

Яндекс



Работа с сетью

Алексей Шведчиков , Разработчик

Без сети - никуда



6.8.1



6.8.2

Тупик



6.8.3

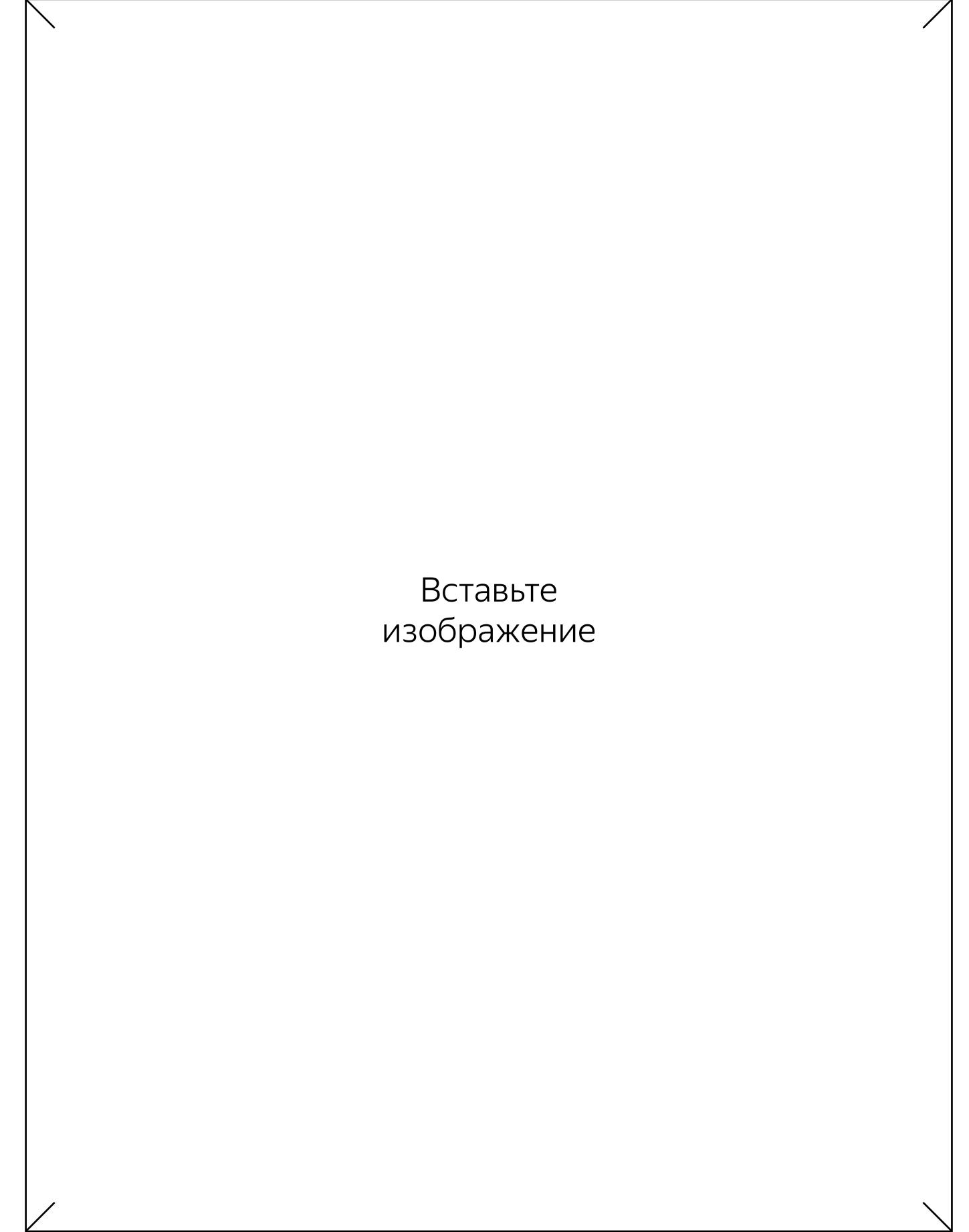


А как работать с сетью?

