

# Flutter: 最新进展和未来展望

董韬

高级研究员（开发者体验）  
Google Flutter 团队

# 极客邦科技 会议推荐2019

5月

**QCon** 北京

全球软件开发大会

大会: 5月6-8日  
培训: 5月9-10日

**QCon** 广州

全球软件开发大会

培训: 5月25-26日  
大会: 5月27-28日

6月

**GTLC**  
GLOBAL  
TECH LEADERSHIP  
CONFERENCE

上海

技术领导力峰会

时间: 6月14-15日

**GMTC** 北京

全球大前端技术大会

大会: 6月20-21日  
培训: 6月22-23日

7月

**ArchSummit** 深圳

全球架构师峰会

大会: 7月12-13日  
培训: 7月14-15日

10月

**QCon** 上海

全球软件开发大会

大会: 10月17-19日  
培训: 10月20-21日

11月

**GMTC** 深圳

全球大前端技术大会

大会: 11月8-9日  
培训: 11月10-11日

**AiCon** 北京

全球人工智能与机器学习大会

大会: 11月21-22日  
培训: 11月23-24日

12月

**ArchSummit** 北京

全球架构师峰会

大会: 12月6-7日  
培训: 12月8-9日



# 自我介绍

## 工业界背景

- 2016年下半年加入 Flutter 团队
- 目前负责 Flutter 的 API 可用性，开发工具设计，和文档可用性的研究工作



## 学术届背景

- 2009到2014年在密歇根大学从事人机交互方向的博士研究工作
- 多次担任ACM会议组委会成员和论文审稿人



# 目录

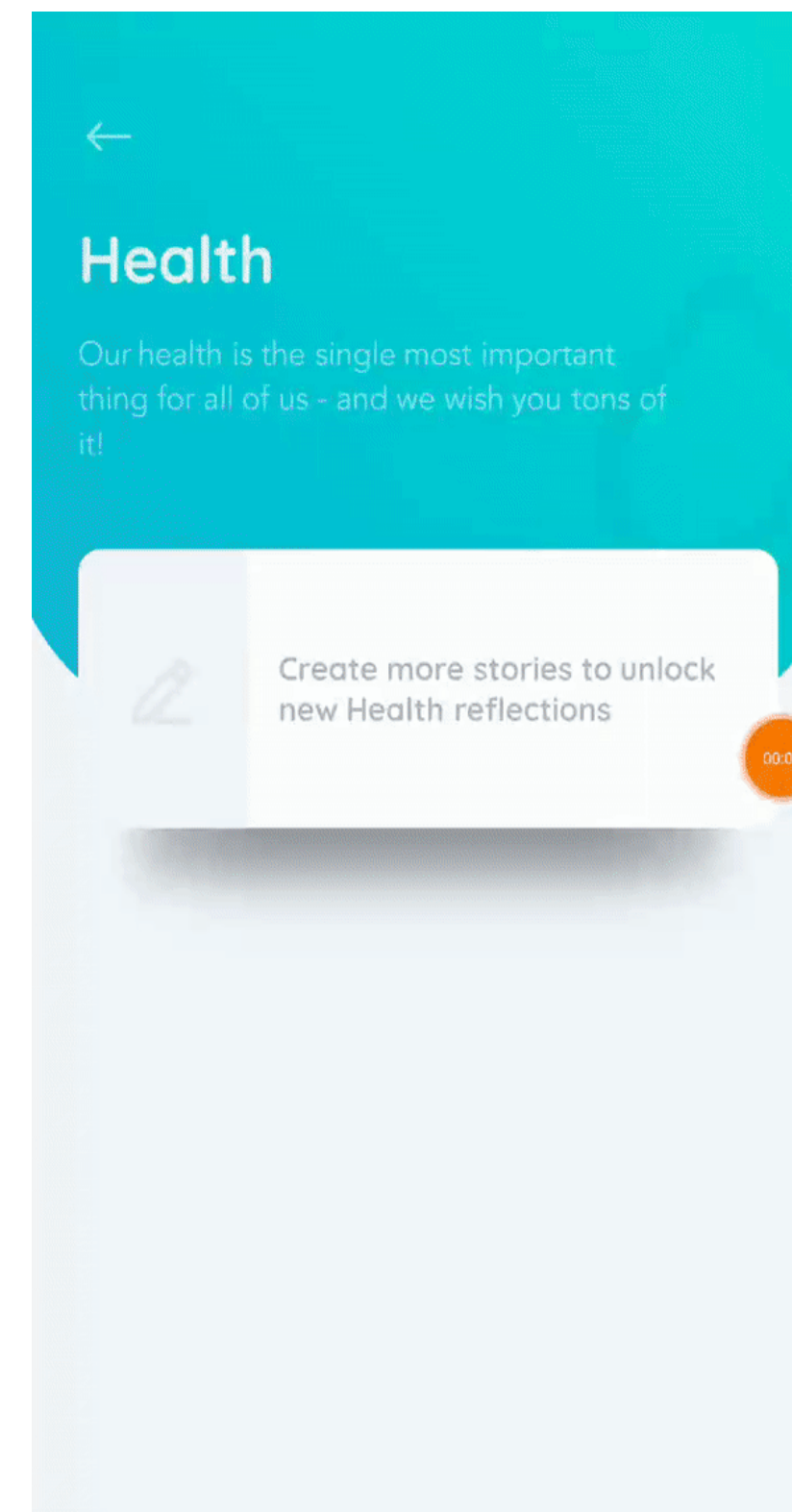
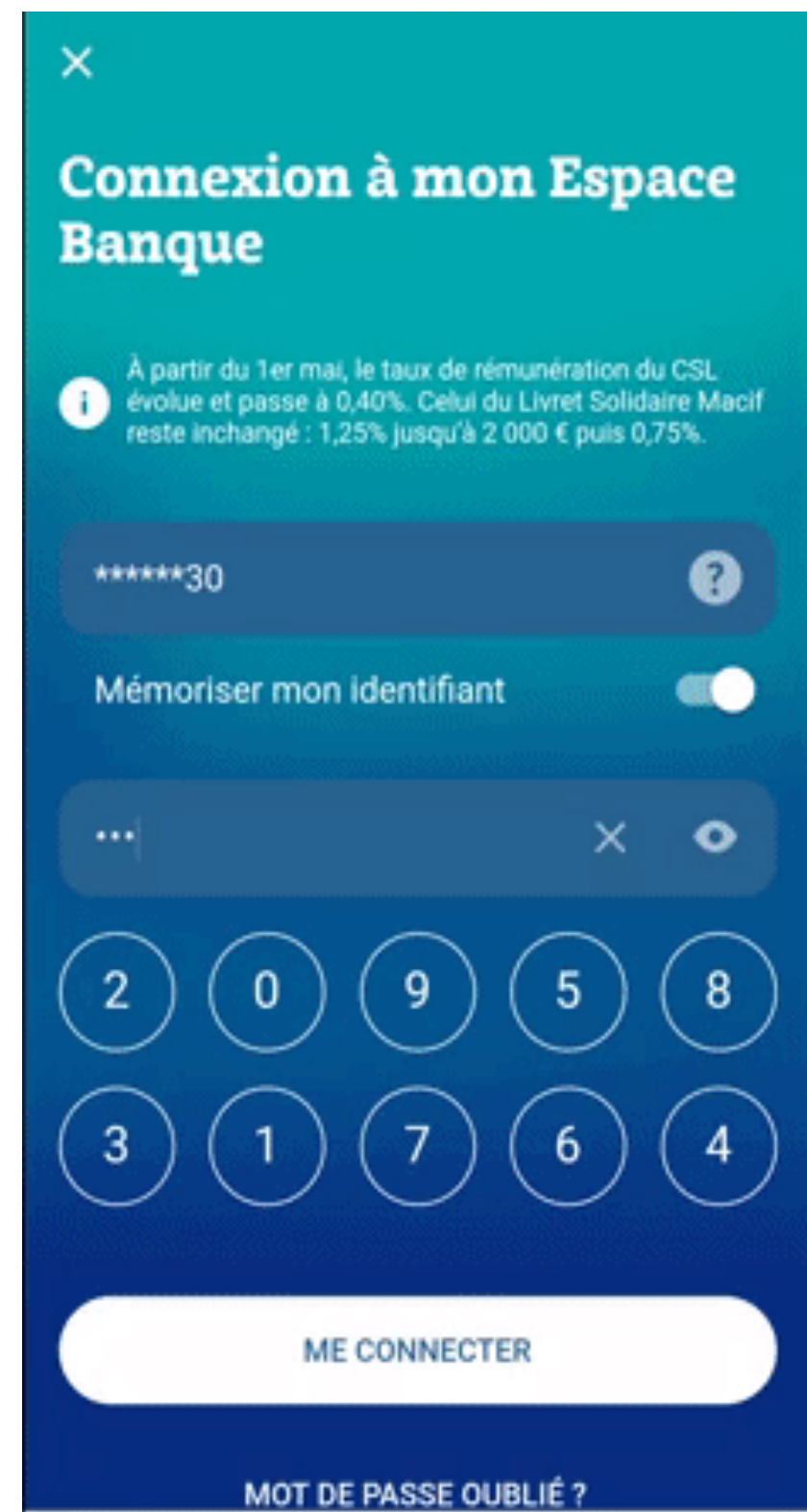
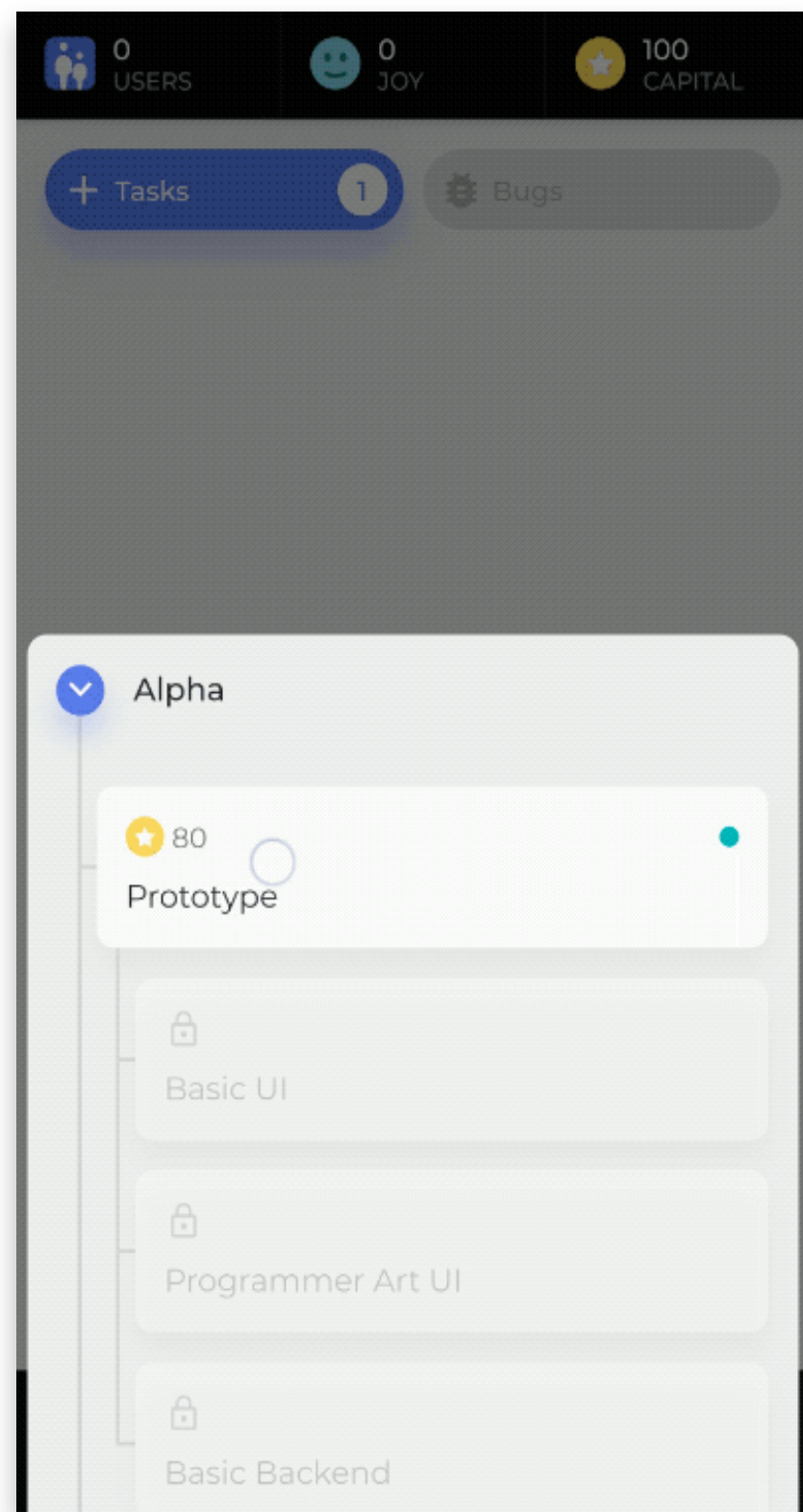
1. Flutter 基本介绍和在中国的发展状况
2. 多平台愿景 : Flutter for Web
3. 生态建设 : 与社区共建的状态管理解决方案
4. 开发体验: “UI as Code” – 编程语言和IDE的改进
5. Flutter/Dart 近期展望



