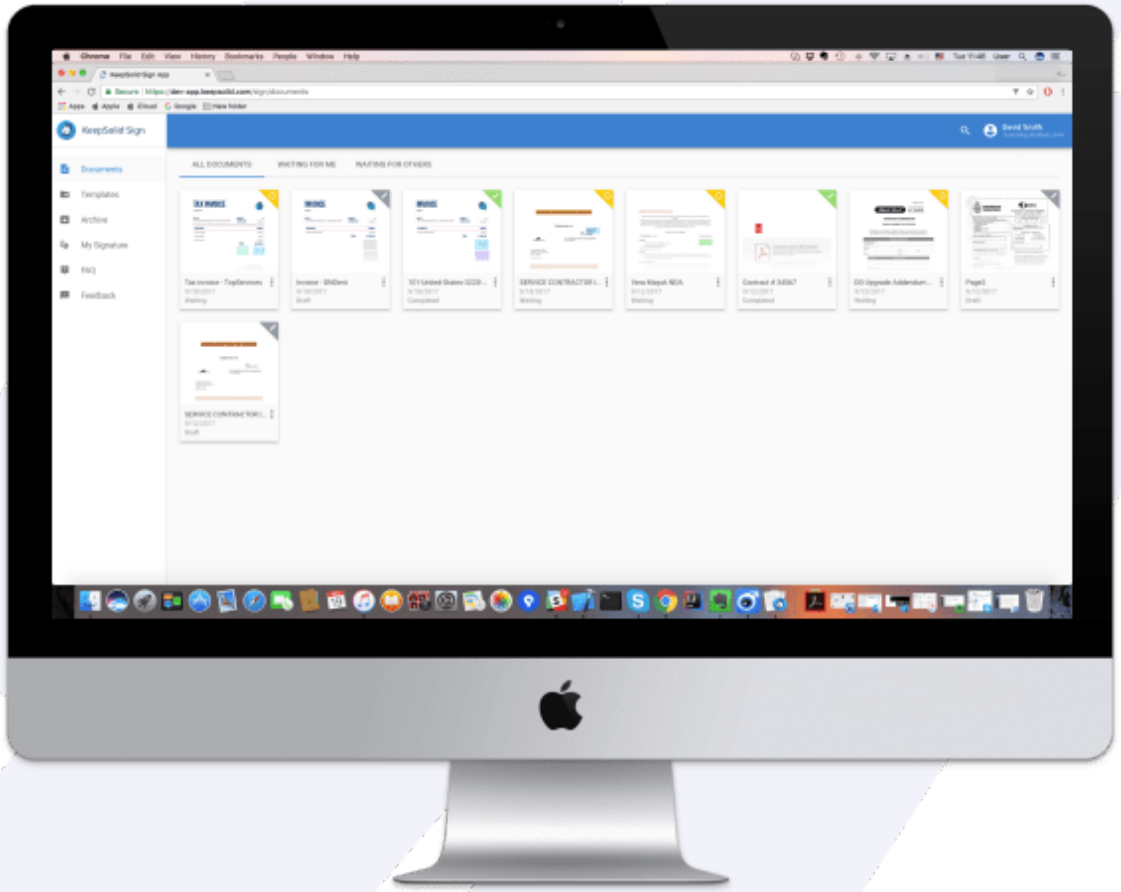
- Монолитное C++ ядро

- Монолитное C++ ядро
- Отсутствие помощи

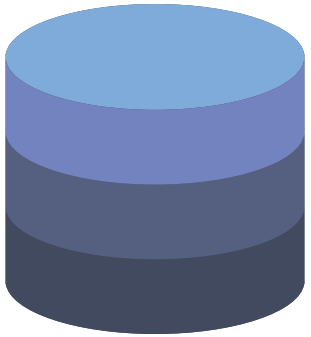
- Монолитное C++ ядро
- Отсутствие помощи
- Некомпилируемые библиотеки


SEVERAL
MONTHS
LATER...

WEB-Версия


KeepSolid Sign
ALL DOCUMENTS
WAITING FOR ME
WAITING FOR OTHERS

Общение с RPC-сервером

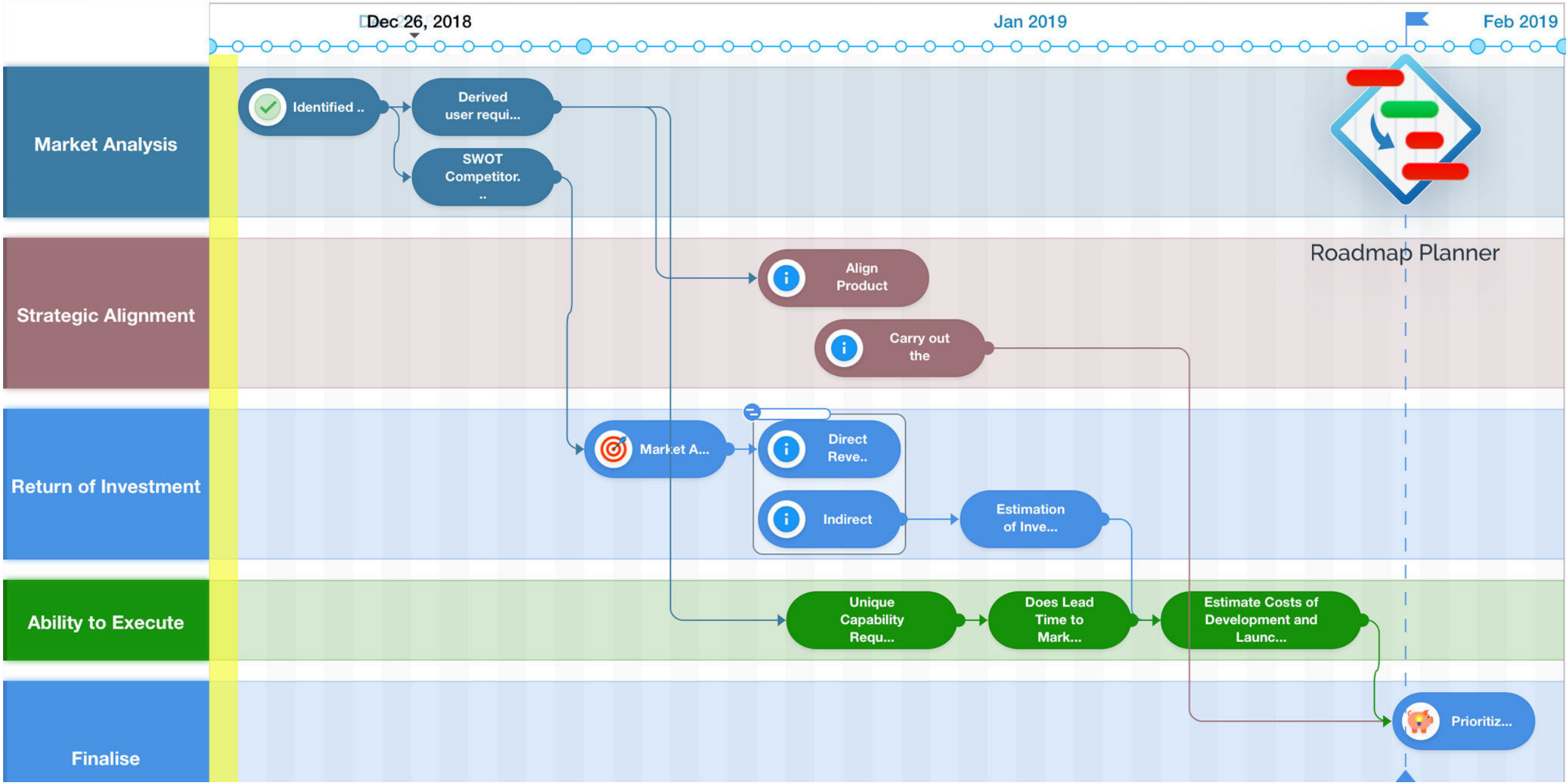


Offline

Криптография

Работа с PDF


TS


Dec 26, 2018
Jan 2019
Feb 2019
Market Analysis
Strategic Alignment
Return of Investment
Ability to Execute
Finalise
Identified ..
Derived user requi...
SWOT Competitor. ..
Roadmap Planner
Align Product
Carry out the
Market A...
Direct Reve..
Indirect
Estimation of Inve...
Unique Capability Requ...
Does Lead Time to Mark...
Estimate Costs of Development and Launc...
Prioritiz...

- Говнокод + утечки памяти

- Говнокод + утечки памяти
- Сложный дебаг

- Говнокод + утечки памяти
- Сложный дебаг
- CMake


10
01

Нужно писать на TypeScript

Нужно писать на TypeScript

?

Research: на чем можно писать Front-End, кроме TypeScript

- Язык от JetBrains
- Много информации
- Сыроват для Web
- Нет UI-фреймворка
- Нет пакетного менеджера

- Синтаксис OCaml от Facebook
- Мало информации
- Есть UI-framework
- Есть пакетный манеджер
- Много функциональщины

- Язык от Mozilla
- Компилируется в WebAssembly, может общаться с JS кодом
- Нет стабильного UI-фреймворка
- Есть пакетный менеджер

Dart

- Язык от Google
- Не много, но и не мало информации
- Есть UI-фреймворк
- Есть пакетный менеджер

Наш выбор

Dart

2

Почему мы выбрали Dart?


JS
TS

- Огромная экосистема (все быстро развивается и устаревает)
- Свой ООП (Прототипноориентрованный)
- Динамическая слабая типизация
- В свое время был написан на коленке, сейчас совершенствуется TC39
- ...

```js
class SomeClass {
  sayHello(value) {
    alert(value);
  }
  _callPrivate(obj) {
    alert(obj.value)
  }
}
```


```ts
interface ISomeClassMain {
  public sayHello: (value: string) => void;
  private callPrivate: (obj: ISomeObject)
                    => void;
}

interface ISomeObject {
  value: 'one' | 'two' | null | undefined;
  zalepon: any;
}

class SomeClass implements ISomeClassMain {
  sayHello(value: string) {
    alert(value);
  }
  private callPrivate(obj: ISomeObject): void
    alert(obj.value);
  }
}
```

Basic Types

Variable Declarations

Interfaces

Classes

Functions

Generics

Enums

Type Inference

Type Compatibility

Advanced Types

Symbols

Вывод ошибок

3

Что такое Dart?

A client-optimized language for fast apps on any platform

“ *Google*

Кто использует

2011

Появился

2011

InfoWorld
FROM IDG

Google launches Dart as a JavaScript killer

The new Dart programming language can be used for both small Web projects as well as large ones, company engineers said

By Joab Jackson
U.S. Correspondent, IDG News Service | OCT 10, 2011

Google has launched a preview version of a new Web programming language, called Dart, which the company's engineers hope will address some of the shortcomings of the widely used JavaScript language.

Появился

2011

Появился

2011

медленно развивался

Появился

2011

2018

медленно развивался

Worst Programming Languages to Learn in 2018 Rankings

Ranked from Worst to Best Languages to Learn

	Overall Rankings	Community Engagement	Job Market	Growth and Trends
1	Dart	Dart	Dart	Objective-C
2	Objective-C	CoffeeScript	Rust	CoffeeScript
3	CoffeeScript	Objective-C	Elm	Dart

https://www.codementor.io/blog/worst-languages-to-learn-3phycr98zk

https://www.codementor.io/blog/worst-languages-to-learn-3phycr98zk

https://www.codementor.io/blog/worst-languages-to-learn-3phycr98zk

https://www.codementor.io/blog/worst-languages-to-learn-3phycr98zk

https://www.codementor.io/blog/worst-languages-to-learn-3phycr98zk

С чего мы начали?

DartPad New Pad... Reset... Format Fibonacci • Share... Samples

DART ▶ Run

```dart
void main() {
  var i = 20;
  print('fibonacci($i) = ${fibonacci(i)}');
}

/// Computes the nth Fibonacci number.
int fibonacci(int n) {
  return n < 2 ? n : (fibonacci(n - 1) + fibonacci(n - 2));
}
```