Обзор

1. Сокеты
2. Пакеты
3. СерIALIZАЦИЯ

Вставьте
изображение



Сокеты



HTTP



gRPC

TCP

VS

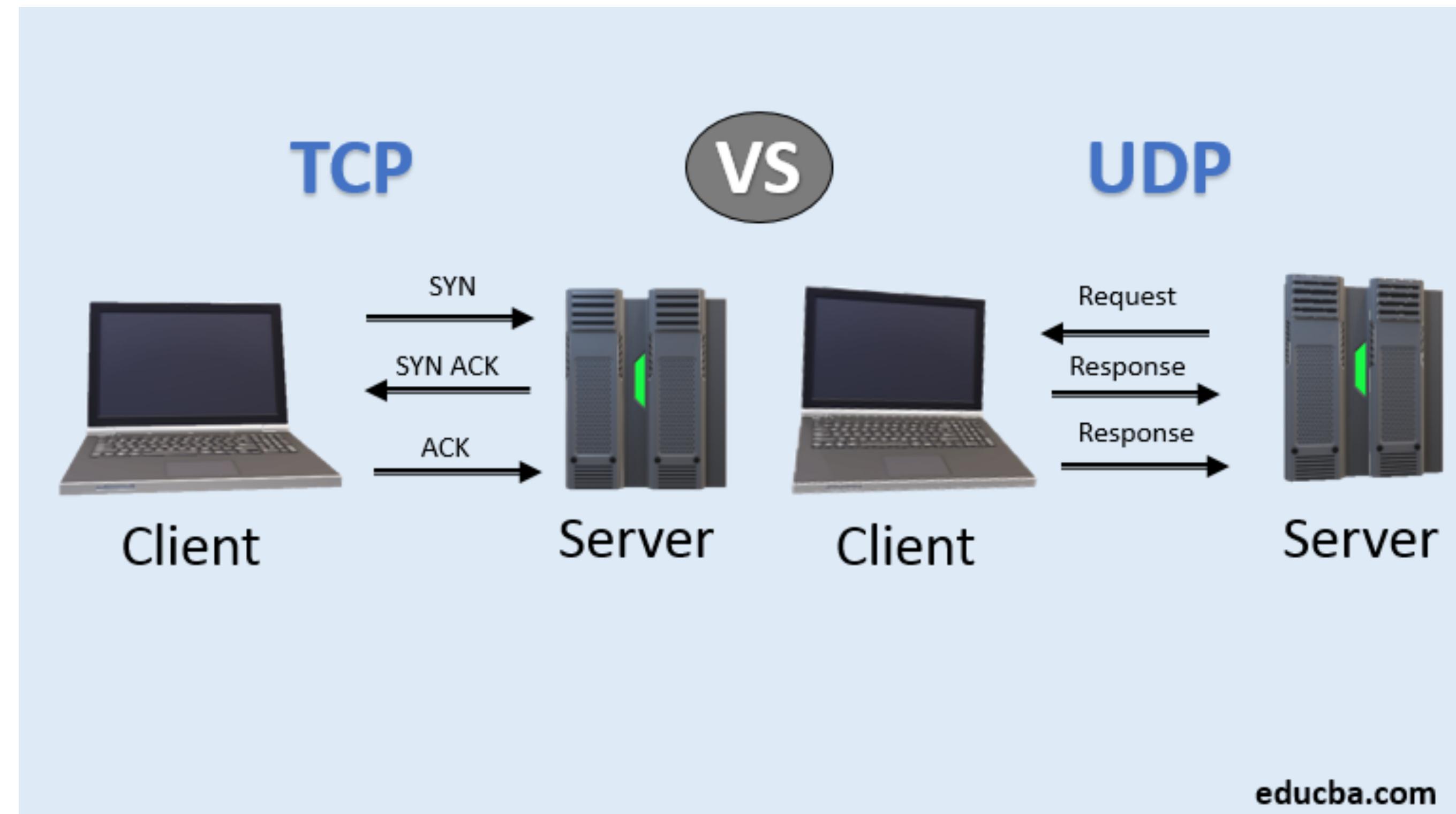
UDP



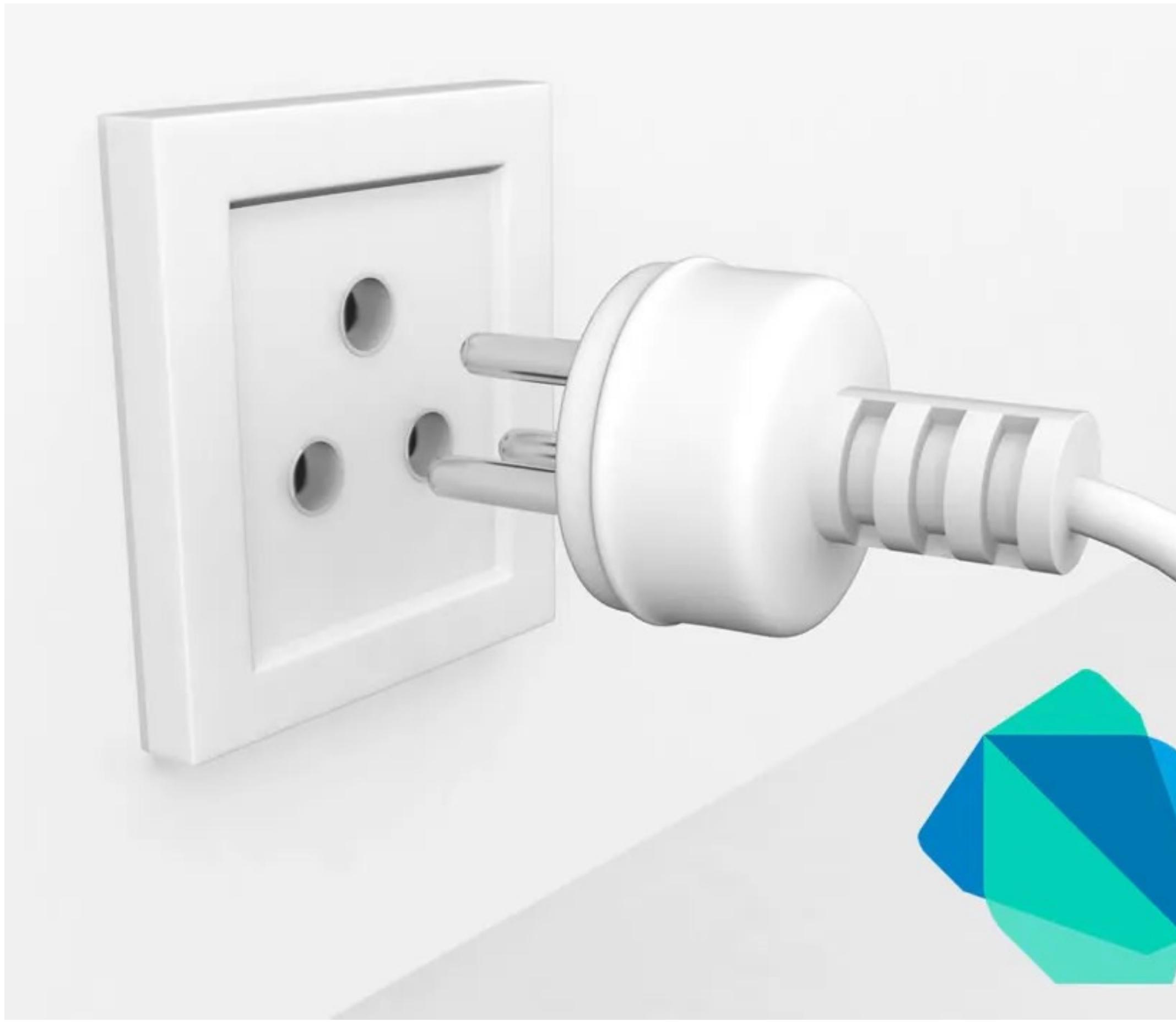
educba.com



FTP



Сокеты



Dart из коробки

```
//открыть соединение  
final socket = await Socket.connect('localhost', 4567);
```

```
//получать данные  
socket.listen(...)
```

```
//отправлять данные  
socket.write(...)
```

HTTP



Велосипед.

Жак Фреско

http



A screenshot of a web browser displaying the pub.dev package page for the `http` package version 0.13.4. The page has a dark theme with a light background. At the top, the URL bar shows `pub.dev`. The main header features the `http` logo and the text "http | Dart Package". Below the header, there's a search bar, a "Sign in" button, and a "Help" dropdown menu. The package name "http 0.13.4" is prominently displayed in large font. Below it, a timestamp "Published 2 months ago" and a "Null safety" badge are shown. A popularity chart indicates 3734 likes, 130 pub points, and 100% popularity. The package is listed for multiple platforms: DART, NATIVE, JS, FLUTTER, ANDROID, IOS, LINUX, MACOS, WEB, and WINDOWS. A "Readme" tab is currently selected. The description states: "A composable, Future-based library for making HTTP requests." Below the description, there are badges for "pub v0.13.4" and "Dart CI passing". The package details section notes: "This package contains a set of high-level functions and classes that make it easy to consume HTTP resources. It's multi-platform, and supports mobile, desktop, and the browser." The "Using" section explains: "The easiest way to use this library is via the top-level functions. They allow you to make individual HTTP requests with minimal hassle:" followed by a code snippet:

```
import 'package:http/http.dart' as http;
```

http

Как сделать запрос?