## 1.1. 什么是 Flutter？

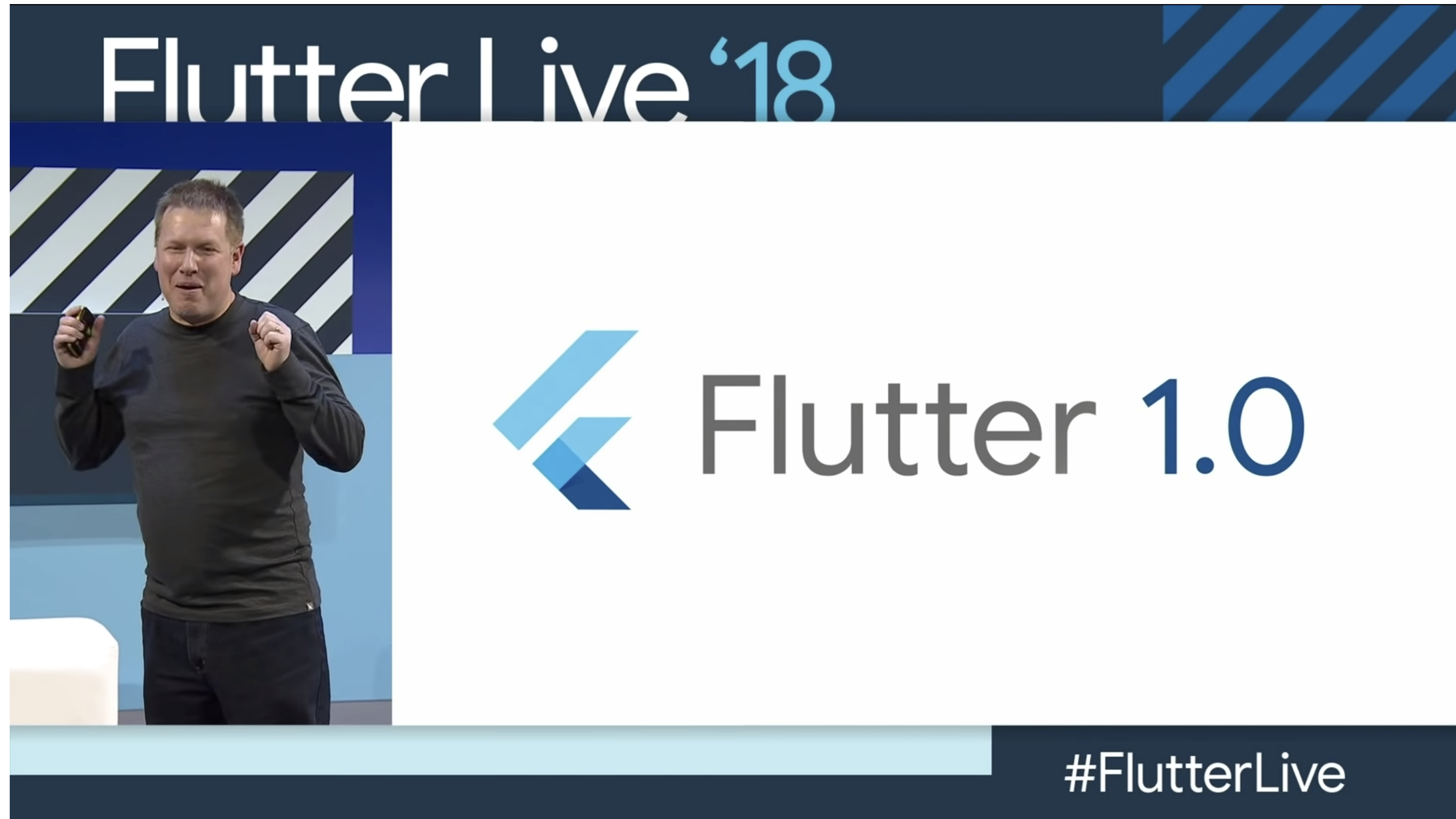


# Flutter 是谷歌主导研发的一个多平台UI工具包





Flutter 在2018年12月发布了1.0



美观

快速

开放

高效





# 美观

对 UI 实现像素级的控制

不需要对设计师说“不”

做有品牌特色的app



# 快速

编译成原生机器代码

每秒 60 帧渲染速率, GPU 加速





Your code in Dart

Flutter framework in Dart  
(widgets, gestures, etc)

C++ Flutter engine  
(Skia, Text, Dart runtime, dart:ui)

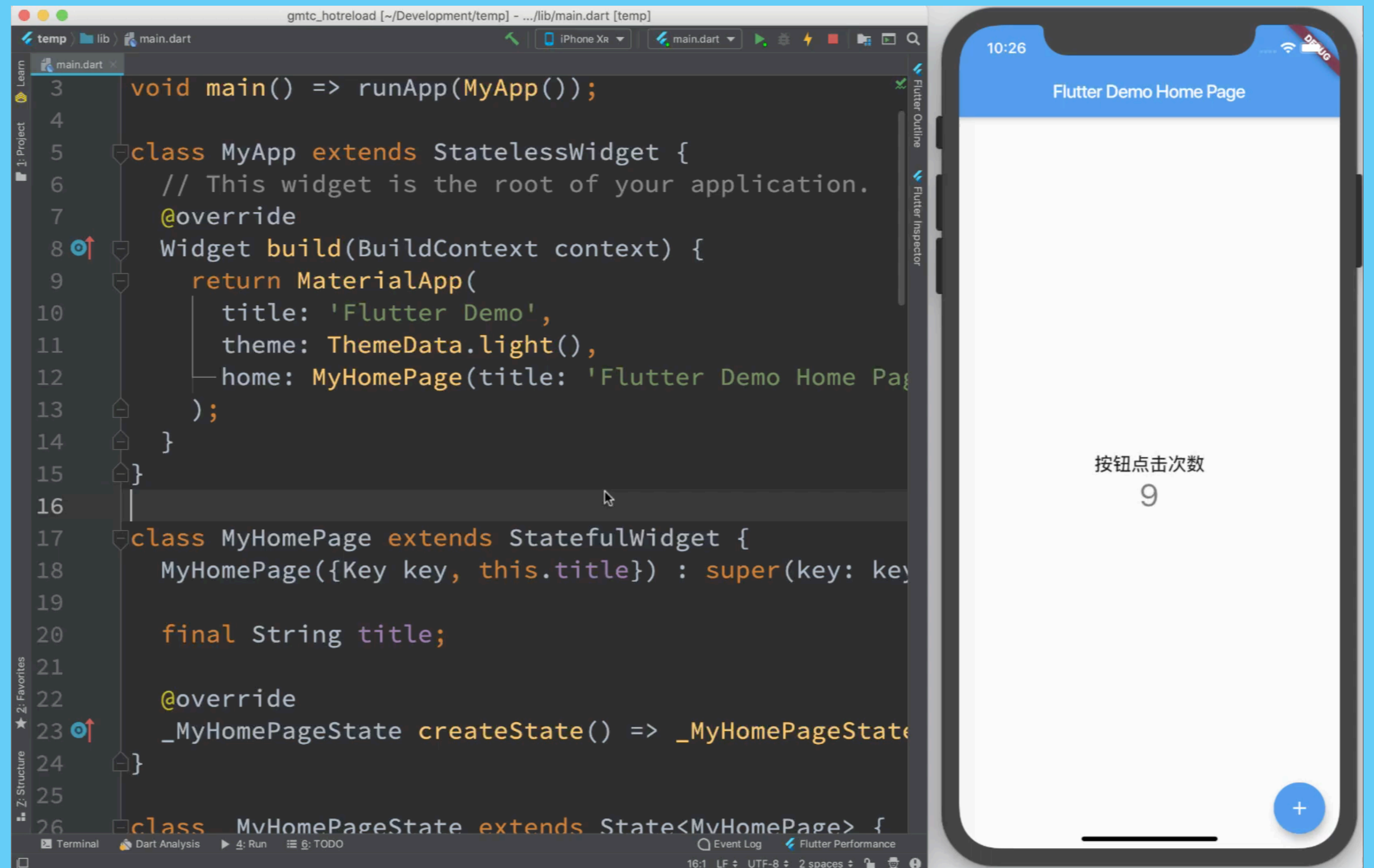
iOS/Android runner (embedder)