CONSOLE

```
fibonacci(20) = 6765
```

SDK

IDE

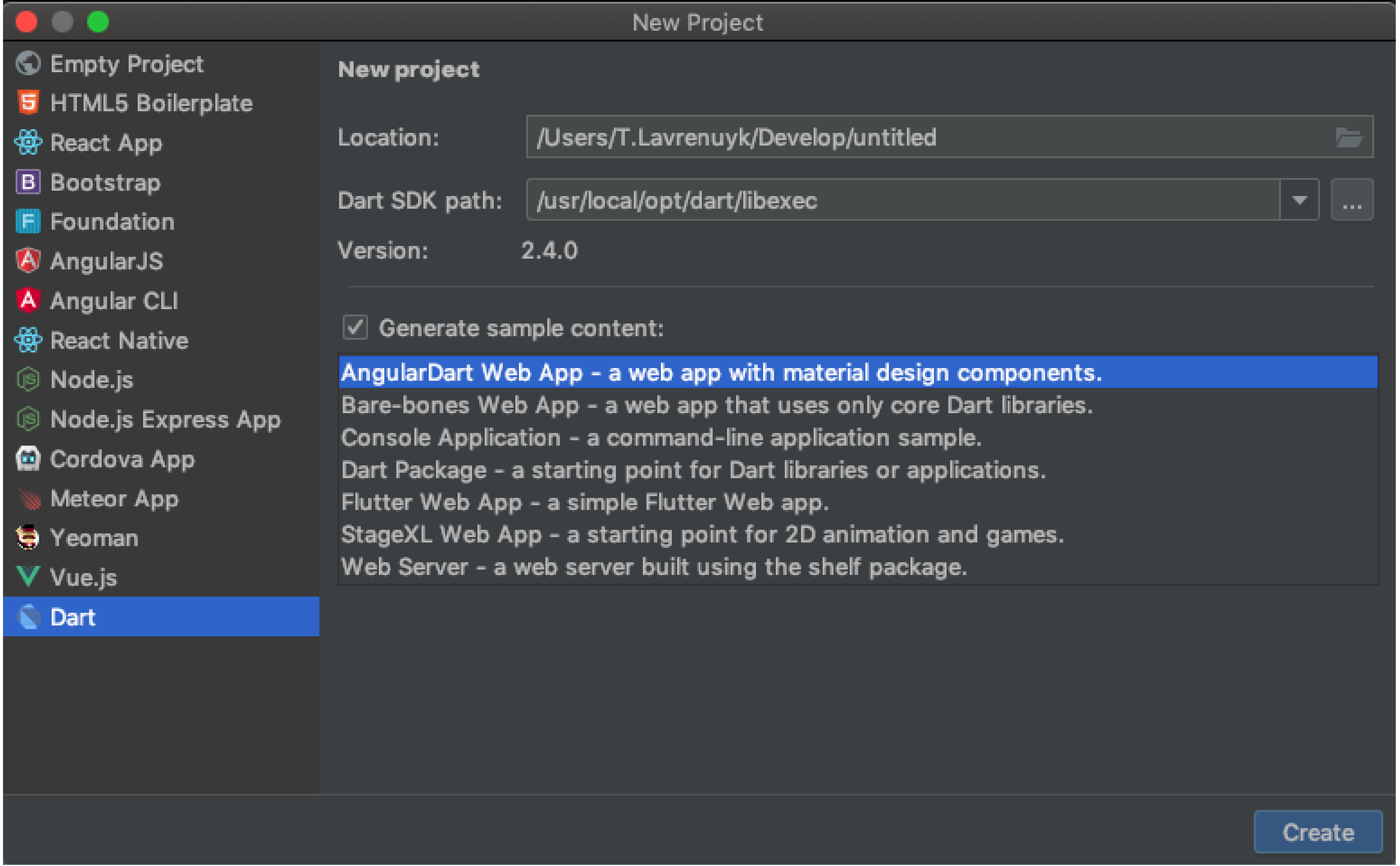
Java

JS


WS

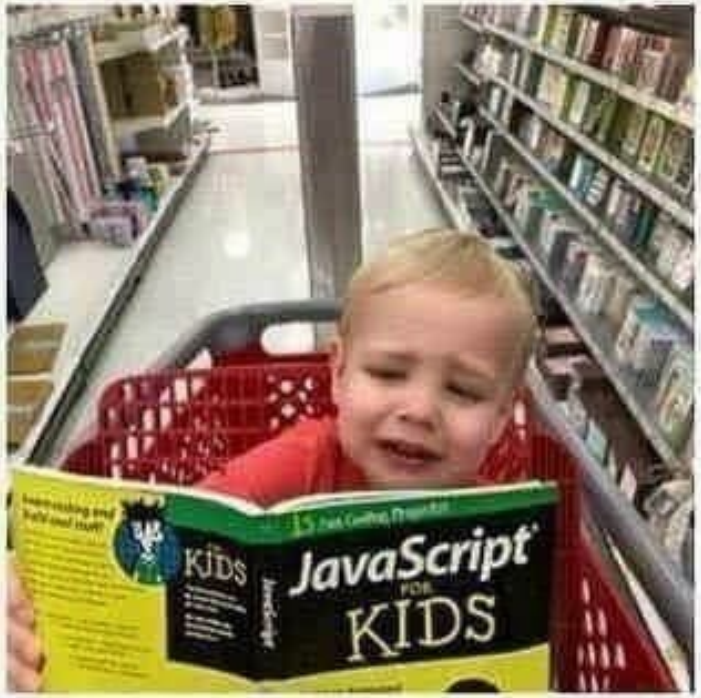

СОЛНЦЕ
НЕЙТРОННАЯ ЗВЕЗДА
ЧЕРНАЯ ДЫРА
САМЫЕ ТЯЖЕЛЫЕ ОБЪЕКТЫ ВО ВСЕЛЕННОЙ

Победил


New Project
Empty Project
HTML5 Boilerplate
React App
Bootstrap
Foundation
AngularJS
Angular CLI
React Native
Node.js
Node.js Express App
Cordova App
Meteor App
Yeoman
Vue.js
Dart
New project
Location:
/Users/T.Lavrenuyk/Develop/untitled
Dart SDK path:
/usr/local/opt/dart/libexec
...
Version:
2.4.0
Generate sample content:
AngularDart Web App - a web app with material design components.
Bare-bones Web App - a web app that uses only core Dart libraries.
Console Application - a command-line application sample.
Dart Package - a starting point for Dart libraries or applications.
Flutter Web App - a simple Flutter Web app.
StageXL Web App - a starting point for 2D animation and games.
Web Server - a web server built using the shelf package.
Create

```
1 void main() {
2   print('Hello, World!');
3 }
```

Как мы смогли выучить язык?



Как учат JS?


JAVASCRIPT.RU

JAVASCRIPT.RU

JavaScript Style Guide

JavaScript Style Guide

MDN MOZILLA DEVELOPER NETWORK

JavaScript Style Guide

MDN MOZILLA DEVELOPER NETWORK


ata Types Arithmetic If Sv
ernary Operator Arrays Fo
oop Do While Loop User In
tring
ecursion File I/O Exceptio
lass Objects Private Publi
rototypes Static Encapsul
onstructors Destructors T
LEARN C++
IN ONE VIDEO
2:52 / 35040:04

МНОГО

Что предлагает Dart?

Dart 2.4.0 is live! For more information, see the change log.

Paint your UI to life

with Dart VM's instant **hot reload**

Language

Tour

Type system

Specification

Effective Dart

Effective Dart

Overview

Style

Documentation

Usage

Design

Names

Naming is an important part of writing readable, maintainable code. The following best practices can help you achieve that goal.

DO use terms consistently.

Use the same name for the same thing, throughout your code. If a precedent already exists outside your API that users are likely to know, follow that precedent.

good

```
pageCount         // A field.
updatePageCount() // Consistent with pageCount.
toSomething()     // Consistent with Iterable's toList().
asSomething()     // Consistent with List's asMap().
Point             // A familiar concept.
```

bad

```
renumberPages()      // Confusingly different from pageCount.
convertToSomething() // Inconsistent with toX() precedent.
wrappedAsSomething() // Inconsistent with asX() precedent.
Cartesian            // Unfamiliar to most users.
```

Круто, что еще нужно?


ДОКУМЕНТАЦИЯ
Я
Shrinking Teton Glaciers
stackoverflow

4

Особенности Dart

Строгая типизация

Типизация в Javascript (ее нет)

The Dart language is type safe: it uses a combination of static type checking and runtime checks to ensure that a variable's value always matches the variable's static type. Although *types* are mandatory, type *annotations* are optional because of type inference.

Типы данных

Типы данных

- Numbers

Типы данных

- Numbers
- Strings

Типы данных

- Numbers
- Strings
- Booleans

Типы данных

- Numbers
- Strings
- Booleans
- Lists

Типы данных

- Numbers
- Strings
- Booleans
- Lists
- Sets

Типы данных

- Numbers
- Strings
- Booleans
- Lists
- Sets
- Maps

```
List numbers = [1,2,3];

print(numbers.first); // 1
print(numbers.last); // 3

print(numbers.isEmpty); // false
print(numbers.isNotEmpty); // true
```

```
Map<String, String> users =
        {"1": "Foo", "2": "Bar"};

List<String> usernames = users.values;
```

```dart
class SomeClass {
  double number;
  callFunction() {}
}
SomeClass nullClass;

final value = nullClass ?? SomeClass()
final number ??= SomeClass().number;

print(nullClass?.callFunction());
```

```dart
class SomeClass {
  double number;
  callFunction() {}
}
SomeClass nullClass;

final value = nullClass ?? SomeClass();
final number ??= SomeClass().number;

print(nullClass?.callFunction());
```

```dart
class SomeClass {
  double number;
  callFunction() {}
}
SomeClass nullClass;

final value = nullClass ?? SomeClass()
final number ??= SomeClass().number;

print(nullClass?.callFunction());
```

```dart
class SomeClass {
  double number;
  callFunction() {}
}
SomeClass nullClass;

final value = nullClass ?? SomeClass()
final number ??= SomeClass().number;

print(nullClass?.callFunction());
```

```
1  class Foo {
2      Foo({
3          this.paramOneName = '1',
4          this.paramTwoName = '2' });
5
6      String paramOneName;
7      String paramTwoName;
8  }
9
10 var foo = Foo(paramOneName: '3');
```

```
1  class Foo {
2      Foo({
3          this.paramOneName = '1',
4          this.paramTwoName = '2' });
5
6      String paramOneName;
7      String paramTwoName;
8  }
9
10 var foo = Foo(paramOneName: '3');
```

```dart
class Bar {
    Bar();

    Foo f1;
    Foo f2;
}

Bar bar = Bar()
    ..f1 = Foo(paramTwoName: '2')
    ..f2 = Foo(paramOneName: '10');
```

```
1  class Bar {
2      Bar();
3
4      Foo f1;
5      Foo f2;
6  }
7
8  Bar bar = Bar()
9      ..f1 = Foo(paramTwoName: '2')
10     ..f2 = Foo(paramOneName: '10');
```

```cpp
#include <vector>
class foo
{
  public:
    class builder;
    foo(int prop1, bool prop2, bool prop3, std::vector<int> prop4)
      : prop1{prop1}, prop2{prop2}, prop3{prop3}, prop4{prop4}
    { }
    int prop1;
    bool prop2;
    bool prop3;
    std::vector<int> prop4;
};
class foo::builder
{
  public:
    builder& set_prop1(int value) { prop1 = value; return *this; };
    builder& set_prop2(bool value) { prop2 = value; return *this; };
    builder& set_prop3(bool value) { prop3 = value; return *this; };
    builder& set_prop4(std::vector<int> value) { prop4 = value; return *this; };
    foo build() const
    {
      return foo{prop1, prop2, prop3, prop4};
    }
  private:
    int prop1 = 0;
    bool prop2 = false;
    bool prop3 = false;
    std::vector<int> prop4 = {};
};
int main()
{
  foo f = foo::builder{}.set_prop1(5)
                        .set_prop3(true)
                        .build();
}
```

```
mixin Musical {
  bool canPlayPiano = false;
  bool canCompose = false;
  bool canConduct = false;

  void entertainMe() {
    if (canPlayPiano) {
      print('Playing piano');
    } else if (canConduct) {
      print('Waving hands');
    } else {
      print('Humming to self');
    }
  }
}
```

```
1  class Musician extends Performer with Musical {
2    // ...
3  }
4
5  class Maestro extends Person
6      with Musical, Aggressive, Demented {
7    Maestro(String maestroName) {
8      name = maestroName;
9      canConduct = true;
10   }
11 }
```

```
1  class Musician extends Performer with Musical {
2    // ...
3  }
4
5  class Maestro extends Person
6      with Musical, Aggressive, Demented {
7    Maestro(String maestroName) {
8      name = maestroName;
9      canConduct = true;
10   }
11 }
```

Вспомогательные классы

Вспомогательные классы

- DateTime

Вспомогательные классы

- DateTime
- Duration

Вспомогательные классы

- DateTime
- Duration
- Uri

Core libraries

- dart:core
- dart:async
- dart:collection
- dart:html
- dart:io
- ...

dart:async

```dart
Future<int> future = Future(asyncFunction);
future.then((int value) {
  print(value);
}).catchError((e) {
  print('Error');
});

someFunc () async {
  try {
    int value = await future;
  } catch(e) {
    print('Error');
  }
}
```

```dart
Future<int> future = Future(asyncFunction);
future.then((int value) {
  print(value);
}).catchError((e) {
  print('Error');
});

someFunc () async {
  try {
    int value = await future;
  } catch(e) {
    print('Error');
  }
}
```

Streams

```
1 controller = StreamController<int>(
2   onListen: startTimer,
3   onPause: stopTimer,
4   onResume: startTimer,
5   onCancel: stopTimer);
6
7 controller.stream.listen((timer) {
8   print(timer);
9 });
```

```
controller = StreamController<int>(
  onListen: startTimer,
  onPause: stopTimer,
  onResume: startTimer,
  onCancel: stopTimer);

controller.stream.listen((timer) {
  print(timer);
});
```

```
someStream
   .map(...)
   .reduce(...)
   .skip(...)
   .take(...)
   .timeout(...)
   .transform(...)
   .listen((result) {
      print(result);
   })
```

asyncMapBuffer

Like `asyncMap` but events are buffered in a List until previous events have been processed rather than being called for each element individually.

asyncMapSample

Like `asyncMap` but events are discarded, keeping only the latest, until previous events have been processed rather than being called for every element.

asyncWhere

Like `where` but allows an asynchronous predicate.

audit

Audit waits for a period of time after receiving a value and then only emits the most recent value.

buffer

Collects values from a source stream until a `trigger` stream fires and the collected values are emitted.

combineLatest

Combine the most recent event from two streams through a callback and emit the result.

combineLatestAll

Combines the latest events emitted from multiple source streams and yields a list of the values.

debounce, debounceBuffer

Prevents a source stream from emitting too frequently by dropping or collecting values that occur

stream_transform

Available Methods

- buffer / BufferStreamTransformer
- bufferCount / BufferStreamTransformer / onCount
- bufferTest / BufferStreamTransformer / onTest
- bufferTime / BufferStreamTransformer / onTime
- concatMap (alias for `asyncExpand`)
- concatWith
- debounce / DebounceStreamTransformer
- debounceTime / DebounceStreamTransformer
- delay / DelayStreamTransformer
- dematerialize / DematerializeStreamTransformer
- distinctUnique / DistinctUniqueStreamTransformer
- doOnCancel / DoStreamTransformer
- doOnData / DoStreamTransformer
- doOnDone / DoStreamTransformer
- doOnEach / DoStreamTransformer
- doOnError / DoStreamTransformer
- doOnListen / DoStreamTransformer
- doOnPause / DoStreamTransformer
- doOnResume / DoStreamTransformer
- exhaustMap / ExhaustMapStreamTransformer
- flatMap / FlatMapStreamTransformer
- flatMapIterable
- groupBy / GroupByStreamTransformer
- interval / IntervalStreamTransformer
- mapTo / MapToStreamTransformer
- materialize / MaterializeStreamTransformer
- mergeWith
- max / StreamMaxFuture
- min / StreamMinFuture
- onErrorResume / OnErrorResumeStreamTransformer
- onErrorResumeNext / OnErrorResumeStreamTransformer
- onErrorReturn / OnErrorResumeStreamTransformer
- onErrorReturnWith / OnErrorResumeStreamTransformer
- sample / SampleStreamTransformer

rx_dart

dart:collection

Classes

DoubleLinkedQueue<E>
A Queue implementation based on a double-linked list. [...]