1. Подключи http

```
http: ^0.13.4
```

2. Импортируй

```
import 'package:http/http.dart' as http;
```

3. Используй

```
http.get(Uri.parse('https://api.ipify.org'));
```

Виды запросов

http.	
atios	λ <code>delete(Uri url, {Map<String, String>? headers, Encoding encoding})</code> Future<Response>
ht	λ <code>get(Uri url, {Map<String, String>? headers, Encoding encoding})</code> Future<Response>
	λ <code>head(Uri url, {Map<String, String>? headers, Encoding encoding})</code> Future<Response>
	λ <code>patch(Uri url, {Map<String, String>? headers, Encoding encoding})</code> Future<Response>
	λ <code>post(Uri url, {Map<String, String>? headers, Encoding encoding})</code> Future<Response>
	λ <code>put(Uri url, {Map<String, String>? headers, Encoding encoding})</code> Future<Response>
	Свойства

Параметры запросов

`Uri url, {Map<String, String>} headers, Object body, Encoding encoding}`

Про параметры

Query

`https://www.domain.com/url?variable=value&variable=value`



start of query string

separator



```
Uri(  
...  
queryParameters: { 'search': 'Sochi' }  
...  
) ;
```

Headers

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Host: vk.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/605.1.15
(KHTML, like Gecko) Version/15.1 Safari/605.1.15
Accept-Language: en-us
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

```
http.get(
  ...
  headers: { 'Auth': 'alexs' },
  ...
);
```

Тело запроса

```
http.get(  
  ...  
  body: 'some string',  
  ...  
) ;
```

Клиент

```
final client = http.Client();
```

```
client.|
```

(m) send (BaseRequest request)	Future<StreamedResponse>
(m) delete (Uri url, {Map<String, String>? hea...	Future<Response>
(m) close ()	void
(m) get (Uri url, {Map<String, String>? header...	Future<Response>
(m) head (Uri url, {Map<String, String>? heade...	Future<Response>
(m) patch (Uri url, {Map<String, String>? head...	Future<Response>
(m) post (Uri url, {Map<String, String>? header...	Future<Response>
(m) put (Uri url, {Map<String, String>? header...	Future<Response>
(m) read (Uri url, {Map<String, String>? headers...	Future<String>
(m) readBytes (Uri url, {Map<String, String>?...	Future<Uint8List>

```
client.send(request);
```

MultipartRequest extends BaseRequest

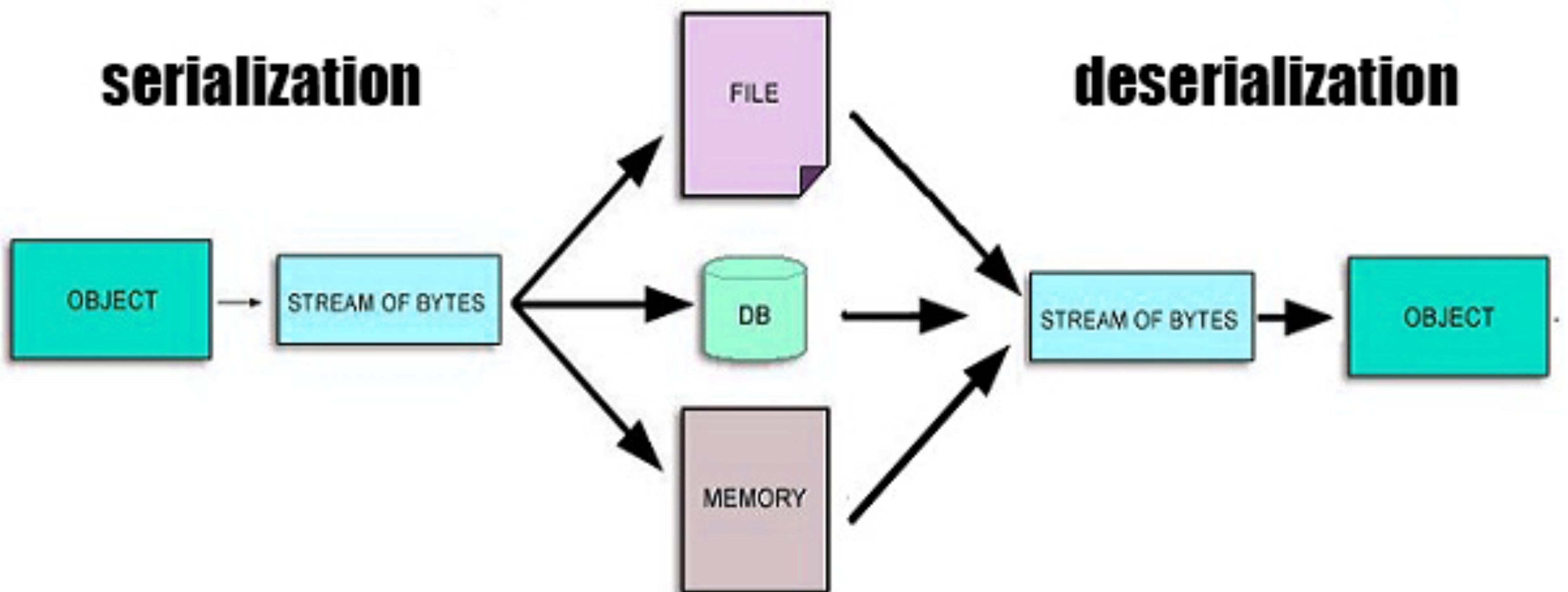
```
final multipart = http.MultipartRequest('POST', uri);
multipart.files.add(MultipartFile(...));

client.send(multipart);
```