Hardware  
GPU, ARM, x86 chips

# 高效

亚秒级代码热重载 (Hot Reload)

一套代码适用多个平台

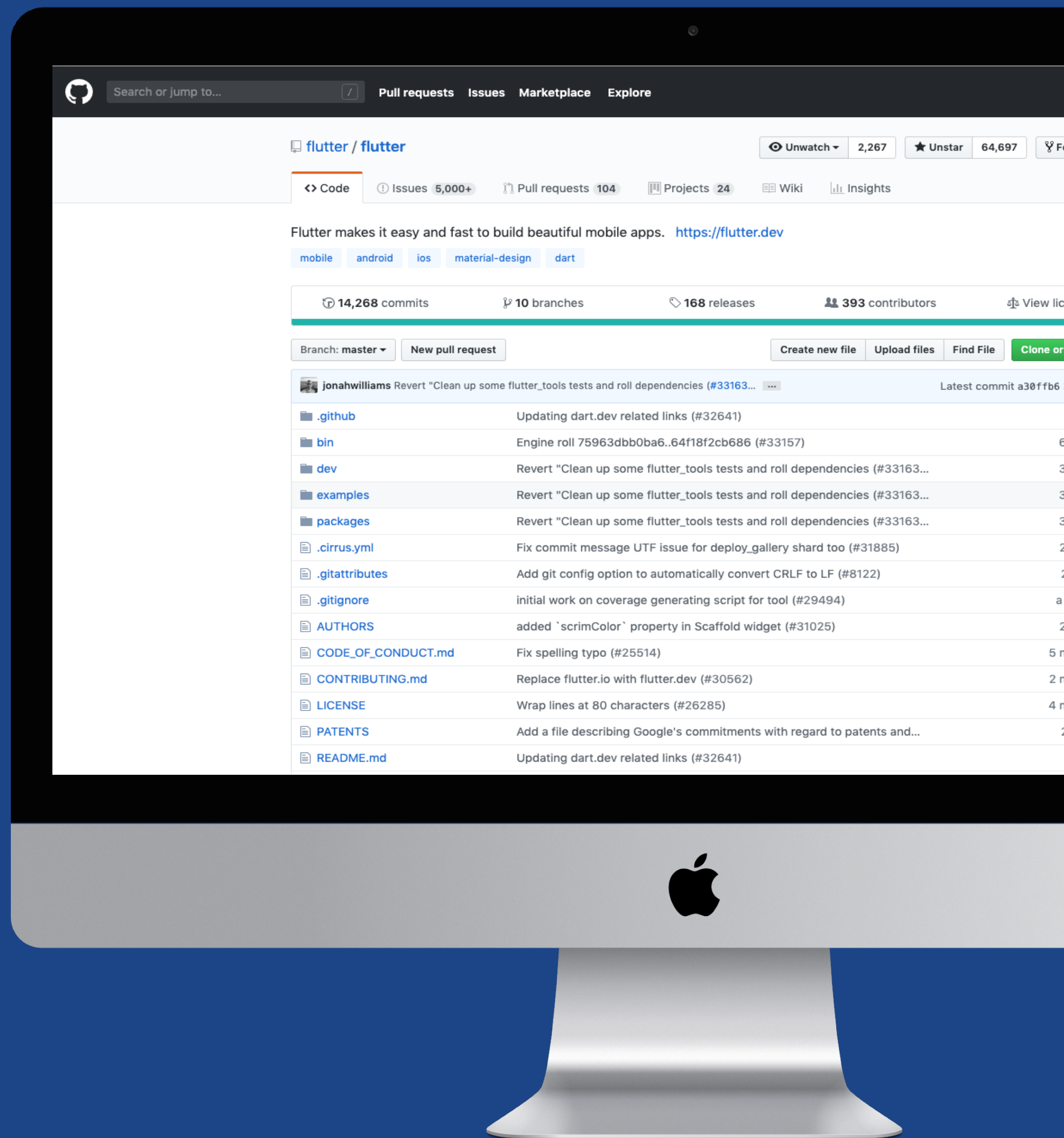




# 开放

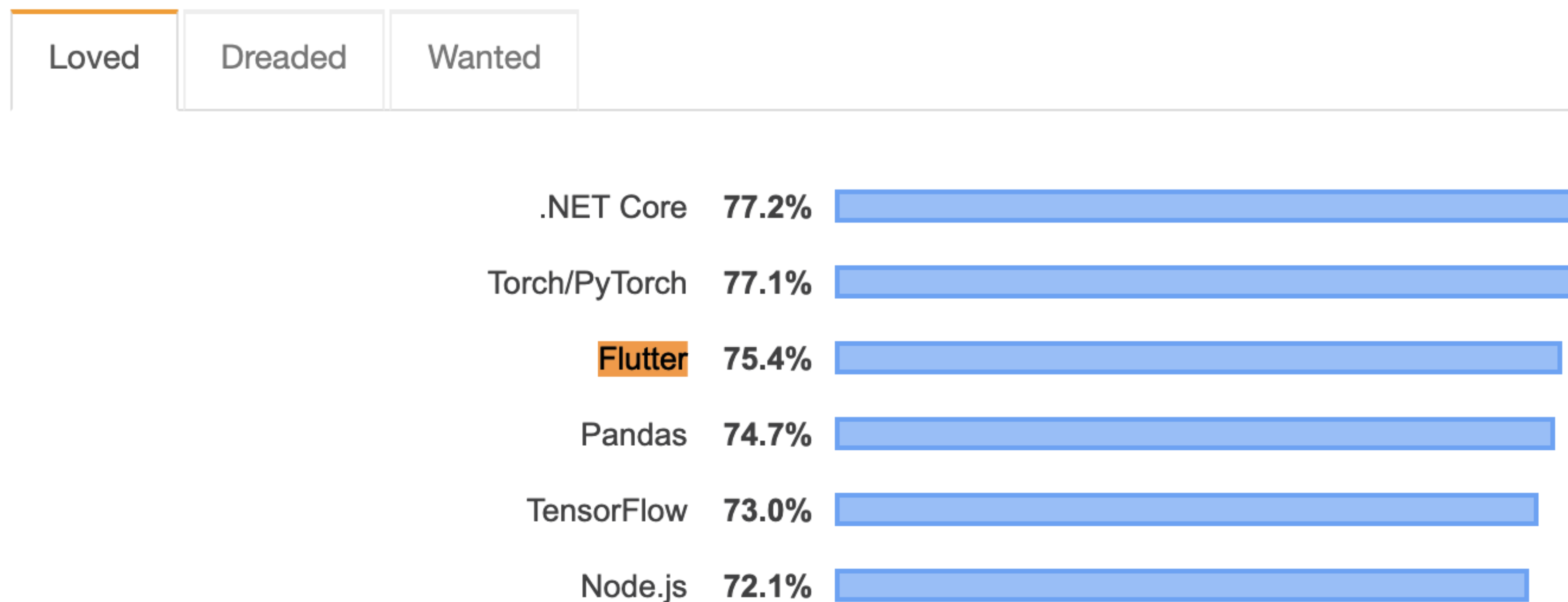
所有代码都免费且开源

快速成长的生态



## 1.2. Flutter 发展状况

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




StackOverflow 2019 全球开发者问卷调查结果  
Flutter 是最受开发者欢迎的框架之一

Google

GROUPON®

Capital One

 Square

Tencent 腾讯

H★MILTON