DoubleLinkedQueueEntry<E>
An entry in a doubly linked list. It contains a pointer to the next entry, the previous entry, and the boxed element.

HashMap<K, V>
A hash-table based implementation of Map. [...]

HashSet<E>
An unordered hash-table based Set implementation. [...]

HasNextIterator<E>
The HasNextIterator class wraps an Iterator and provides methods to iterate over an object using `hasNext` and `next`. [...]

IterableBase<E>
Base class for implementing Iterable. [...]

IterableMixin<E>
This Iterable mixin implements all Iterable members except `iterator`. [...]

LinkedHashMap<K, V>
A hash-table based implementation of Map. [...]

LinkedHashSet<E>
A LinkedHashSet is a hash-table based Set implementation. [...]

LinkedList<E extends LinkedListEntry<E>>
A specialized double-linked list of elements that extends LinkedListEntry. [...]

LinkedListEntry<E extends LinkedListEntry<E>>
An object that can be an element in a LinkedList. [...]

ListBase<E>
Abstract implementation of a list. [...]

dart:convert

- Base64Encoder
- Base64decoder
- JsonDecoder
- JsonEncoder
- HtmlEscape
- ...

dart:html

```
var button = querySelector('#confirm');
button.text = 'Confirm';
button.classes.add('important');
button.onClick.listen((e) => window.alert('Confirmed!'));
```

```
final addressBook = (AddressBookBuilder()
        ..name = 'jenny'
        ..email = 'jenny@example.com'
        ..phone = (PhoneNumberBuilder()

var button = querySelector('#confirm');
button.text = 'Confirm';
button.classes.add('important');
button.onClick.listen((e) => window.alert('Confirmed!'));
```

Stream

```
final addressBook = (AddressBookBuilder()
        ..name = 'jenny'
        ..email = 'jenny@example.com'
        ..phone = (PhoneNumberBuilder()
              ..number = '415-555-0100'
              ..label = 'home')
            .build())
    .build();
```

Все из коробки (SDK)

Сторонние библиотеки

json_serializer

```
1 {
2     "name": "Foo",
3     "email": "somemail@example.com"
4 }
```

JSON response

```dart
import 'package:json_annotation/json_annotation.dart';

part 'user.g.dart';


@JsonSerializable()
class User {
  User(this.name, this.email);

  String name;
  String email;

  factory User.fromJson(Map<String, dynamic> json) => _$UserFromJson(json);
  Map<String, dynamic> toJson() => _$UserToJson(this);
}
```

built_value

built_collection

```dart
abstract class User implements Built<User, UserBuilder> {

  factory User([updates(UserBuilder b)]) = _$User;
  User._();

  @nullable
  String get firstName;

  @nullable
  String get lastName;
}
```

```
User user = User((builder) =>
    builder..firstName = 'One'
           ..lastName = 'Two'
);

User secondUser = user.rebuild((builder) =>
      builder..lastName = 'Three'
);

// user != secondUser
```

intl

http

5

Экосистема Dart

UI

Веб-разработка

2009

2019

```dart
import 'package:angular/angular.dart';

@Component(
    selector: 'user-info',
    templateUrl: 'user_info.html')
class UserInfo implements OnInit {
    User(this.dataController)

    DataController dataController;

    Stream<String> userName;

    void ngOnInit() {
        userName = dataController.info.userName;
    }
}
```

```dart
import 'package:angular/angular.dart';

@Component(
    selector: 'user-info',
    templateUrl: 'user_info.html')
class UserInfo implements OnInit {
    User(this.dataController)

    DataController dataController;

    Stream<String> userName;

    void ngOnInit() {
        userName = dataController.info.userName;
    }
}
```

```dart
import 'package:angular/angular.dart';

@Component(
    selector: 'user-info',
    templateUrl: 'user_info.html')
class UserInfo implements OnInit {
    User(this.dataController)

    DataController dataController;

    Stream<String> userName;

    void ngOnInit() {
        userName = dataController.info.userName;
    }
}
```

```dart
import 'package:angular/angular.dart';

@Component(
    selector: 'user-info',
    templateUrl: 'user_info.html')
class UserInfo implements OnInit {
    User(this.dataController)

    DataController dataController;

    Stream<String> userName;

    void ngOnInit() {
        userName = dataController.info.userName;
    }
}
```

```dart
import 'package:angular/angular.dart';

@Component(
    selector: 'user-info',
    templateUrl: 'user_info.html')
class UserInfo implements OnInit {
    User(this.dataController)

    DataController dataController;

    Stream<String> userName;

    void ngOnInit() {
        userName = dataController.info.userName;
    }
}
```

```
<div>{{userName | async}}</div>
```

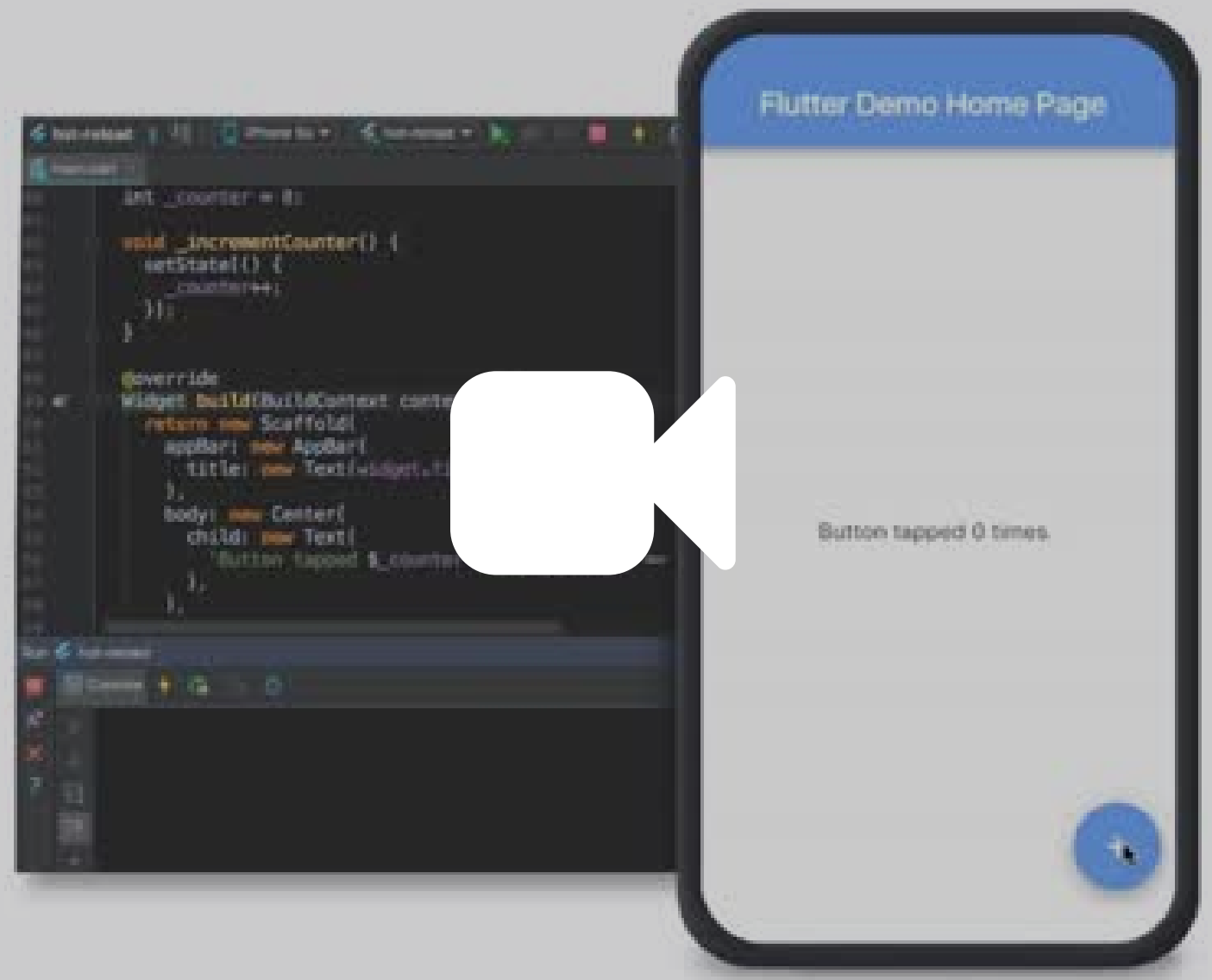
А что если нужна библиотека, которой нет на Dart?

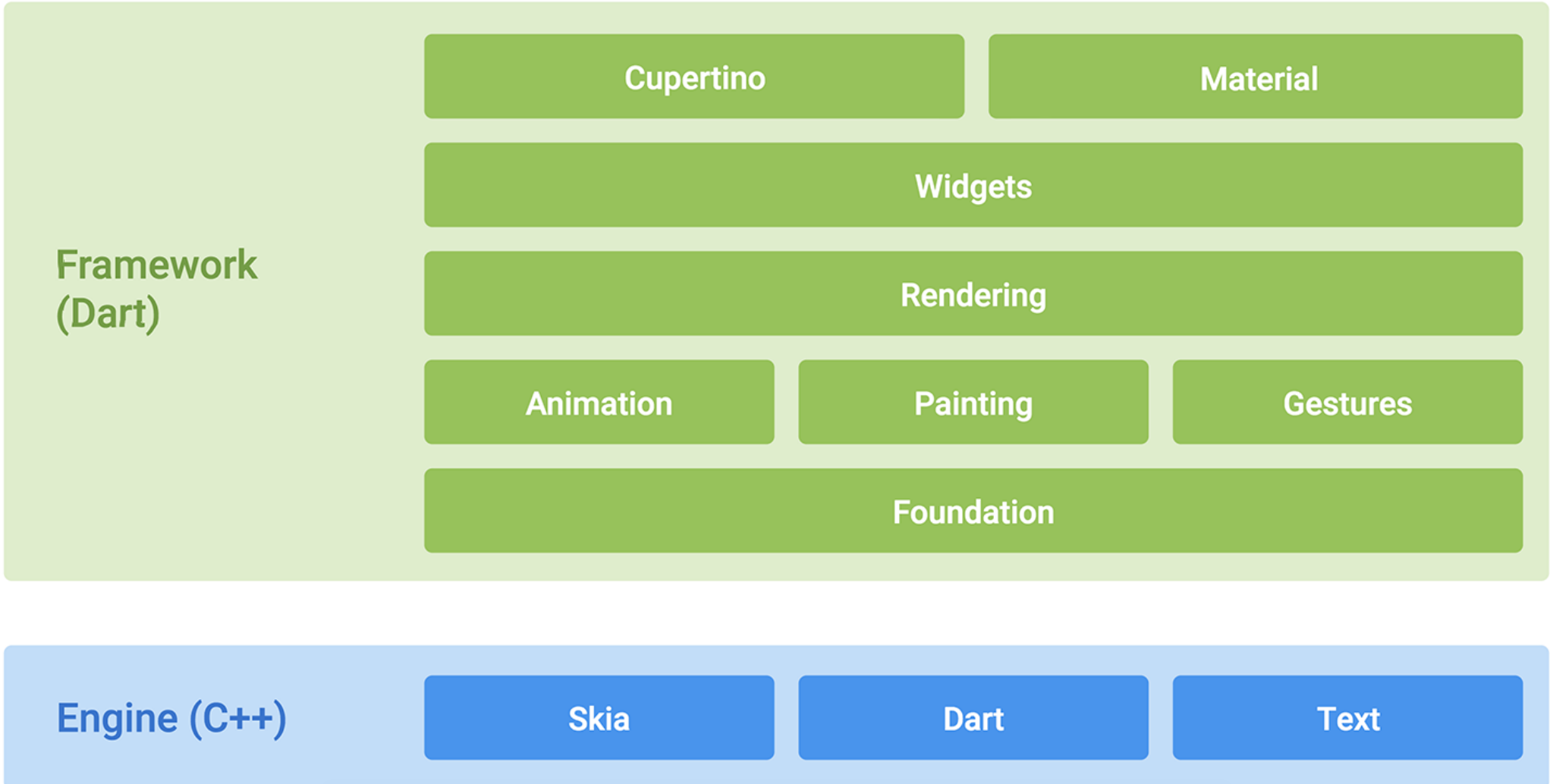
Подключаем JS библиотеку

```
@JS('draw2d')
library draw2d;

@JS('Canvas')
class Canvas {
  external factory Canvas(String name, [num x, num y]);

  external Canvas add(Figure figure, [num x, num y]);
  external Canvas addSelection(ArrayList /*Figure|List<Figure>*/ object);
  external Canvas clear();
  external void destroy();
  external void fireEvent(String name, [dynamic attr]); /*Object*/
  external Point fromCanvasToDocumentCoordinate(num x, num y);
  external Point fromDocumentToCanvasCoordinate(num x, num y);
  external num getAbsoluteX();
  external num getAbsoluteY();
}
```

Кроссплатформенная разработка

Flutter


int _counter = 0;
void _incrementCounter() {
setState() {
_counter++;
});
}
@override
Widget build(BuildContext
return new Scaffold(
appBar: new AppBar(
title: new Text(
),
body: new Center(
child: new Text(
'Button tapped $_counter
),
),
Flutter Demo Home Page
Button tapped 0 times.