Сериализация



Сериализация



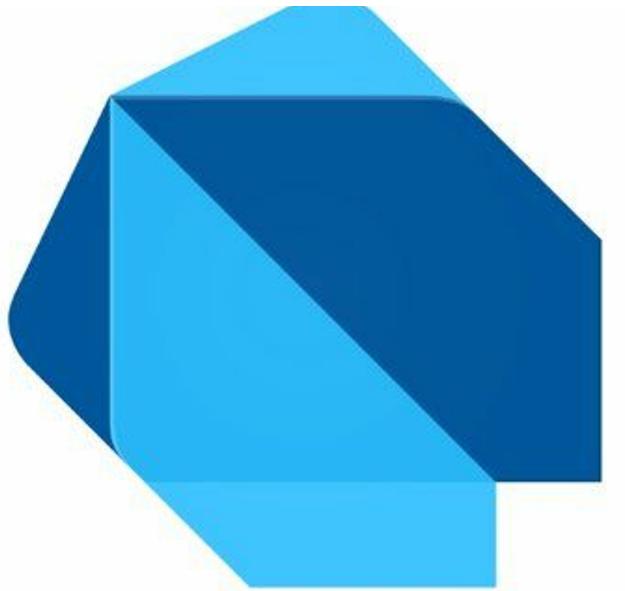
JSON



JSON generated from JSONC.json

```
{  
  "foo": "bar foo",  
  "true": false,  
  "number": 42,  
  "array": [1, 2, 3]  
}
```

Dart



Из коробки

1. Импорт

```
import 'dart:convert';
```

2. Сериализация

```
jsonEncode(...); // передаем Map, получаем строку
```

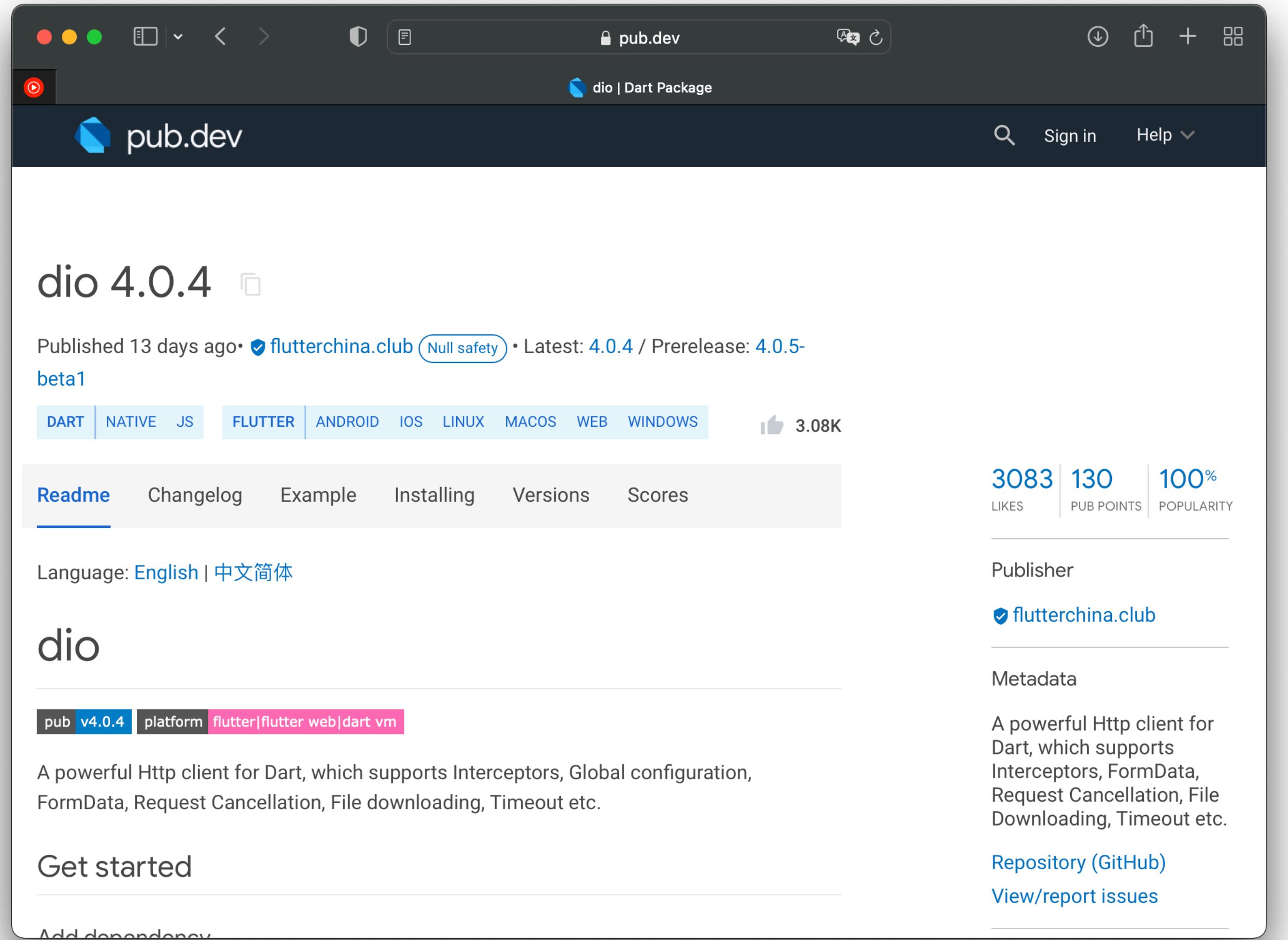
3. Десериализация

```
jsonDecode(...); // передаем строку, получаем Map
```