 JD. 京东  
.COM

 Abbey Road  
Studios

 Alibaba Group

ebay™

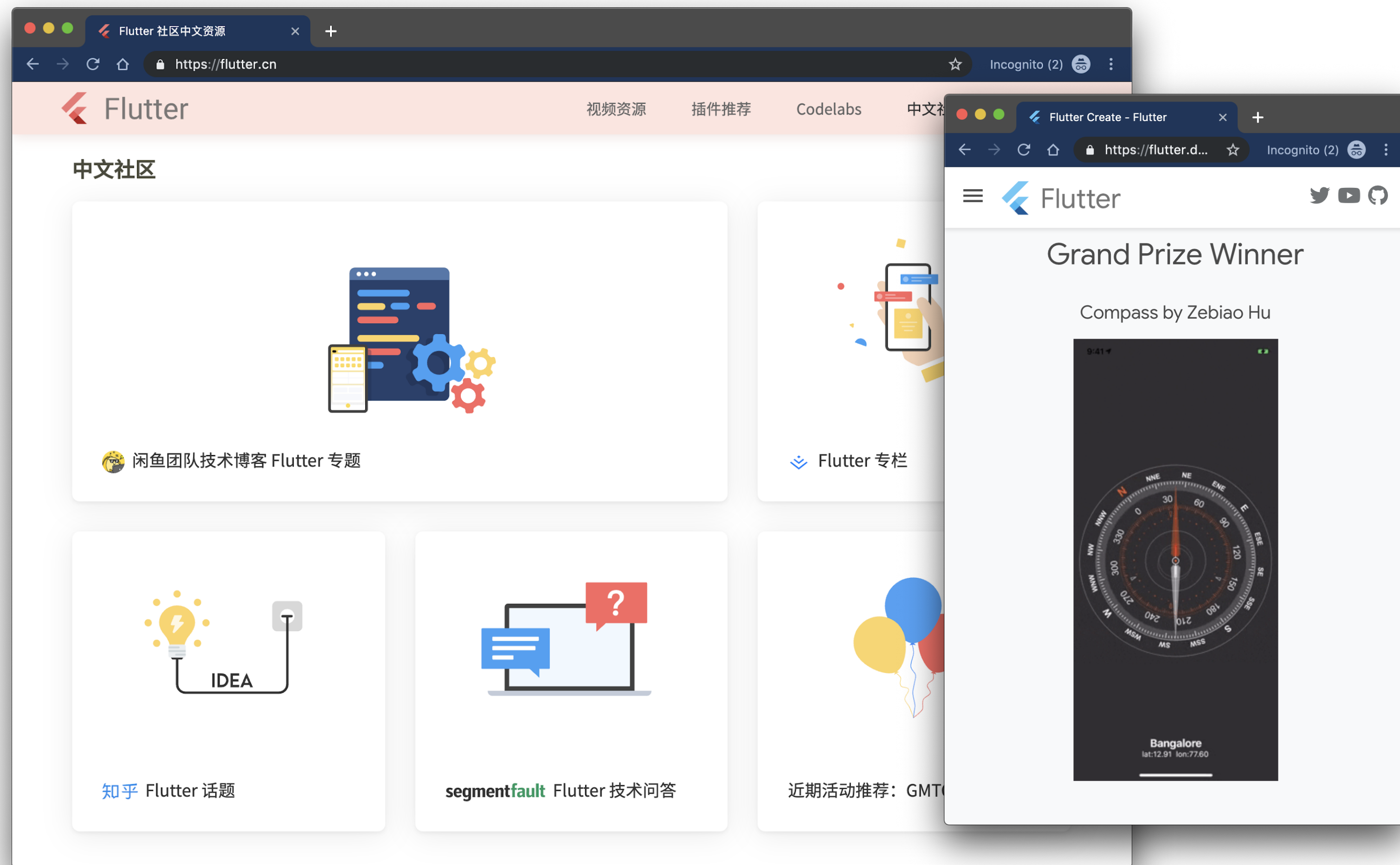
 MACIF





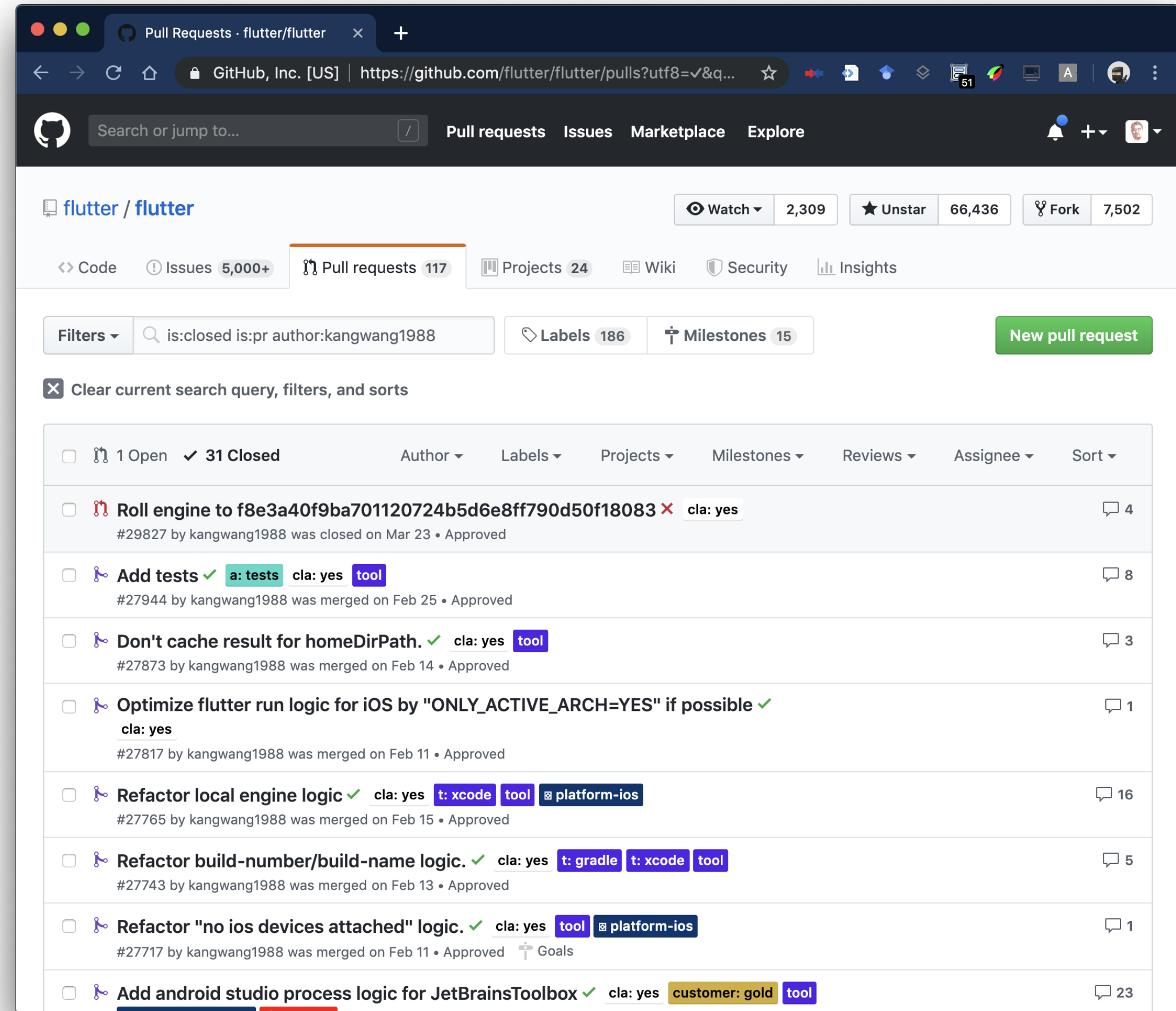
# 1.3. Flutter 中国社区发展状况

- 贡献了大量高质量的技术文章
- 翻译了Flutter官方文档
- 参与到了全球Flutter社区的活动中



# 中国开发者对 Flutter 开源项目的贡献

- Flutter 是一个全球开源项目
- 来自多个公司的中国开发者给 Flutter 代码库提交了 Pull Request
- 我们欢迎每一位开发者通过多种形式来参与到 Flutter 的开发过程中，比如提交 issues 和 PR