Framework
(Dart)
Cupertino
Material
Widgets
Rendering
Animation
Painting
Gestures
Foundation
Engine (C++)
Skia
Dart
Text

Появился

2011

Dart 2.0

2018

медленно развивался

Появился

2011

Dart 2.0

2018

медленно развивался

2019

Появился

2011

медленно развивался

Dart 2.0

2018

2019

Flutter-Хайп

Хайп Flutter-а двигает разработку Dart вперед

Flutter

105 тыс. подписчиков

ГЛАВНАЯ ВИДЕО ПЛЕЙЛИС

Форматирование кода

```
1  class User
2  {
3    String paramNumberOne       = 1;
4    String paramNumbeTwo        = 2;
5    String paramNumbeThree      = 3;
6
7    num age;
8
9    num get age => _age;
10
11   set age (num age)
12   {
13     _age = age;
14   }
15 }
```

```
1 class User {
2   String paramNumberOne = 1;
3   String paramNumbeTwo = 2;
4   String paramNumbeThree = 3;
5
6   num age;
7 }
```

Dart Formatter

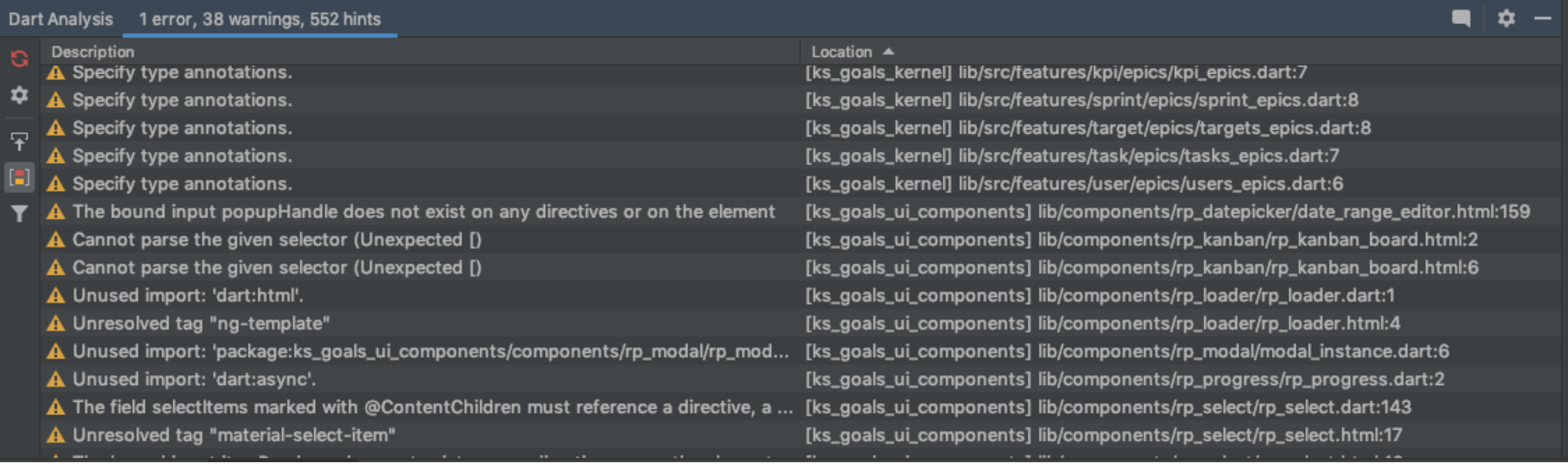
```
dartfmt -w --fix lib
```

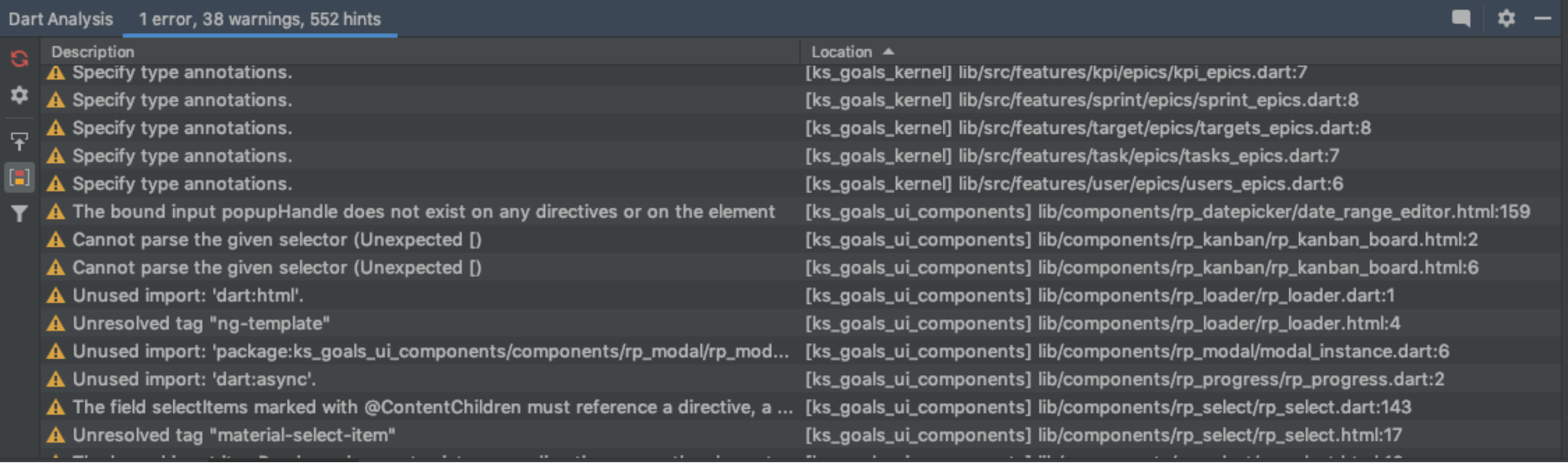
Dart 

Linter

```yaml
# Defines a default set of lint rules enforced for
# projects at Google. For details and rationale,
# see https://github.com/dart-lang/pedantic#enabled-lints.
include: package:pedantic/analysis_options.yaml

# For lint rules and documentation, see http://dart-lang.github.io/linter/lints.
# Uncomment to specify additional rules.
linter:
  rules:
    - always_declare_return_types
    - always_put_control_body_on_new_line
    - always_put_required_named_parameters_first
    - always_require_non_null_named_parameters
    - annotate_overrides
    - avoid_annotating_with_dynamic
    - avoid_bool_literals_in_conditional_expressions
```

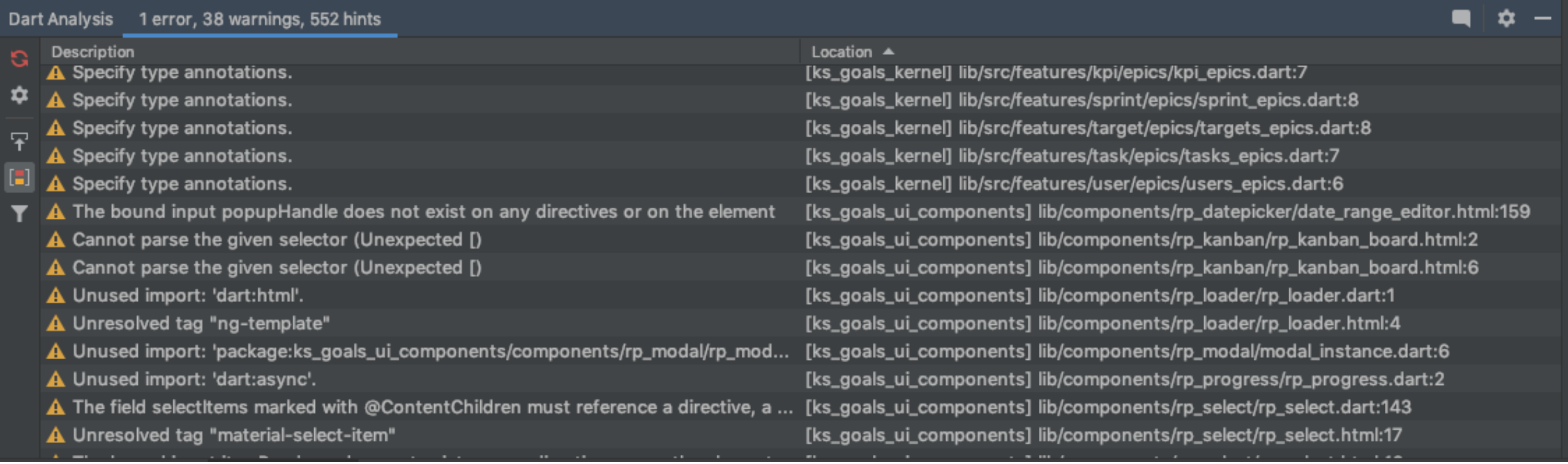
analysis_options.yaml


Dart Analysis
1 error, 38 warnings, 552 hints
Description
Location
Specify type annotations.
[ks_goals_kernel] lib/src/features/kpi/epics/kpi_epics.dart:7
Specify type annotations.
[ks_goals_kernel] lib/src/features/sprint/epics/sprint_epics.dart:8
Specify type annotations.
[ks_goals_kernel] lib/src/features/target/epics/targets_epics.dart:8
Specify type annotations.
[ks_goals_kernel] lib/src/features/task/epics/tasks_epics.dart:7
Specify type annotations.
[ks_goals_kernel] lib/src/features/user/epics/users_epics.dart:6
The bound input popupHandle does not exist on any directives or on the element
[ks_goals_ui_components] lib/components/rp_datepicker/date_range_editor.html:159
Cannot parse the given selector (Unexpected [)
[ks_goals_ui_components] lib/components/rp_kanban/rp_kanban_board.html:2
Cannot parse the given selector (Unexpected [)
[ks_goals_ui_components] lib/components/rp_kanban/rp_kanban_board.html:6
Unused import: 'dart:html'.
[ks_goals_ui_components] lib/components/rp_loader/rp_loader.dart:1
Unresolved tag "ng-template"
[ks_goals_ui_components] lib/components/rp_loader/rp_loader.html:4
Unused import: 'package:ks_goals_ui_components/components/rp_modal/rp_mod...
[ks_goals_ui_components] lib/components/rp_modal/modal_instance.dart:6
Unused import: 'dart:async'.
[ks_goals_ui_components] lib/components/rp_progress/rp_progress.dart:2
The field selectItems marked with @ContentChildren must reference a directive, a ...
[ks_goals_ui_components] lib/components/rp_select/rp_select.dart:143
Unresolved tag "material-select-item"
[ks_goals_ui_components] lib/components/rp_select/rp_select.html:17



+ pre-commit hook

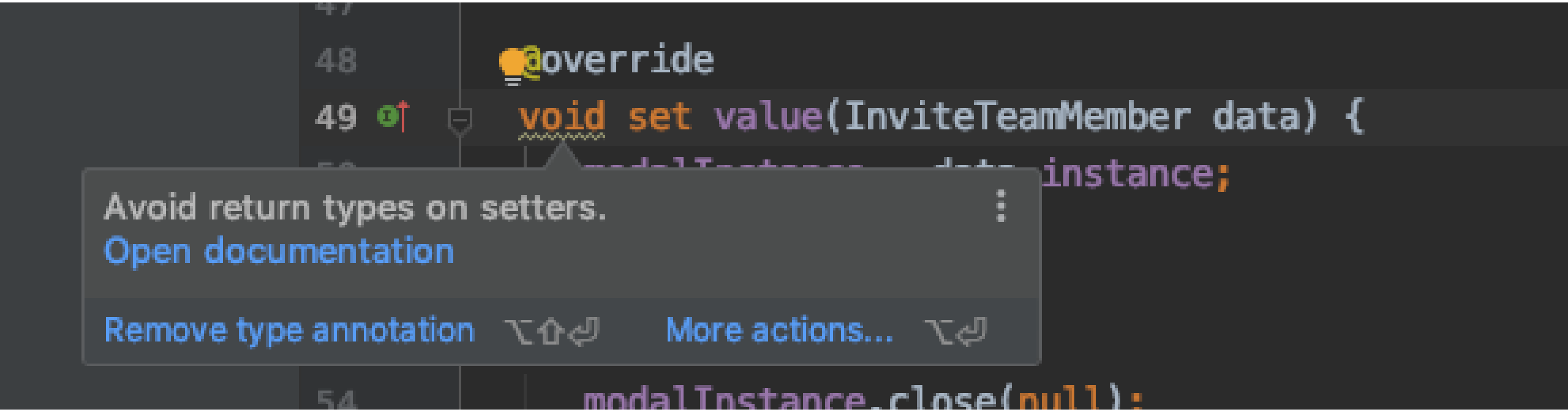
```
1 dartanalyzer --fatal-warnings lib
```



+ pre-commit hook

```
1 dartanalyzer --fatal-warnings lib
```

= 🔥


48
@override
49
void set value(InviteTeamMember data) {
instance;
Avoid return types on setters.
Open documentation
Remove type annotation ⌥⇧⏎
More actions... ⌥⏎
54
modalInstance.close(null);

avoid_return_types_on_setters

Group: style
Maturity: stable
Dart SDK: >= 2.0.0 • (Linter v0.1.11)

style flutter style pedantic

See the Style Guide | List of Lint Rules

AVOID return types on setters.