хочется чего-то большего

dio





The screenshot shows the pub.dev page for the dio package. The title is "dio 4.0.4". It was published 13 days ago by flutterchina.club (Null safety). The latest version is 4.0.4, and the previous version is 4.0.5-beta1. The package is available for Dart, Native, JS, Flutter, Android, iOS, Linux, macOS, Web, and Windows. It has 3.08K likes. The Readme tab is selected. The language is English | 中文简体. The publisher is flutterchina.club. The package is a powerful Http client for Dart, supporting Interceptors, Global configuration, FormData, Request Cancellation, File downloading, and Timeout etc. The repository is on GitHub.

dio 4.0.4

Published 13 days ago •  flutterchina.club (Null safety) • Latest: 4.0.4 / Prerelease: 4.0.5-beta1

DART NATIVE JS FLUTTER ANDROID IOS LINUX MACOS WEB WINDOWS  3.08K

[Readme](#) [Changelog](#) [Example](#) [Installing](#) [Versions](#) [Scores](#)

Language: [English](#) | [中文简体](#)

dio

pub v4.0.4 platform flutter|flutter web|dart vm

A powerful Http client for Dart, which supports Interceptors, Global configuration, FormData, Request Cancellation, File downloading, Timeout etc.

Get started

Add dependency

Publisher
 flutterchina.club

Metadata
A powerful Http client for Dart, which supports Interceptors, FormData, Request Cancellation, File Downloading, Timeout etc.

[Repository \(GitHub\)](#)
[View/report issues](#)

dio

Как сделать запрос?

1. Подключи dio

```
dio: ^4.0.4
```

2. Импортируй

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

3. Создай клиент

```
final Dio _dio = Dio();
```

4. Используй

```
_dio.get('/data');
```

Фичи dio

Конфигурация клиента

```
9     final options = BaseOptions();  
10  
11 {String method, int connectTimeout, int receiveTimeout,  
12 int sendTimeout, String baseUrl: '', Map<String,  
13 dynamic> queryParameters, Map<String, dynamic> extra,  
14 Map<String, dynamic> headers,  
15 ResponseType responseType: ResponseType.json,  
16 String contentType, ValidateStatus validateStatus,  
17 bool receiveDataWhenStatusError, bool followRedirects,  
18 int maxRedirects, RequestEncoder requestEncoder,  
19 ResponseDecoder responseDecoder, ListFormat listFormat,  
20 setRequestContentTypeWhenNoPayload: false}
```

```
final Dio _dio = Dio(options);
```