Mobile



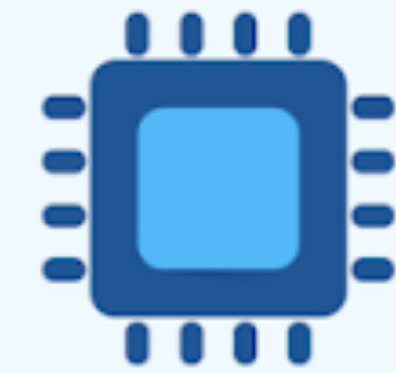
Web



Flutter



Desktop



Embedded



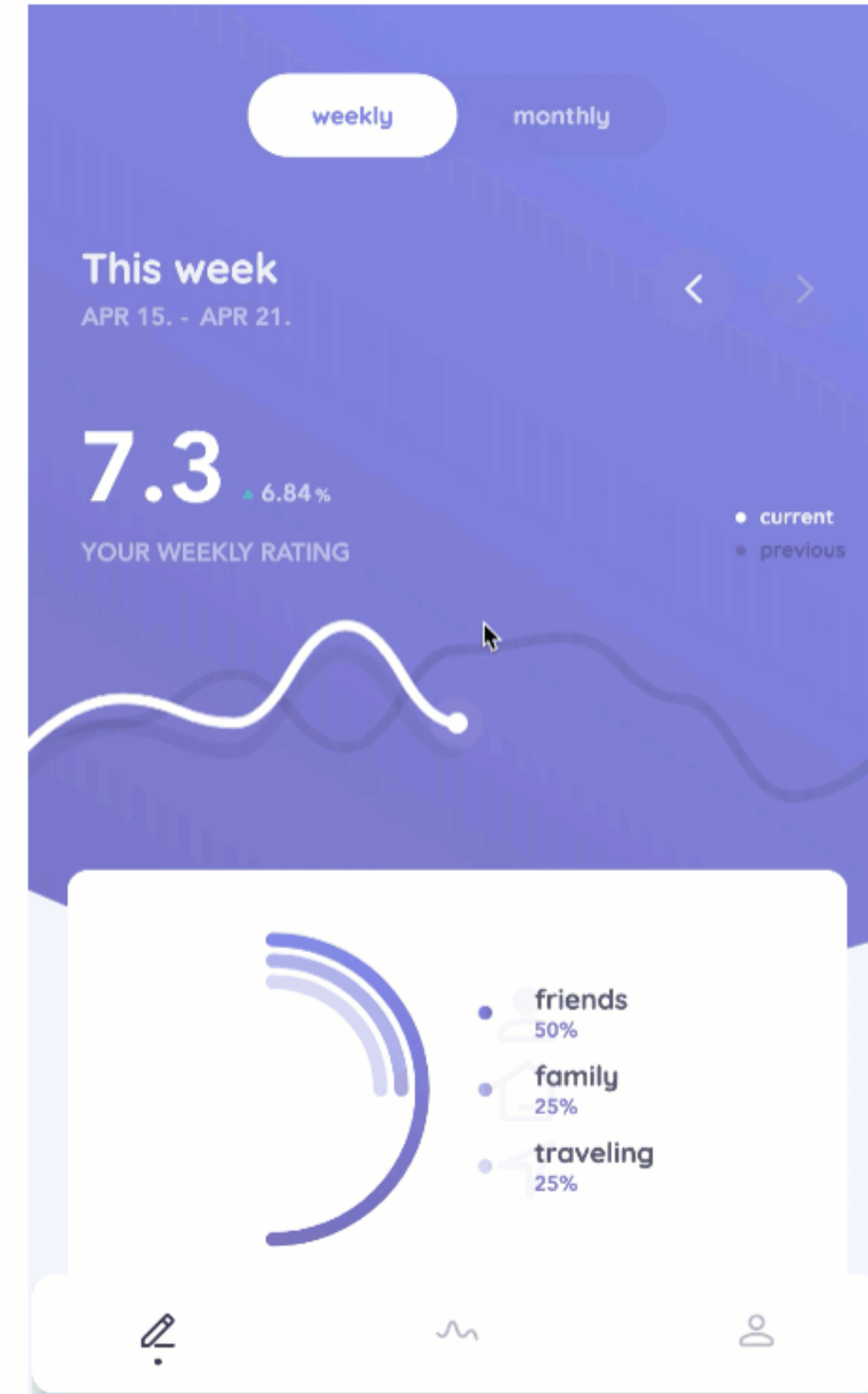
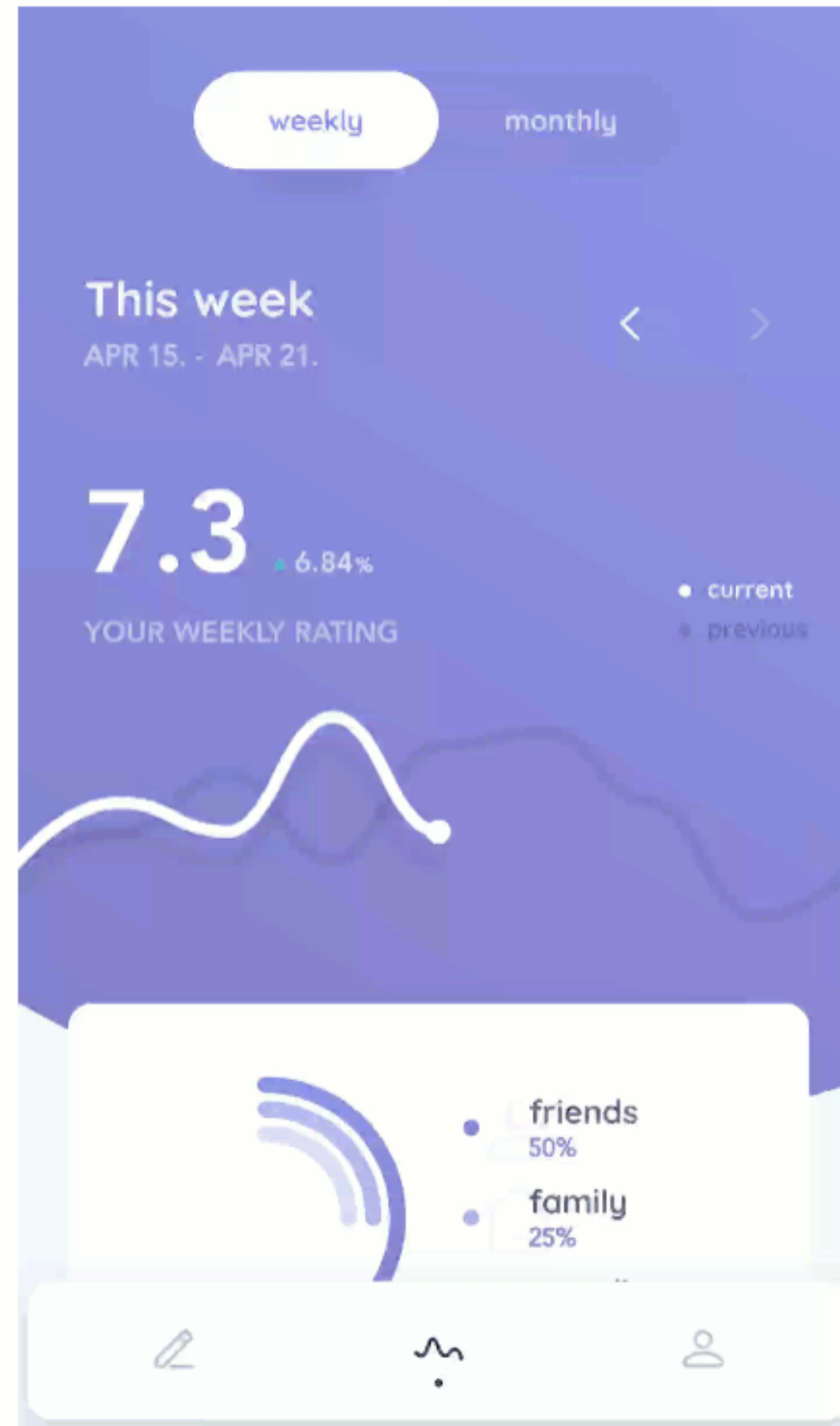


## 2. Flutter for Web 技术预览

[flutter.dev/web](https://flutter.dev/web)

# 哪一个 app 在 Web 浏览器里运行？

Reflectly.app





# Flutter for Web 原理

The screenshot shows the GitHub repository page for flutter/flutter\_web. At the top, it displays the repository name, a 'Watch' button with 178 subscribers, a 'Star' button with 3,474 stars, and a 'Fork' button with 183 forks. Below this, there are navigation tabs for 'Code', 'Pull requests' (1), 'Wiki', 'Security', and 'Insights'. A message says 'Bring your Flutter code to web browsers' with a link to https://flutter.dev/web. A summary bar shows 154 commits, 2 branches, 0 releases, 9 contributors, and the BSD-3-Clause license. Action buttons include 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find File', and 'Clone or download'. A commit history table is visible, with the latest commit by mdebbbar and kevmoo. Below the table, the README.md file is highlighted.

flutter / flutter\_web

Watch 178 Star 3,474 Fork 183

Code Pull requests 1 Wiki Security Insights

Bring your Flutter code to web browsers <https://flutter.dev/web>

154 commits 2 branches 0 releases 9 contributors BSD-3-Clause

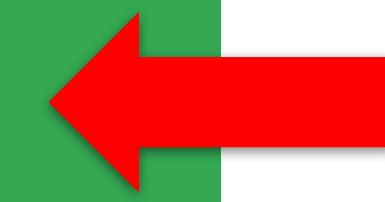
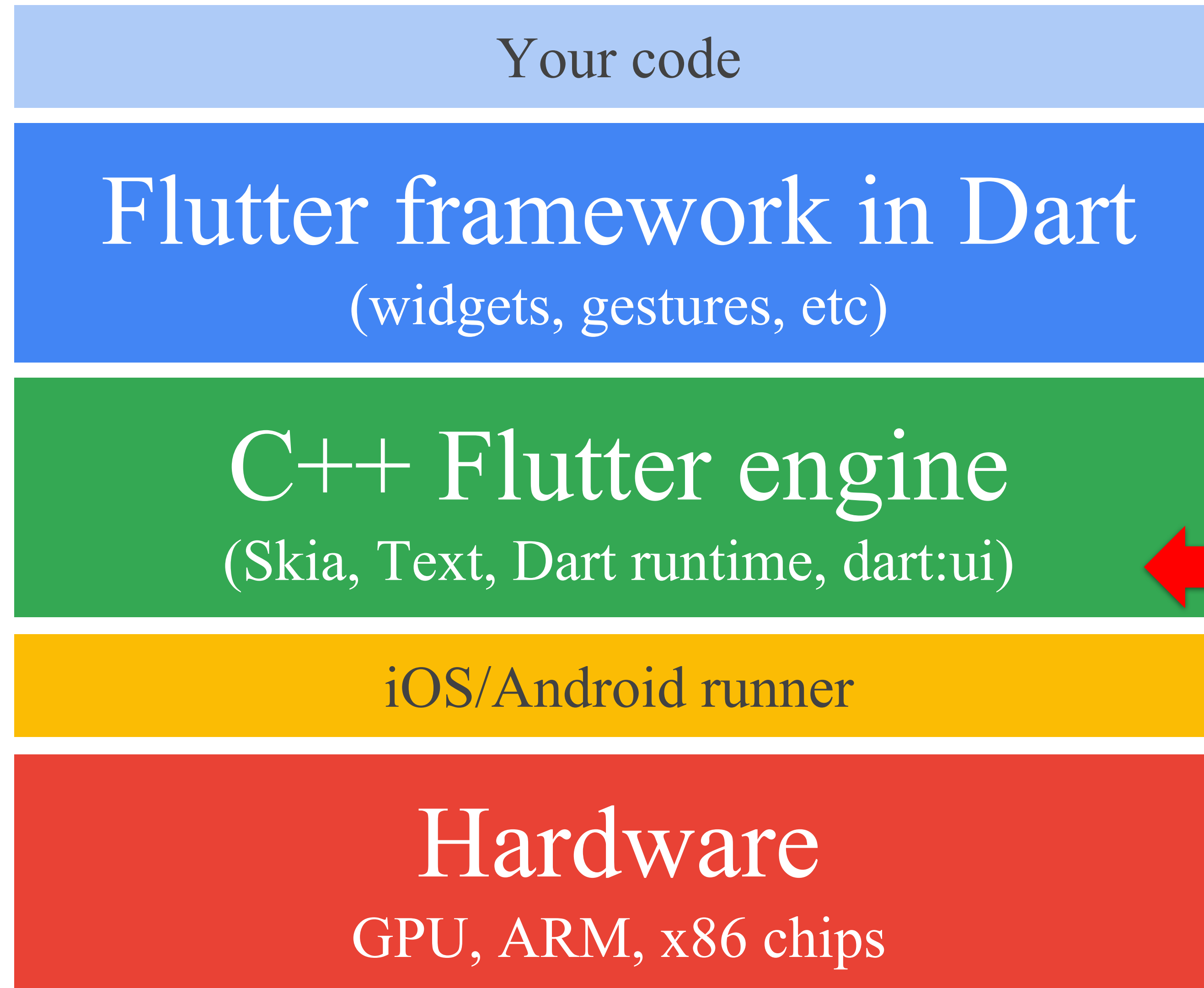
Branch: master New pull request Create new file Upload files Find File Clone or download

mdebbbar and kevmoo Full handling of text overflow in canvas-based text measurement. ... Latest commit 7f588d1 7 hours ago

.github	Project import generated by Copybara.	3 months ago
docs	Fix migration guide link in FAQ	14 days ago
examples	Examples: update pubspec dependencies	26 days ago
packages	Full handling of text overflow in canvas-based text measurement.	7 hours ago
tool	More example cleanup	last month
.gitignore	Project import generated by Copybara.	4 months ago
LICENSE	Project import generated by Copybara.	4 months ago
README.md	Update readme: change to flutter pub	14 days ago

README.md

# Flutter for Mobile 架构图



绘制 Picture  
对象到屏幕

# Flutter for Web 架构图

和移动端共  
享代码

Your code

Flutter framework in Dart  
(widgets, gestures, etc)

Web 特有的  
代码

Flutter *Web* engine  
(dart:ui)