As setters do not return a value, declaring the return type of one is redundant.

GOOD:

```
set speed(int ms);
```

BAD:

```
void set speed(int ms);
```

Dart 👍👍👍

Документация

PREFER starting library or type comments with noun phrases.

Doc comments for classes are often the most important documentation in your program. They describe the type's invariants, establish the terminology it uses, and provide context to the other doc comments for the class's members. A little extra effort here can make all of the other members simpler to document.

good

```
/// A chunk of non-breaking output text terminated by a hard or soft newline.
///
/// ...
class Chunk { ... }
```

CONSIDER including code samples in doc comments.

good

```
/// Returns the lesser of two numbers.
///
/// ```dart
/// min(5, 3) == 3
/// ```
num min(num a, num b) => ...
```

Dart documentation guide

dartdoc

Observable<T>.concat constructor

```
Observable<T>.concat(
    Iterable<Stream<T>> streams
)
```

Concatenates all of the specified stream sequences, as long as the previous stream sequence terminated successfully.

It does this by subscribing to each stream one by one, emitting all items and completing before subscribing to the next stream.

Interactive marble diagram

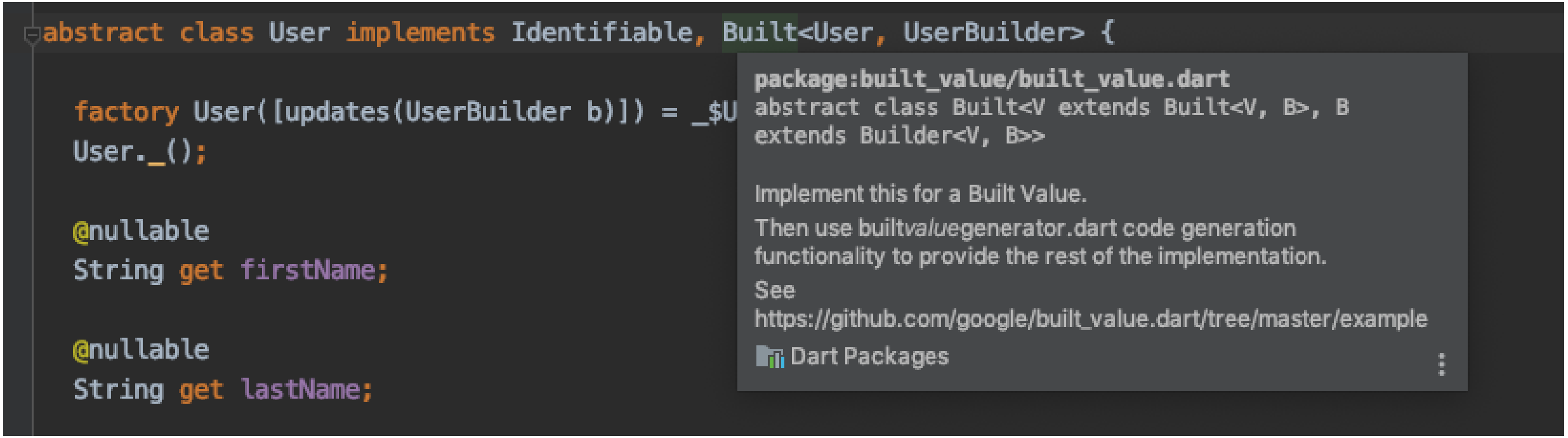
Example

```
new Observable.concat([
  new Observable.just(1),
  new Observable.timer(2, new Duration(days: 1)),
  new Observable.just(3)
])
.listen(print); // prints 1, 2, 3
```

Implementation

```
factory Observable.concat(Iterable<Stream<T>> streams) =>
    Observable<T>(ConcatStream<T>(streams));
```

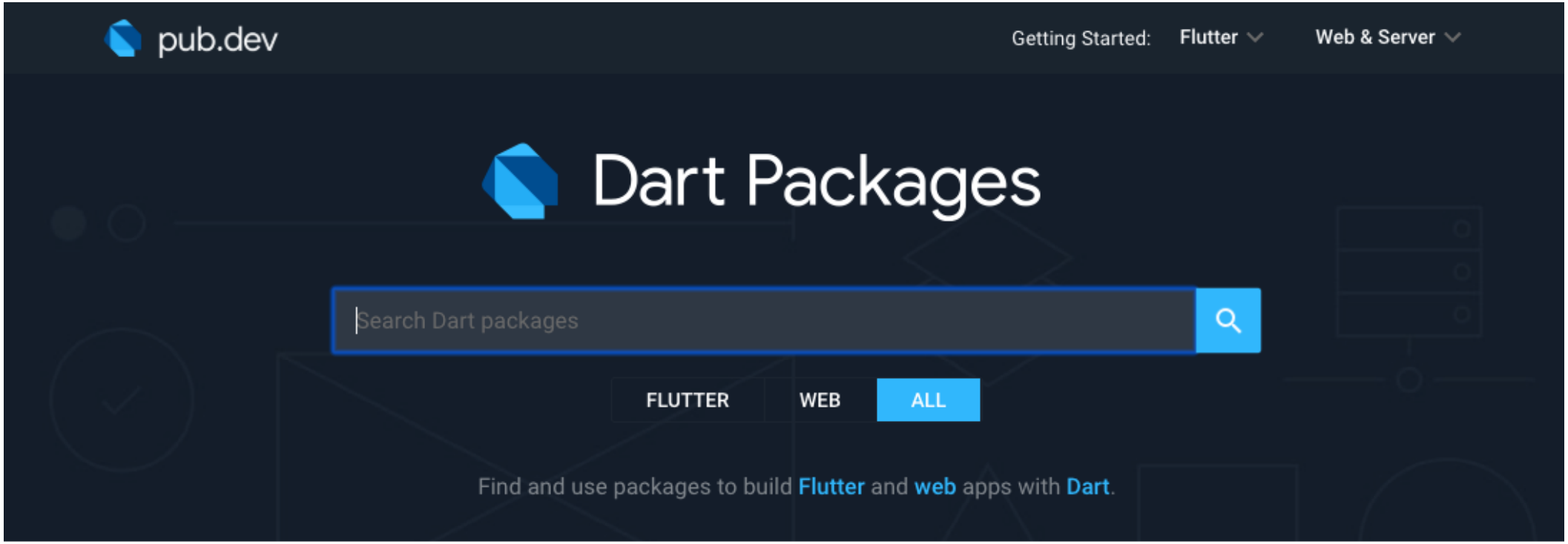
```dart
/// Implement this for a Built Value.
///
/// Then use built_value_generator.dart code generation functionality to
/// provide the rest of the implementation.
///
/// See https://github.com/google/built_value.dart/tree/master/example
abstract class Built<V extends Built<V, B>, B extends Builder<V, B>> {
  /// Rebuilds the instance.
  ///
  /// The result is the same as this instance but with [updates] applied.
  /// [updates] is a function that takes a builder [B].
  ///
  /// The implementation of this method will be generated for you by the
  /// built_value generator.
  V rebuild(updates(B builder));

  /// Converts the instance to a builder [B].
  ///
  /// The implementation of this method will be generated for you by the
  /// built_value generator.
  B toBuilder();
}
```


abstract class User implements Identifiable, Built<User, UserBuilder> {
factory User([updates(UserBuilder b)]) = _$U
User._();
@nullable
String get firstName;
@nullable
String get lastName;
package:built_value/built_value.dart
abstract class Built<V extends Built<V, B>, B
extends Builder<V, B>>
Implement this for a Built Value.
Then use built*value*generator.dart code generation
functionality to provide the rest of the implementation.
See
https://github.com/google/built_value.dart/tree/master/example
Dart Packages

F1

Dart 👍👍👍

Пакетный менеджер



Top Dart packages

http

FLUTTER WEB OTHER

A composable, cross-platform, Future-based API for making HTTP requests.

url_launcher

FLUTTER

Flutter plugin for launching a URL on Android and iOS. Supports web, phone, SMS, and email schemes.

path_provider

FLUTTER

Flutter plugin for getting commonly used locations on the Android & iOS file systems, such as the temp and app data directories.

image_picker

FLUTTER

Flutter plugin for selecting images from the Android and iOS image library, and taking new pictures with the camera.

rxdart

FLUTTER WEB OTHER

RxDart is an implementation of the popular reactiveX api for asynchronous programming, leveraging the native Dart Streams api.

sqflite

FLUTTER

Flutter plugin for SQLite, a self-contained, high-reliability, embedded, SQL database engine.

intl 0.16.0

Published Aug 26, 2019 FLUTTER WEB OTHER

Readme Changelog Installing Versions 88

Intl

This package provides internationalization and localization facilities, including message translation, plurals and genders, date/number formatting and parsing, and bidirectional text.

General

The most important library is intl. It defines the Intl class, with the default locale and methods for accessing most of the internationalization mechanisms. This library also defines the DateFormat, NumberFormat, and BidiFormatter classes.

Current locale

The package has a single current locale, called defaultLocale. Operations will use that locale unless told to do otherwise.

You can explicitly set the global locale

```
Intl.defaultLocale = 'pt_BR';
```

or get it from the browser

```
import 'package:intl/intl_browser.dart';
...
findSystemLocale().then(runTheRestOfMyProgram);
```

About

Contains code to deal with internationalized/localized messages, date and number formatting and parsing, bi-directional text, and other internationalization issues.

Repository (GitHub)

View/report issues

API reference

Author

Dart Team

Uploader

dgrove@google.com
alanknight@google.com
jmesserly@google.com
sigmund@google.com
nweiz@google.com
jacobr@google.com
keertip@google.com
kevmoo@google.com

License

BSD (LICENSE)

Dependencies

path

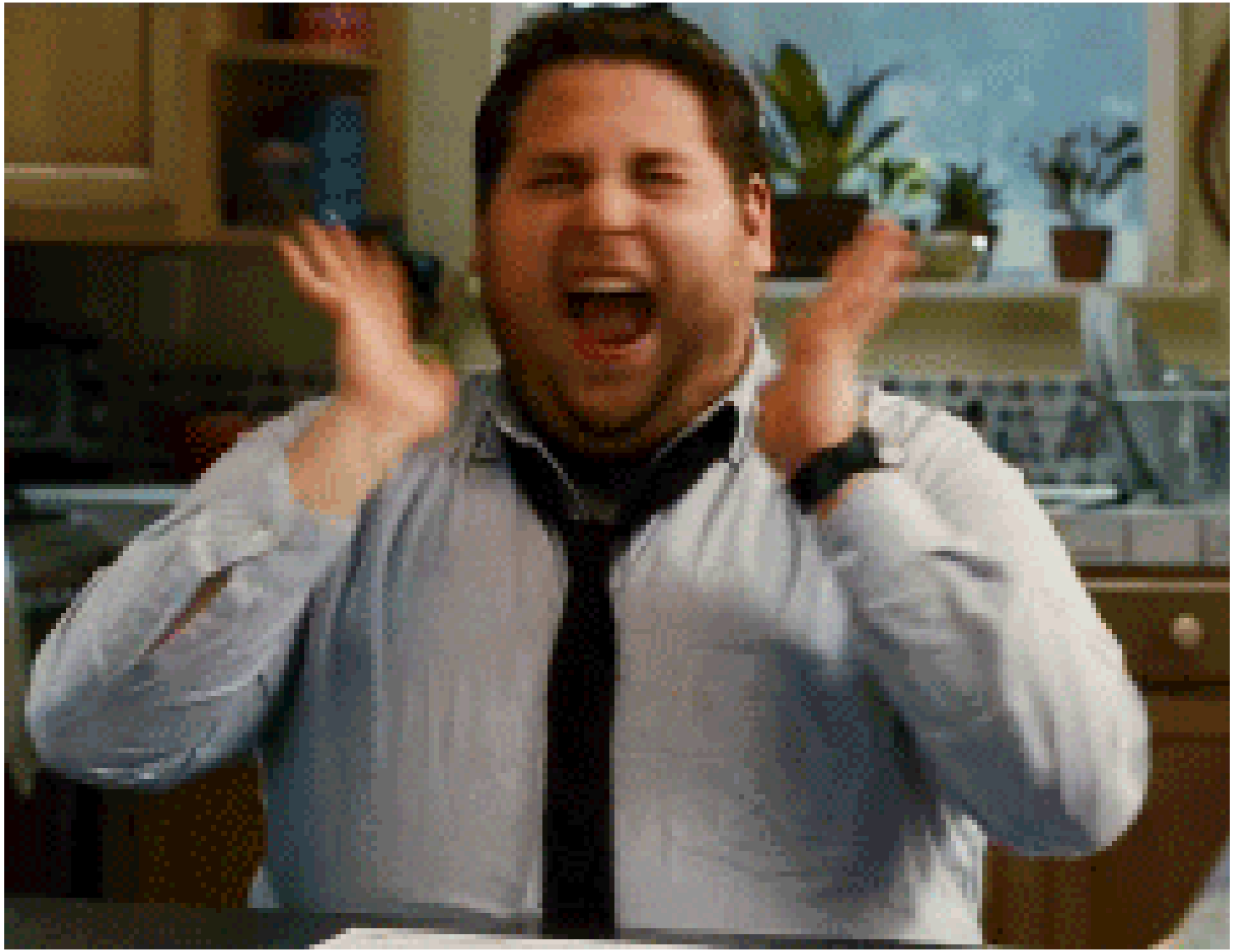
More

Packages that depend on intl

```yaml
name: project name
version: 0.0.1

environment:
  sdk: '>=2.4.0 <3.0.0'

dependencies:
  intl: ^0.15.7
```