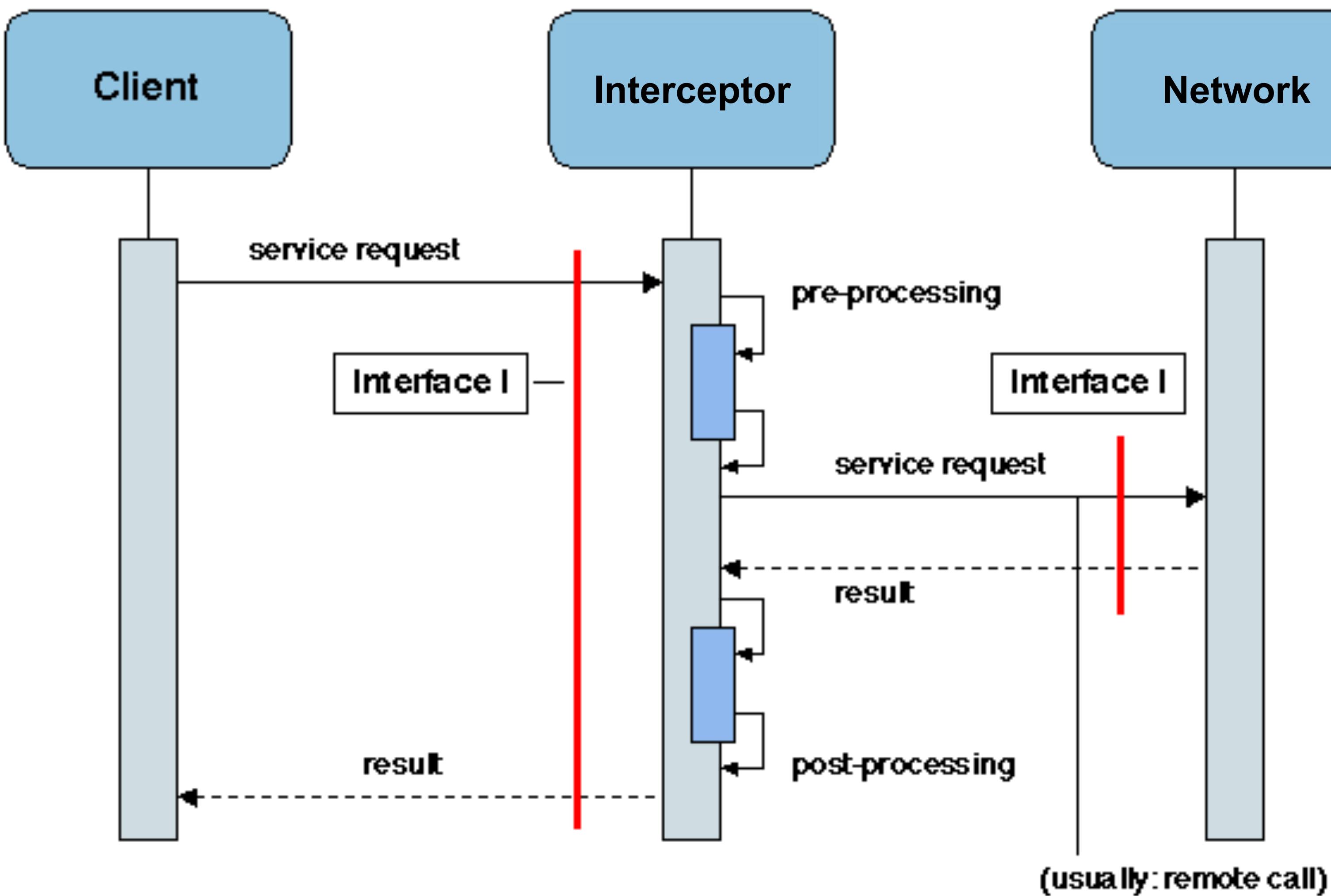
Десериализация // получаем Map

```
_dio.get<Map<String, dynamic>>('/data');
```

Сериализация // отправляем Map

```
_dio.post('/data', data: {  
  'field': 'string'  
});
```

Интерцепторы



Интерцепторы

```
class AuthInterceptor extends Interceptor {  
  
    ...  
  
    @override  
    void onRequest(RequestOptions options, RequestInterceptorHandler handler) {  
        final newOptions = options.copyWith(  
            headers: {  
                ...options.headers,  
                'Auth': token  
            }  
        );  
        handler.next(newOptions);  
    }  
  
    ...  
}
```