Dart2js compiler

Web 代码执  
行环境

Browser

Hardware



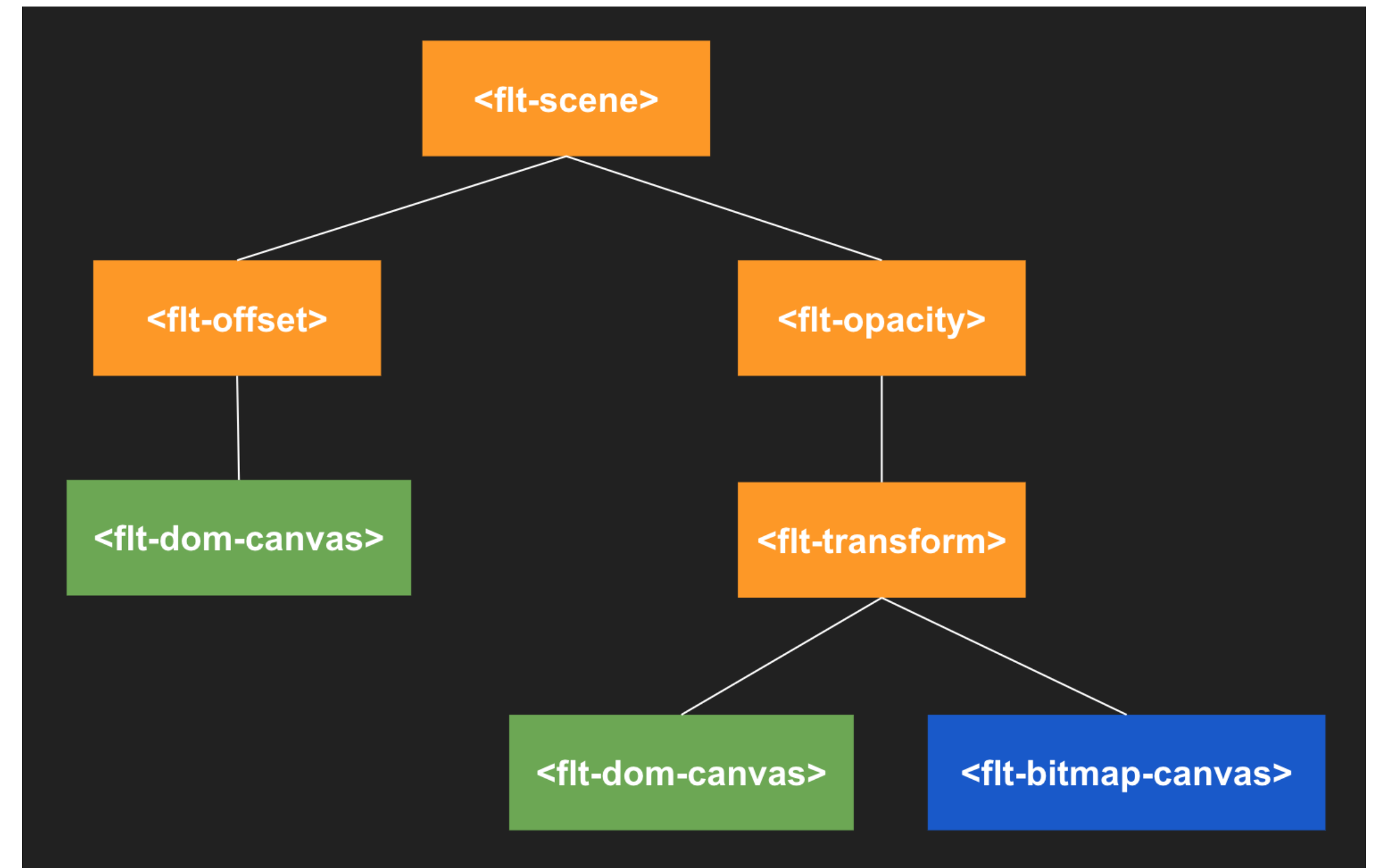
# Flutter Web engine 的任务

- 使用合适的 Web API 绘制图片对象
  - HTML + CSS + Canvas
  - CSS Paint API (有待浏览器支持)

Flutter *Web* engine  
(dart:ui)

# HTML+CSS+Canvas

- 优先使用 HTML + CSS 来绘制图形
  - 可以利用浏览器自身的性能优化
  - 不用担心放大页面时会像素化
- 用 Canvas 2D 作为一个备用绘图选项
- 未来可能用 CSS Paint API 代替 Canvas 以提高性能



绿色节点采用的 HTML / CSS 绘图，蓝色节点采用 Canvas 绘图

# Flutter Web engine 不做的任务

- Flutter Web 引擎依赖 Flutter 框架来：
  - 构建 widgets
  - 实现 UI 布局

Flutter framework in  
Dart

(widgets, gestures, etc)

Flutter *Web* engine

(dart:ui)



Your Flutter code (Dart)



Flutter *Web* engine  
(dart:ui)



Dart code for HTML, CSS, Canvas



Dart2js compiler



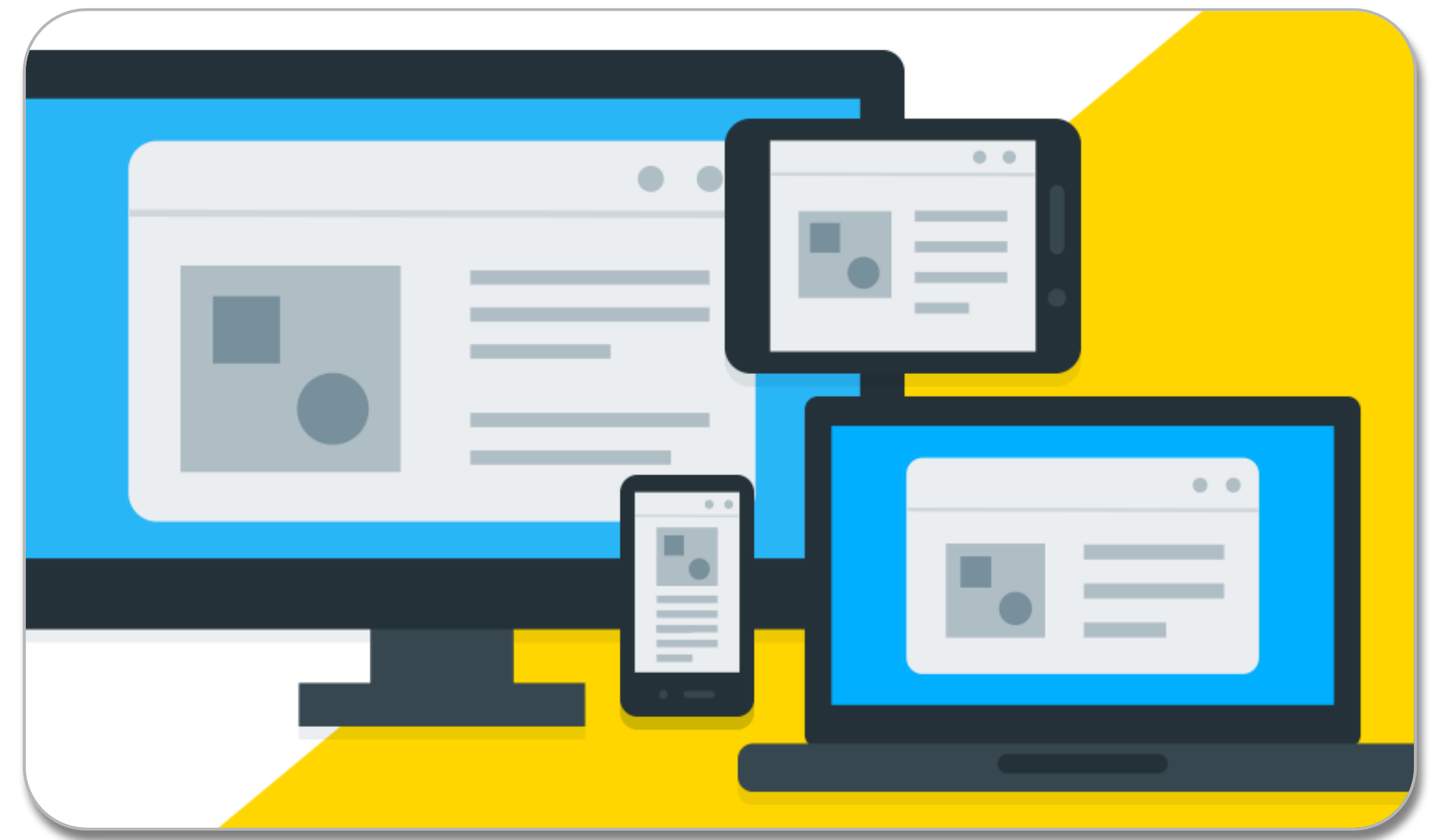
JavaScript



Browser

# Flutter for Web 使用场景

- Flutter for Web 的定位是一个 **Web** 应用框架
- 建议使用场景：
  1. 开发移动应用的 **Web** 端配套应用
  2. 利用现有 **Flutter** 项目中的交互元素