pubspec.yaml

```
pub get
```

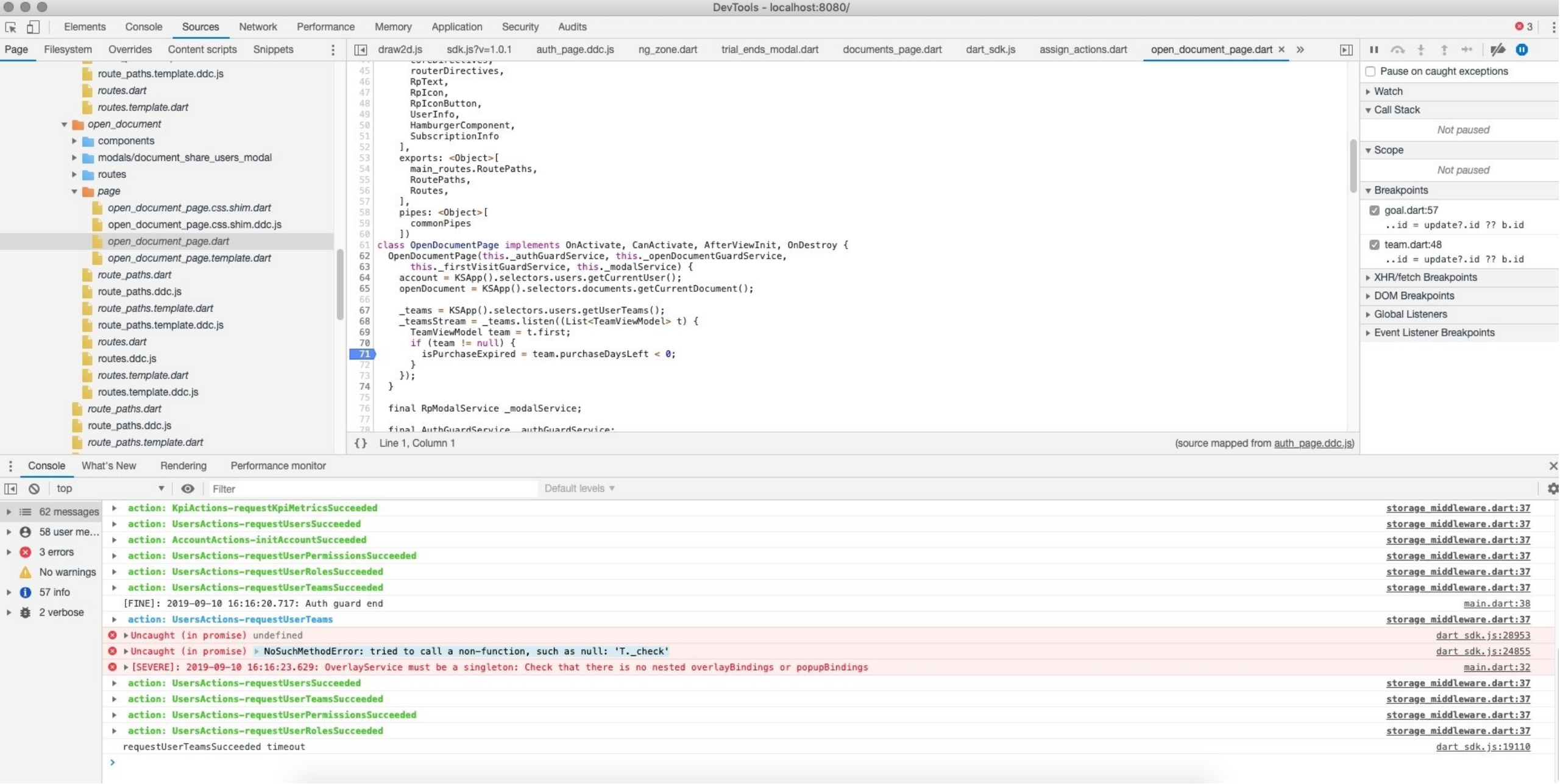


Когда либа импортится за 10 секунд

Dart 👍👍👍👍👍👍👍👍👍👍

А как дебажить?

```
console.log("Best debug ever (No)");
```



Chrome Dev Tools


ДЕБАГ
НУ ДАВАЙ ЖЕ, ПОЙДЕМ! ЭТО ПРИКЛЮЧЕНИЕ НА 20 МИНУТ!


* 10 ЧАСОВ СПУСТЯ *
[adult swim]

Build system

	Develop	Deploy	
	JIT + VM	AOT + runtime	Native x64/ARM
	dartdevc	dart2js	Web JavaScript