Обработка исключений

```
try {  
    //make request  
} on DioError catch (e){  
    //handle error  
}
```

```
// 20150625170715
// https://api.github.com/repos/tulios/json-viewer

4 {
5   "id": 12635853,
6   "name": "json-viewer",
7   "full_name": "tulios/json-viewer",
8   "owner": {
9     "login": "tulios",
10    "id": 33231,
11    "avatar_url": "https://avatars.githubusercontent.com/u/33231?v=3",
12    "gravatar_id": "",
13    "url": "https://api.github.com/users/tulios",
14    "html_url": "https://github.com/tulios",
15    "followers_url": "https://api.github.com/users/tulios/followers",
16    "following_url": "https://api.github.com/users/tulios/following{/other_user}",
17    "gists_url": "https://api.github.com/users/tulios/gists{/gist_id}",
18    "starred_url": "https://api.github.com/users/tulios/starred{/owner}{/repo}",
19    "subscriptions_url": "https://api.github.com/users/tulios/subscriptions",
20    "organizations_url": "https://api.github.com/users/tulios/orgs",
21    "repos_url": "https://api.github.com/users/tulios/repos",
22    "events_url": "https://api.github.com/users/tulios/events{/privacy}",
23    "received_events_url": "https://api.github.com/users/tulios/received_events",
24    "type": "User",
25    "site_admin": false
26  },
27  "private": false,
28  "html_url": "https://github.com/tulios/json-viewer"
```

json_serializable

The screenshot shows the `json_serializable` package page on `pub.dev`. The page has a dark header with the `pub.dev` logo and a search bar. Below the header, there's a large title `json_serializable 6.1.1` with a copy icon. A `Flutter Favorite` badge is visible. The page includes sections for `DART` and `NATIVE`, a `Readme` tab (which is active), and other tabs like `Changelog`, `Example`, `Installing`, `Versions`, and `Scores`. A `pub v6.1.1` button is present. The main content area describes the package as providing Dart Build System builders for handling JSON. It details how builders generate code for classes annotated with `JsonSerializable` and how to use `JsonLiteral`. A `Setup` section provides instructions for configuring the project. The right sidebar displays statistics: 1524 likes, 120 pub points, and 99% popularity. It also lists the publisher (`google.dev`), metadata (automatically generates code for JSON conversion), repository (`GitHub`), documentation, API reference, and license information.

json serializable

кодогенерация.

1. Подключи это // dependencies

```
json_annotation: ^4.4.0
```

2. И это // dev_dependencies

```
json_serializable: ^6.1.1  
build_runner: ^2.1.5
```

3. Опиши модельку

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

part 'user.g.dart';

@JsonSerializable()
class User {
    final String name;
    final String email;

    User(this.name, this.email);

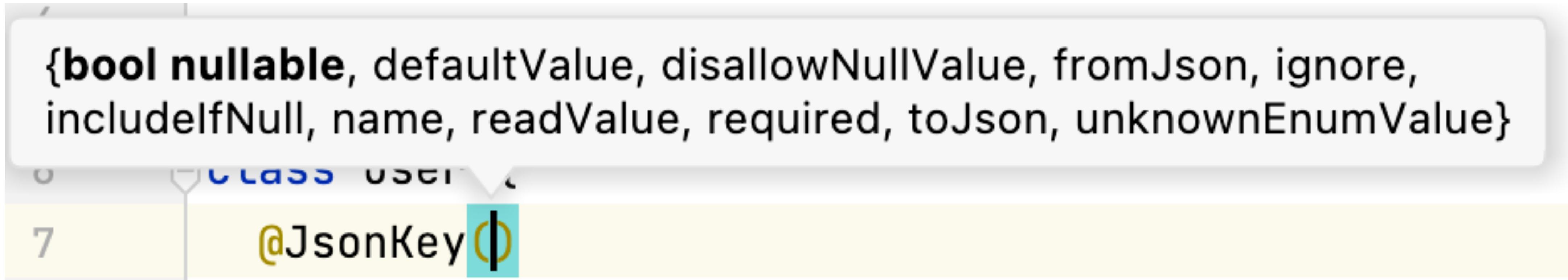
    factory User.fromJson(Map<String, dynamic> json) => _$UserFromJson(json);
    Map<String, dynamic> toJson() => _$UserToJson(this);
}
```

4. Запусти кодогенерацию

```
flutter pub run build_runner build
```

@JsonKey

Аннотация поля



@JsonSerializable Аннотация класса

```
{bool nullable, anyMap, checked, constructor, createFactory, createToJson,  
disallowUnrecognizedKeys, explicitToJson, fieldRename, ignoreUnannotated,  
includeIfNull, genericArgumentFactories}
```

5

@JsonSerializable()

Всякое, всякое

Всякие пакеты

Яндекс Дзен 

Chopper

Http Клиенты

Retrofit

Модельки

Freezed

Итог

1. Работа с сетью
2. http
2. Сериализация
4. dio
5. json_serializable
6. Всякое



Спасибо

Алексей Шведчиков

Разработчик



a1exs@yandex-team.ru



@a1ex5