# 开发移动应用的 Web 端配套应用

- 覆盖更多用户
- 先体验后下载 app
- 使用桌面端的输入输出设备



# 利用现有 Flutter 项目中的交互元素

- 交互式数据图表
- 小游戏
- 工具
  - 比如日历、汽车配置、投资组合分析等

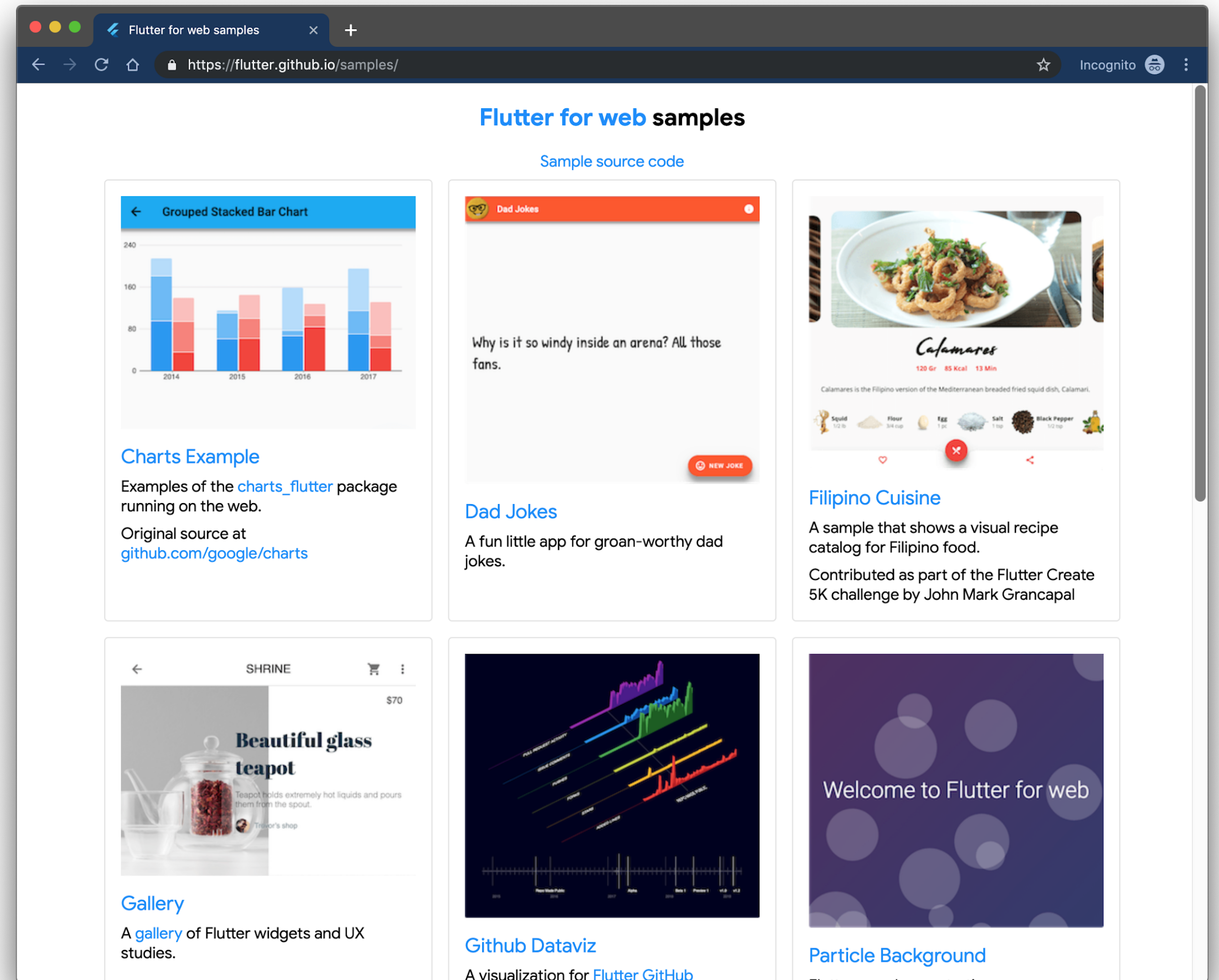




# Flutter for Web 当前的性能

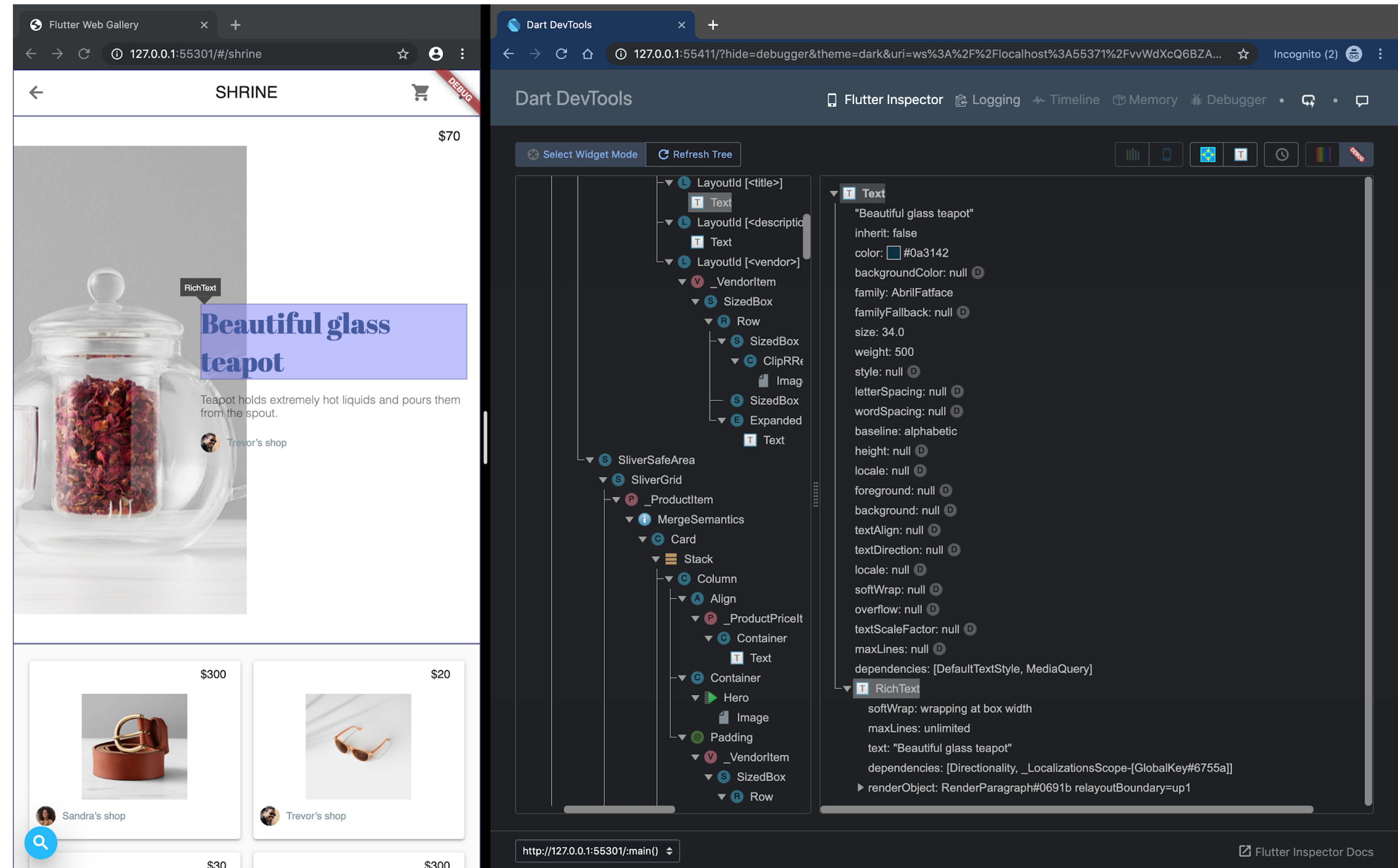
flutter.github.io/samples

- 示例应用在桌面浏览器基本达到 60 FPS
- 移动浏览器，特别是在低端机型上尚有很大的优化空间



# Flutter for Web 当前的开发体验

- Widget API 和移动端一致
- 使用 Dart DevTools 进行调试
- 支持 (Stateless) Hot Reload



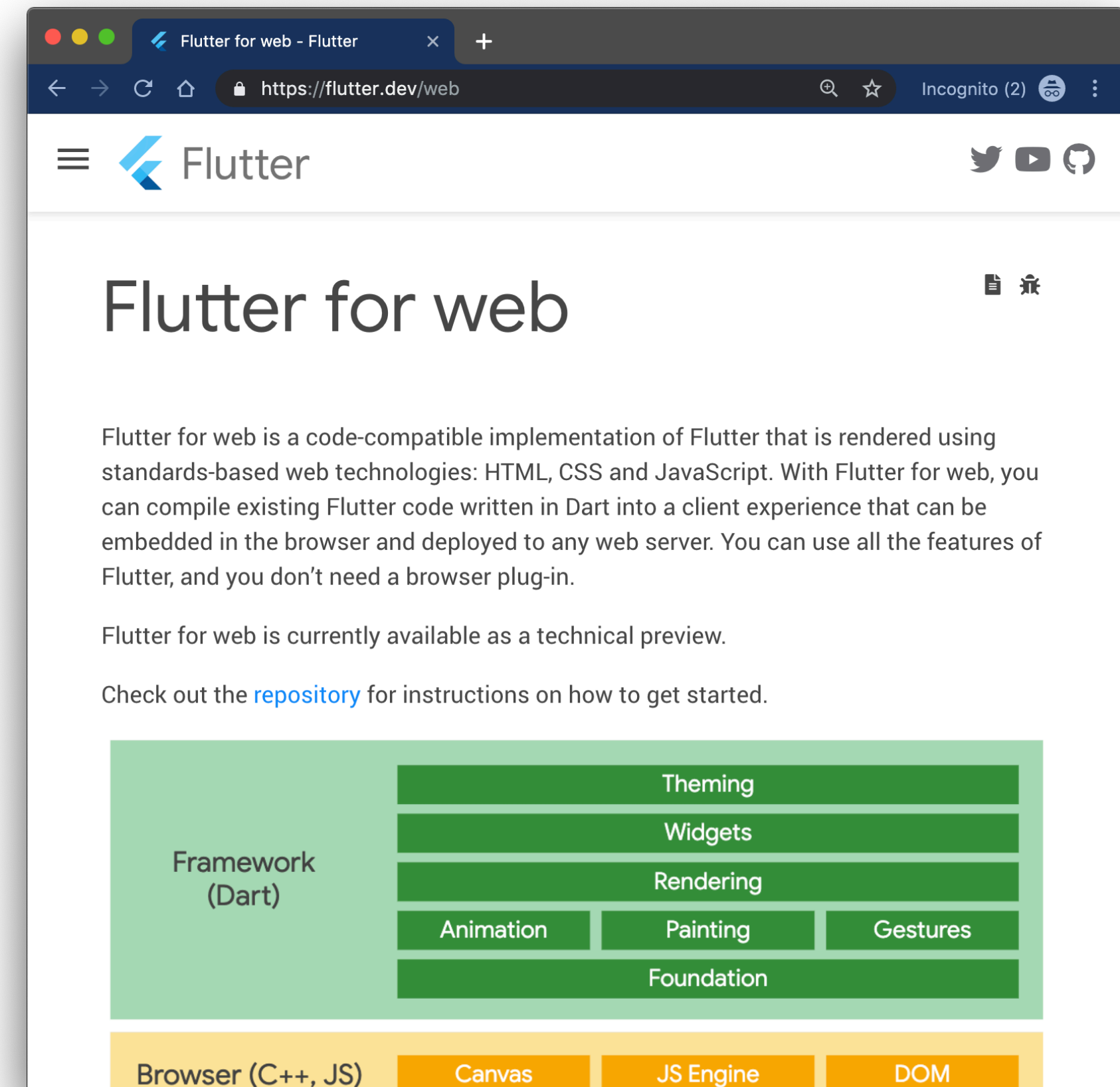


# Flutter for Web 目前的局限和主要研发工作

Flutter for Web 是一个技术预览，不建议在生产环境使用。

当前主要研发工作：

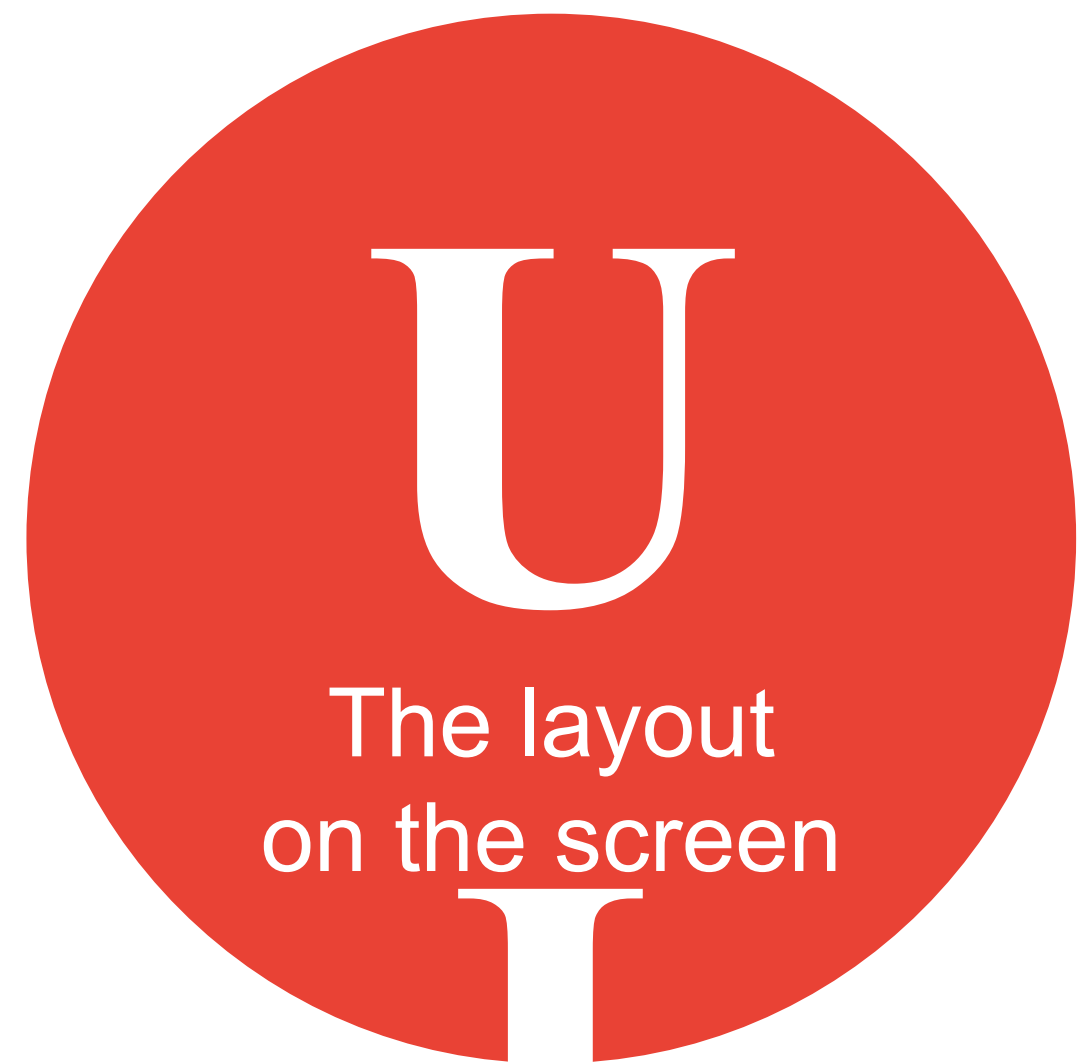
- 合并代码到 **Flutter SDK**
- 保证所有 **widgets** 都能正确渲染
- 优化性能
- 完善对浏览器无障碍功能的支持
- 添加插件系统来使用现有的**JS**库
- 加强开发调试工具的可用性



## 4. 生态建设：与社区共建的状态管理解决方案



# 状态 State 和 UI 的关系

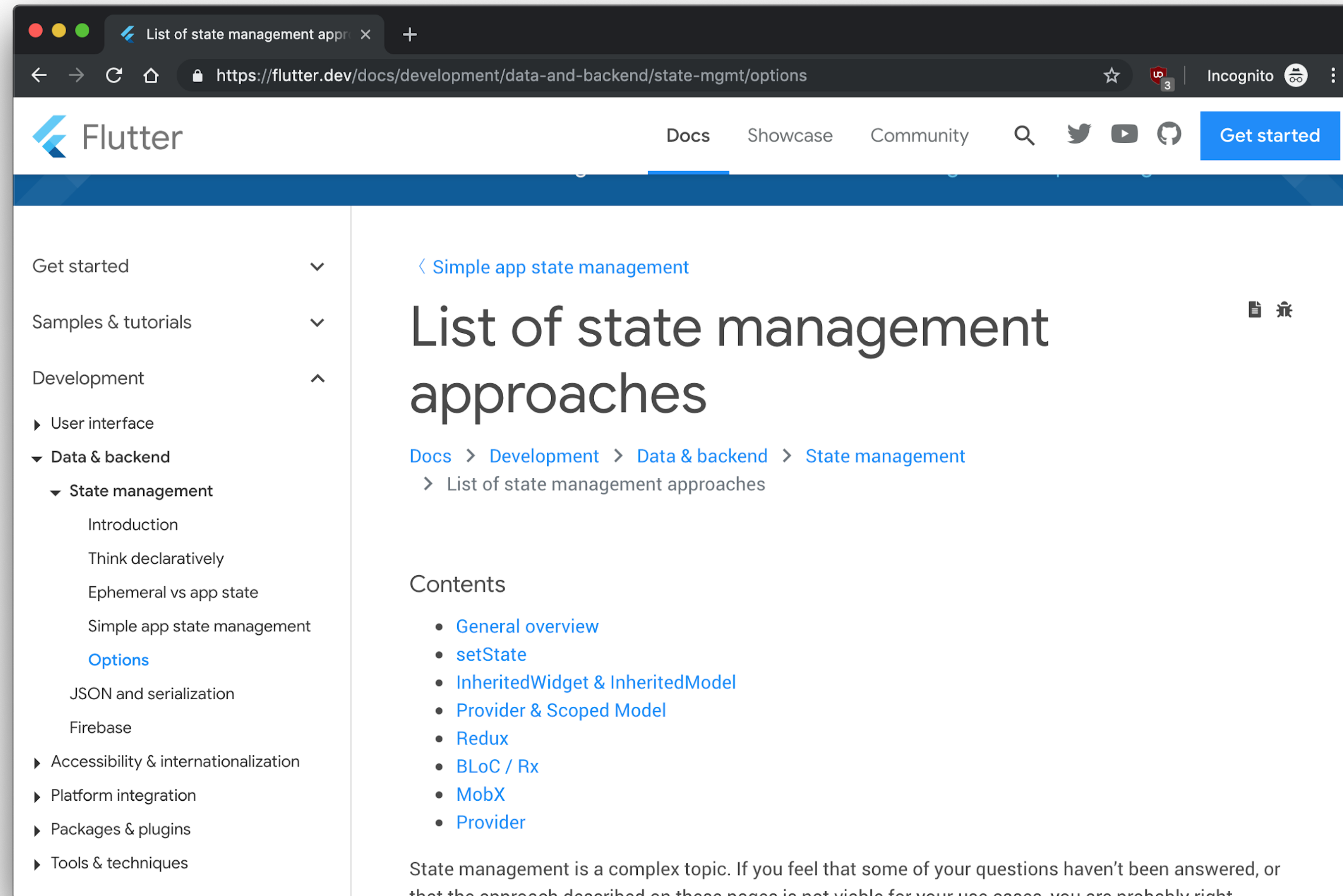


=

Your build  
methods



# Flutter 状态管理解决方案的选择



简单的状态管理方法

package:provider

# 社区贡献的解决方案

The screenshot shows the pub.dev page for the provider 3.0.0 package. The page includes a header with the package name and version, a navigation menu with links to README.md, CHANGELOG.md, Example, Installing, and Versions (99 versions), and a status bar showing build passing, pub v3.0.0, and codecov 100%. The main content area contains a description of the package as a dependency injection system, a migration guide from v2.0.0 to v3.0.0, and a code snippet for the main function. The right sidebar contains an 'About' section, an 'Author' section, an 'Uploader' section, and a 'License' section.