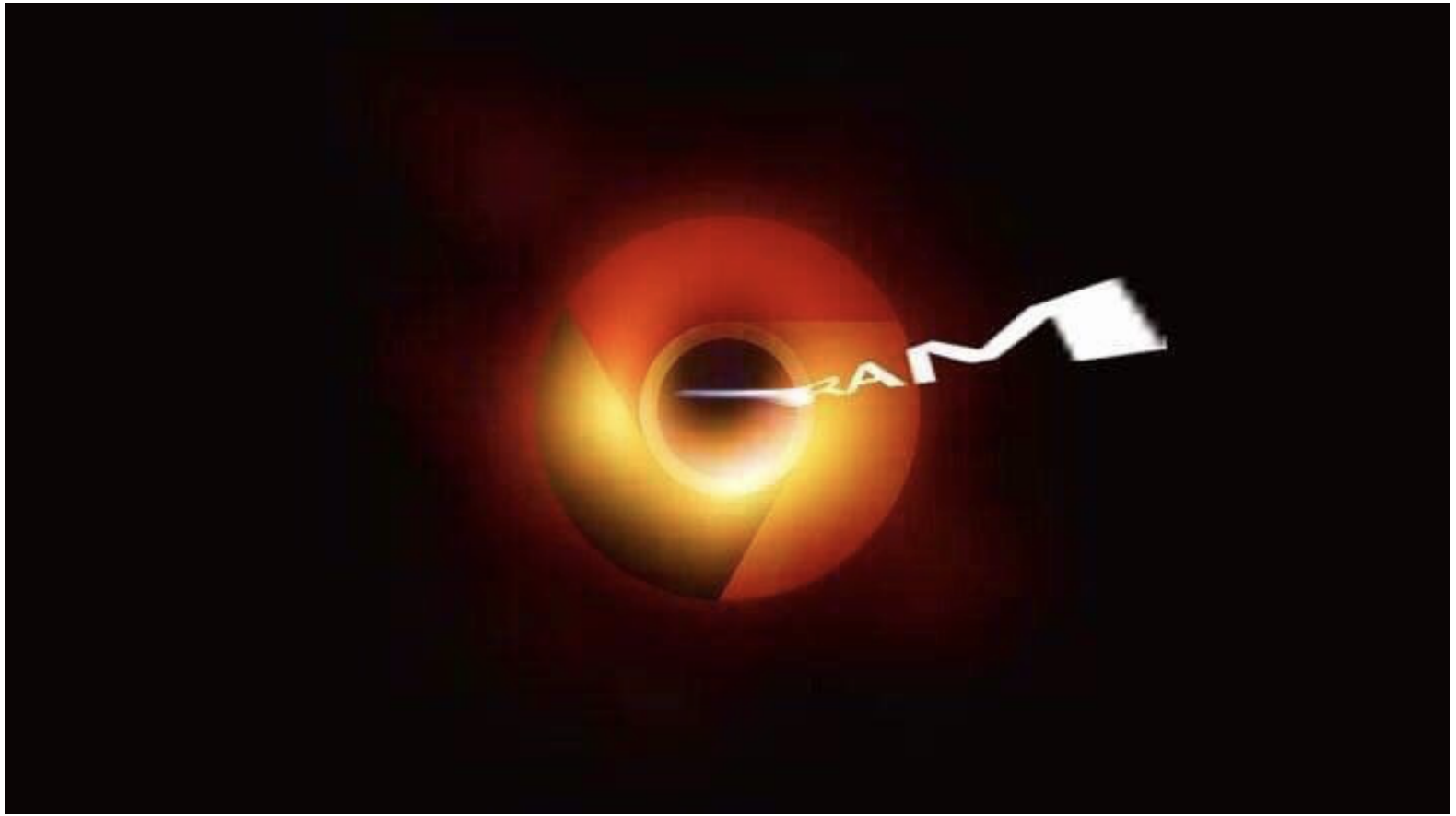
6

Сложности перехода с C++ на веб разработку


JS
ТЫ ЧТО ТАКОЕ

JS Garbage Collector

Забить на потребление памяти



Сложности **web**-разработки на **Dart**

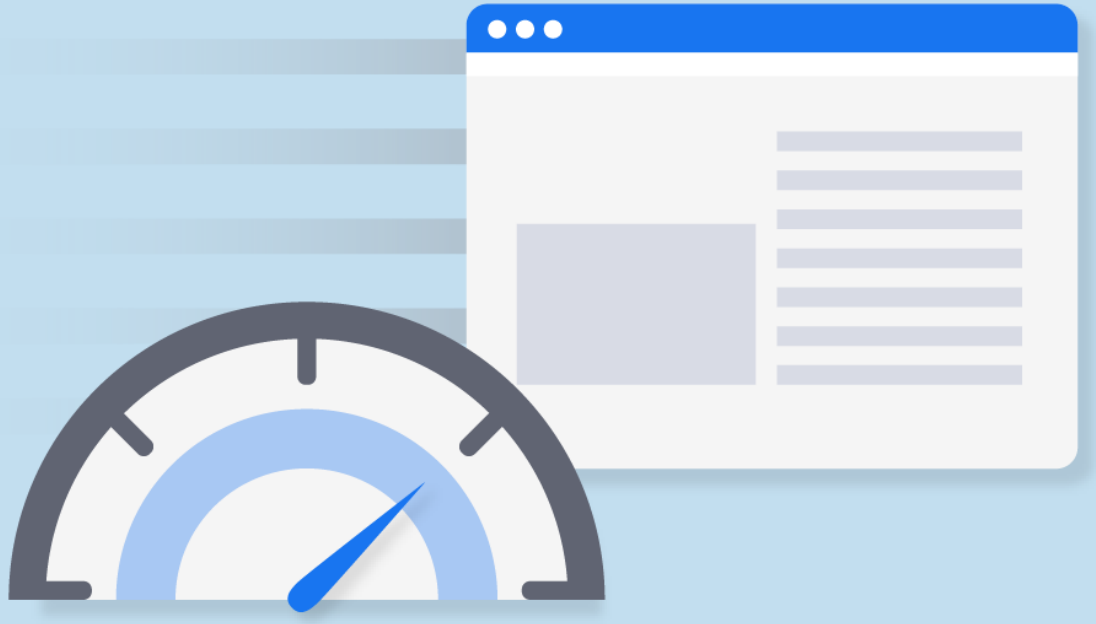
Мало информации

Велосипедо-строение

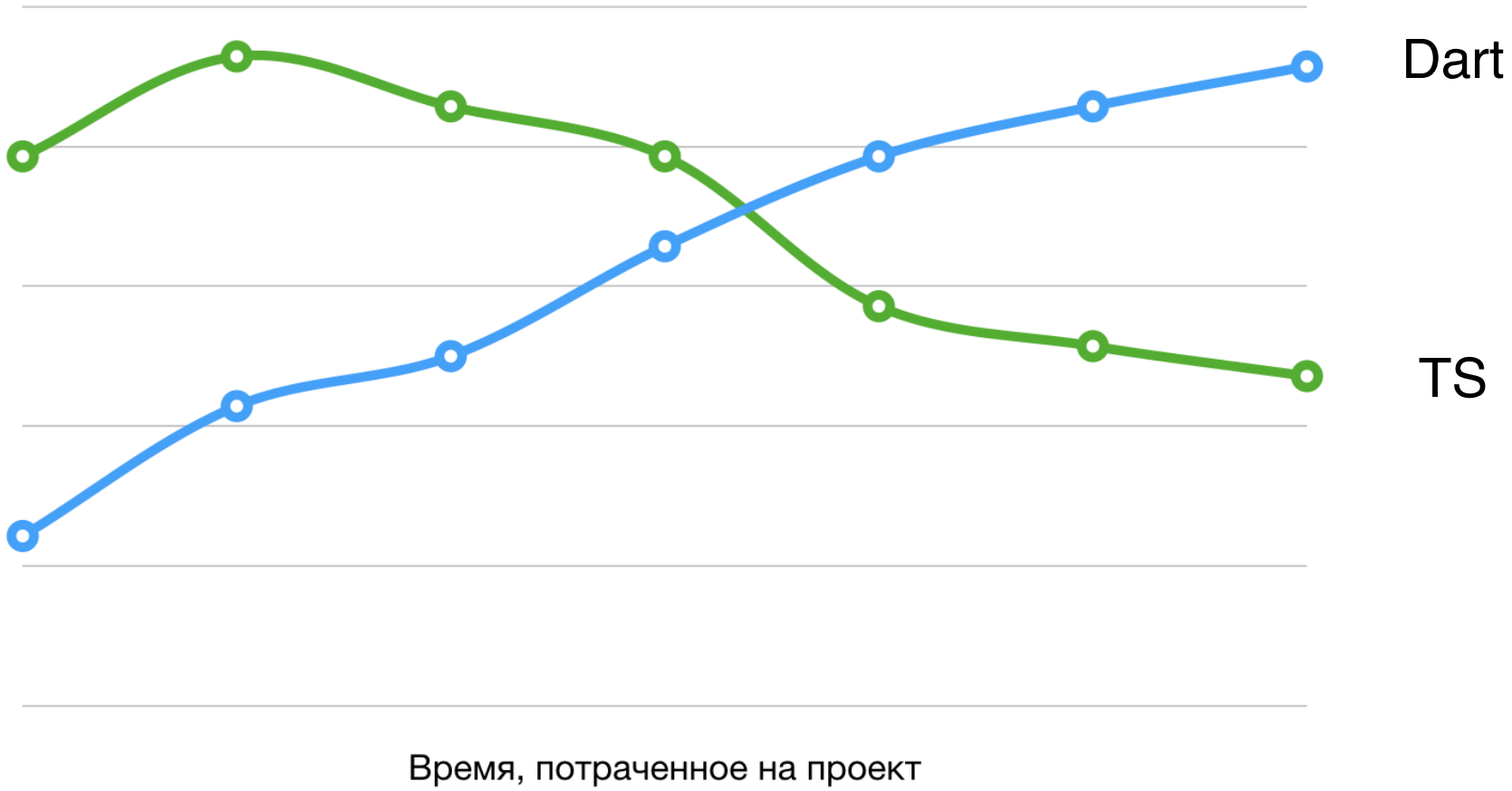
7 Выводы

Batteries included

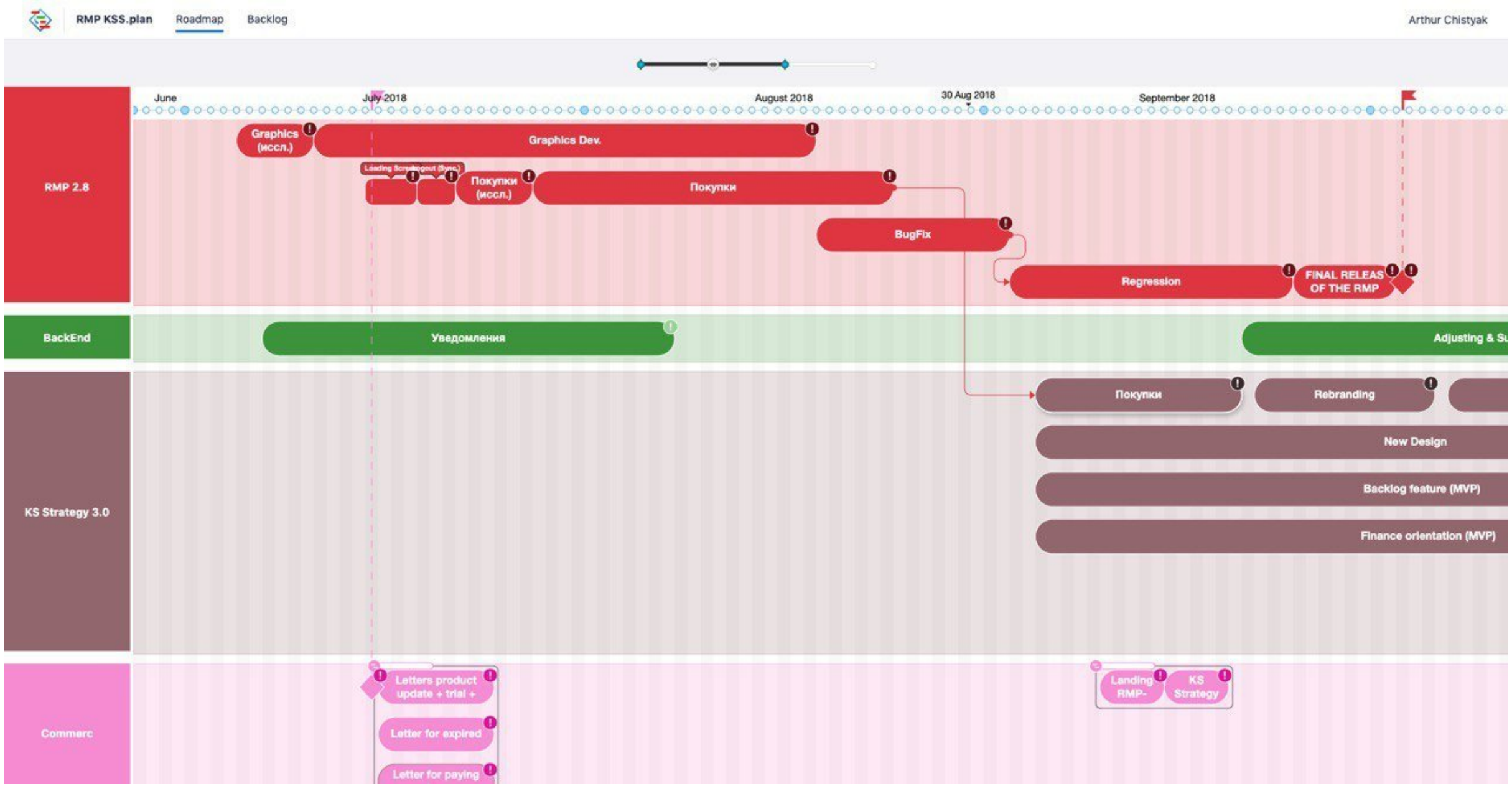
Веб разработка быстрее нативной



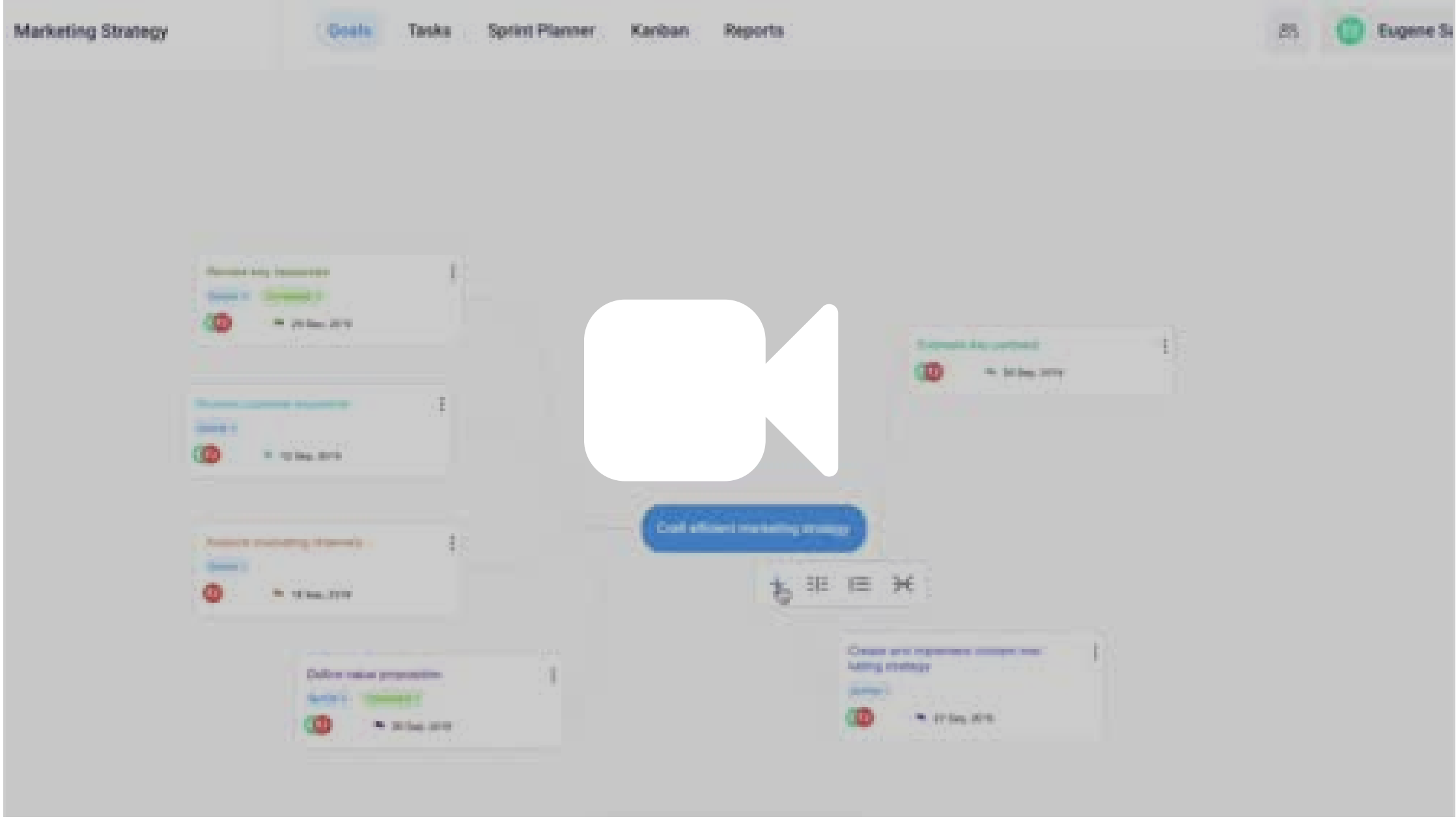
Продуктивность разработки



Реализованные проекты



Roadmap Planner Web Viewer



KeepSolid Goals

Dart хорош, но не везде

Worst Programming Languages to Learn in 2018 Rankings

Ranked from Worst to Best Languages to Learn

Появился

2011

Dart 2.0

2018

медленно развивался

Появился

2011

Dart 2.0

2018

медленно развивался

2019

Появился

2011

медленно развивался

Dart 2.0

2018

2019

Flutter-Хайп

Worst Programing Languages to Learn in 2019: Rankings

What Not to Learn in 2019

	Community Engagement	Job Market	Growth and Trends	Overall Raking
1	CoffeeScript	Dart	CoffeeScript	Elm
2	Erlang	Lua	Objective-C	CoffeeScript
3	Elixir	Clojure	Elm	Erlang
4	Elm	Erlang	Erlang	Lua
5	Objective-C	Elm	Perl	Perl
6	Dart	Haskell	Lua	Clojure
7	Lua	Rust	Clojure	Elixir
8	Clojure	Perl	Scala	Objective-C
9	Perl	CoffeeScript	Haskell	Haskell
10	Scala	Elixir	Ruby	Scala

Worst Programing Languages to Learn in 2019: Rankings

What Not to Learn in 2019

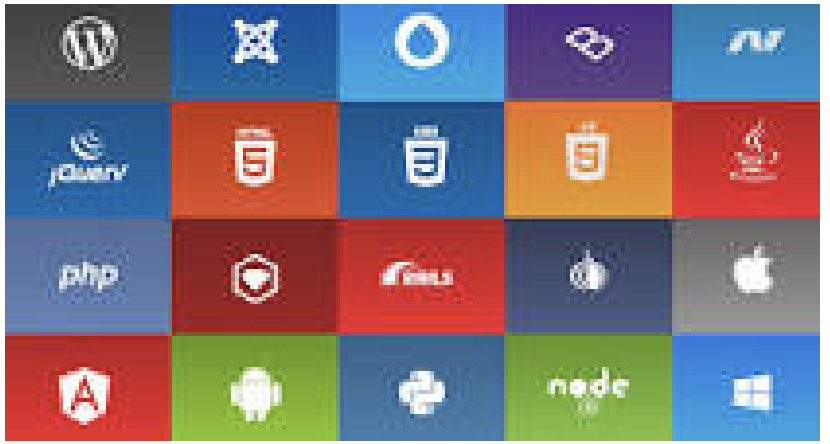
	Community Engagement	Job Market	Growth and Trends	Overall Raking
1	CoffeeScript	Dart	CoffeeScript	Elm
2	Erlang	Lua	Objective-C	CoffeeScript
3	Elixir	Clojure	Elm	Erlang
4	Elm	Erlang	Erlang	Lua
5	Objective-C	Elm	Perl	Perl
6	Dart	Haskell	Lua	Clojure
7	Lua	Rust	Clojure	Elixir
8	Clojure	Perl	Scala	Objective-C
9	Perl	CoffeeScript	Haskell	Haskell
10	Scala	Elixir	Ruby	Scala

Worst Programing Languages to Learn in 2019: Rankings

What Not to Learn in 2019

	Community Engagement	Job Market	Growth and Trends	Overall Raking
1	CoffeeScript	Dart	CoffeeScript	Elm
2	Erlang	Lua	Objective-C	CoffeeScript
3	Elixir	Clojure	Elm	Erlang
4	Elm	Erlang	Erlang	Lua
5	Objective-C	Elm	Perl	Perl
6	Dart	Haskell	Lua	Clojure
7	Lua	Rust	Clojure	Elixir
8	Clojure	Perl	Scala	Objective-C
9	Perl	CoffeeScript	Haskell	Haskell
10	Scala	Elixir	Ruby	Scala

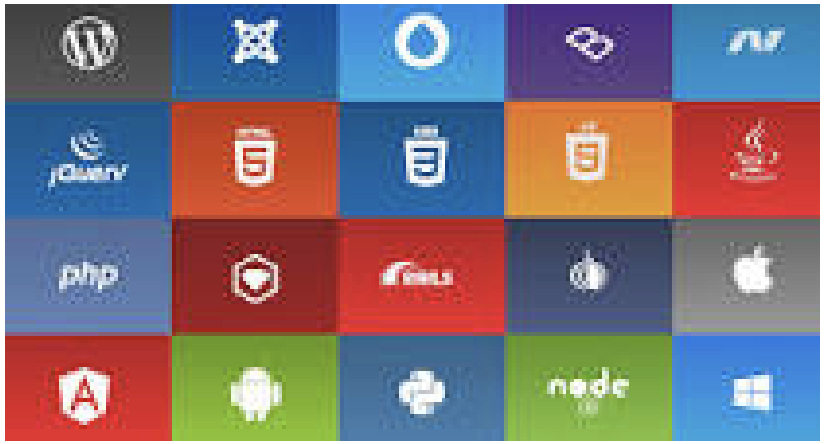
Top 10 Programming Languages To Learn (And Earn) in 2019 - And Why?



- Ruby. If you're looking for a **language** that's syntax-friendly, Ruby is a great choice. ...
- Python. As a **programming language**, Python is appreciated for being very user-friendly. ...
- Swift. ...
- JavaScript. ...
- Kotlin. ...
- Rust. ...
- Java. ...
- **Dart**.

More items... • Feb 8, 2019

Top 10 Programming Languages To Learn (And Earn) in 2019 - And Why?

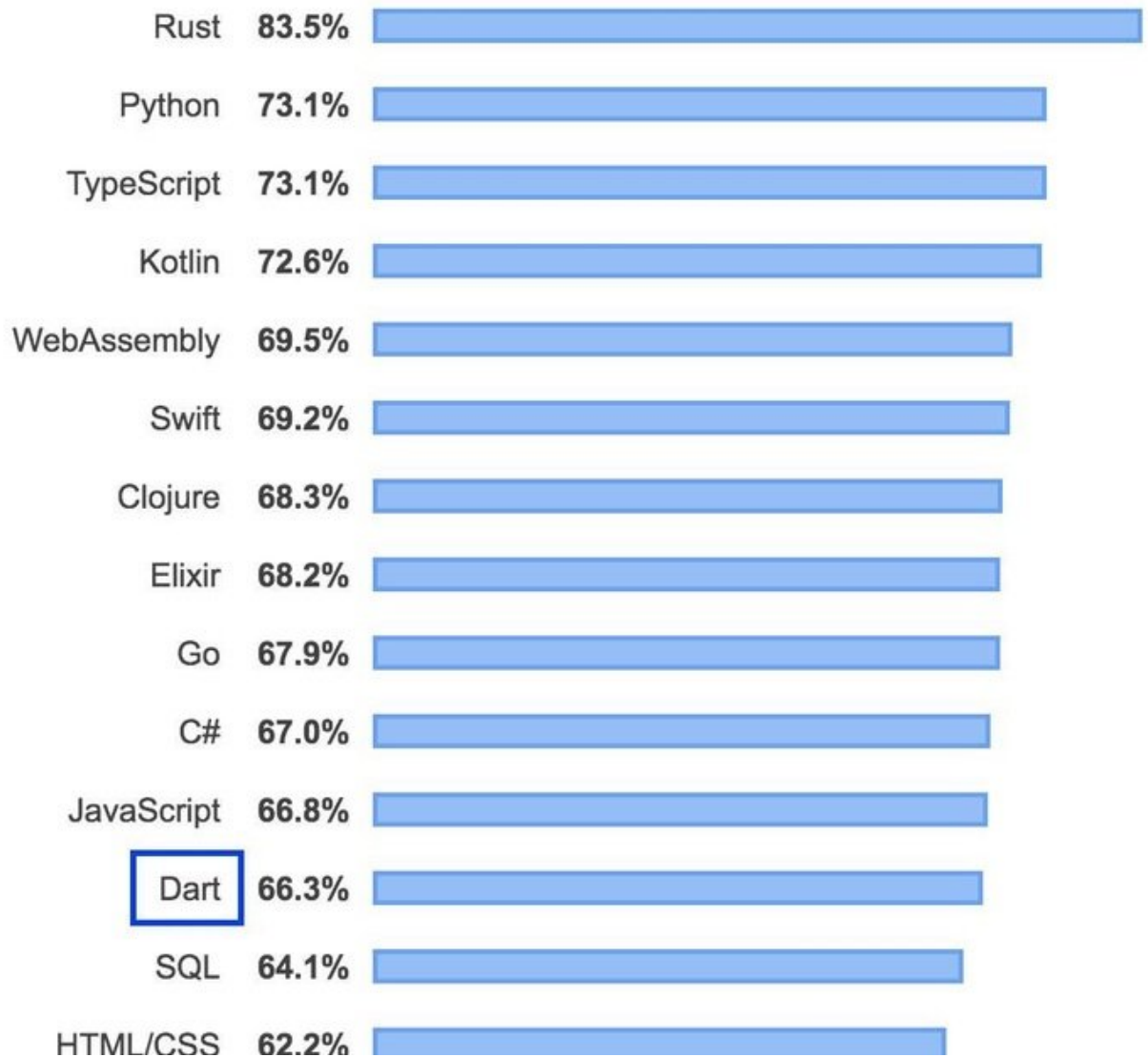


- Ruby. If you're looking for a **language** that's syntax-friendly, Ruby is a great choice. ...
- Python. As a **programming language**, Python is appreciated for being very user-friendly. ...
- Swift. ...
- JavaScript. ...
- Kotlin. ...
- Rust. ...
- Java
- **Dart**.

More items... • Feb 8, 2019

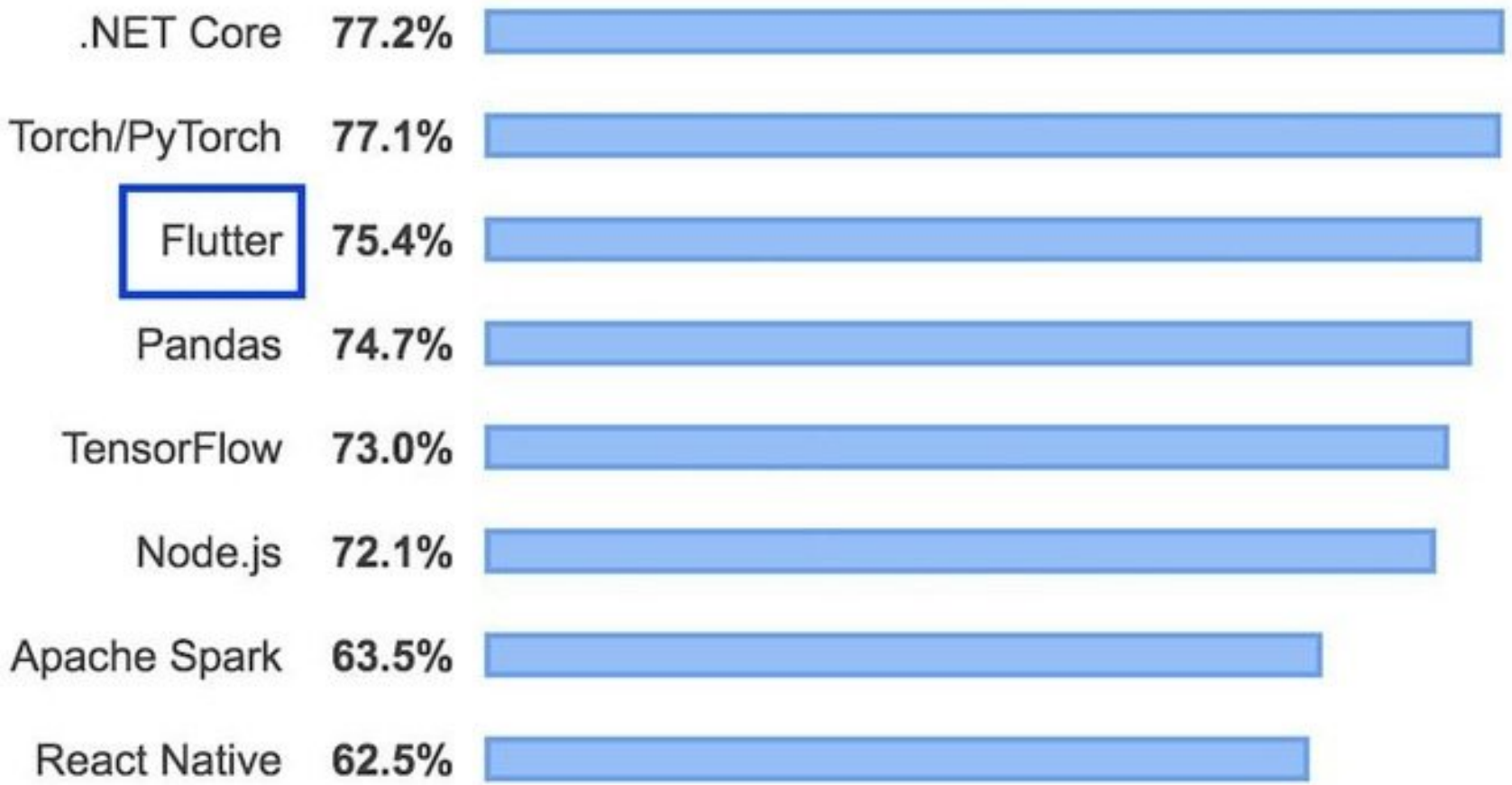
Most Loved, Dreaded, and Wanted Languages

Loved | Dreaded | Wanted



Most Loved, Dreaded, and Wanted Other Frameworks, Libraries, and Tools

Loved | Dreaded | Wanted



Появился

2011

медленно развивался

Dart 2.0

2018

2019

Flutter-Хайп

Появился

2011

Dart 2.0

2018

медленно развивался

2019

Flutter-Хайп

Появился

2011

медленно развивался

Dart 2.0

2018

2019

Flutter-Хайп

Будущее

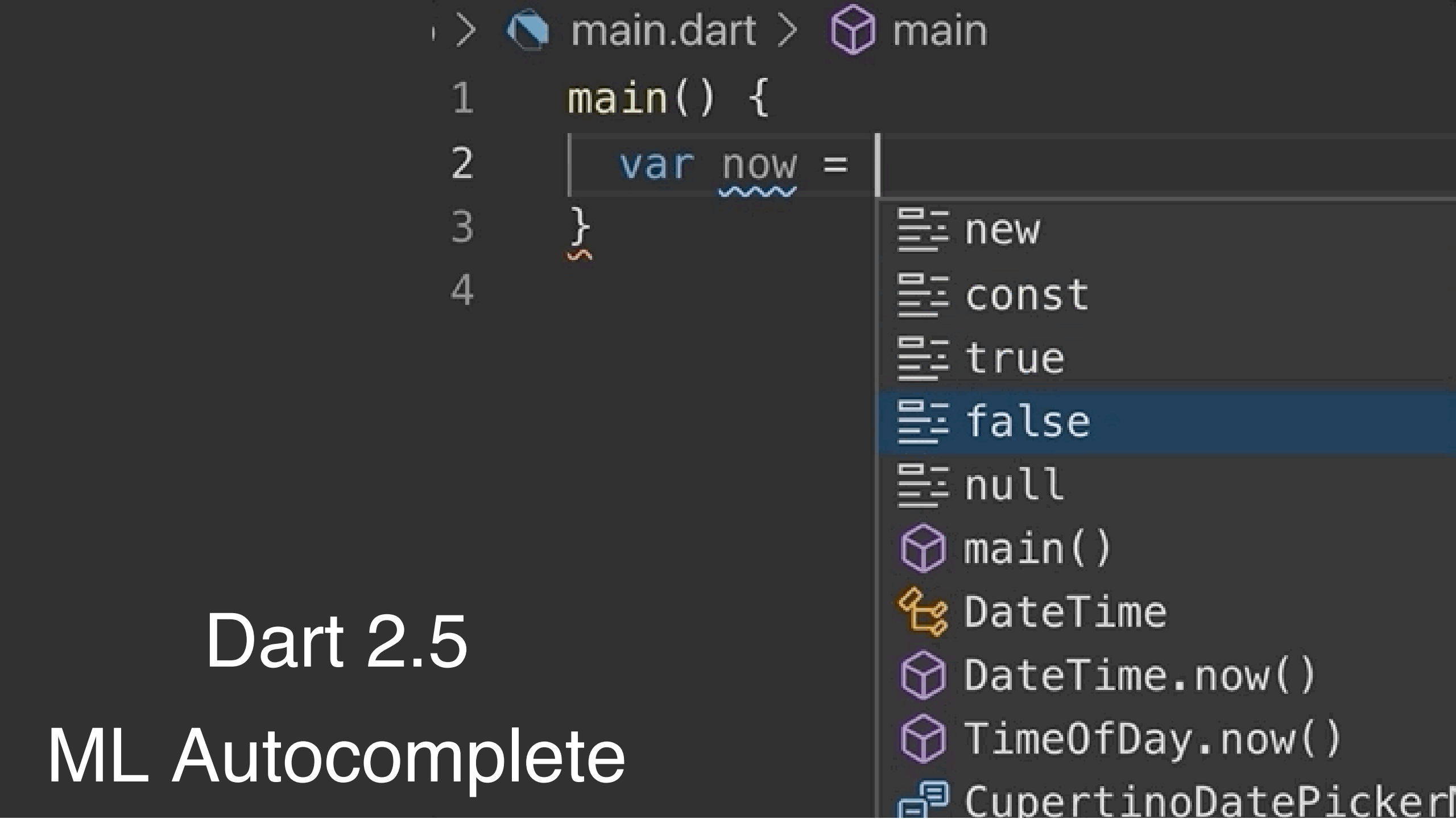
Dart 2.5

Preview
ML Complete:
Code completions
ranked by
machine learning

Preview
`dart:ffi`
foreign function
interface for calling C

Improved const

```
const Object i = 3;
const list = [i as int];
const set = {if (list is
    List<int>) ...list};
```


main.dart > main
1 main() {
2 var now =
3 }
4
new
const
true
false
null
main()
DateTime
DateTime.now()
TimeOfDay.now()
Dart 2.5
ML Autocomplete

```dart
main() {
  Run | Debug
  test("whatever", () {
    final w = Whatever();

    expect(w.foo, equals("foo"));
    expect(w.bar, equals("bar"));
  });
}

class Whatever {
  String get foo => "foo";
}

extension on Whatever {
  String get bar => "bar";
}
```

Dart 2.6

Extensions

Итого:

К плюсам возвращаться не хочется


C++ разработчики
JS
C++