**provider 3.0.0**  
Published Jun 7, 2019 FLUTTER

README.md CHANGELOG.md Example Installing Versions 99

build passing pub v3.0.0 codecov 100%

A dependency injection system built with widgets for widgets. `provider` is mostly syntax sugar for `InheritedWidget`, to make common use-cases straightforward.

### Migration from v2.0.0 to v3.0.0

- Providers can no longer be instantiated with `const`.
- `Provider` now throws if used with a `Listenable / Stream`. Consider using `ListenableProvider / StreamProvider` instead. Alternatively, this exception can be disabled by setting `Provider.debugCheckInvalidValueType` to `null` like so:

```
void main() {  
  Provider.debugCheckInvalidValueType = null;  
  
  runApp(MyApp());  
}
```

- All `XXProvider.value` constructors now use `value` as parameter name.

Before:

**About**  
A dependency injection system built with widgets for widgets. `provider` is mostly syntax sugar for `InheritedWidget`, to make common use-cases straightforward.  
[Repository \(GitHub\)](#)  
[View/report issues](#)  
[API reference](#)

**Author**  
✉ 🔍 Remi Rousselet  
✉ 🔍 Flutter Team

**Uploader**  
✉ 🔍 darky12s@gmail.com  
✉ 🔍 filiph@google.com

**License**  
MIT (LICENSE)

**Dependencies**  
[flutter](#)

More

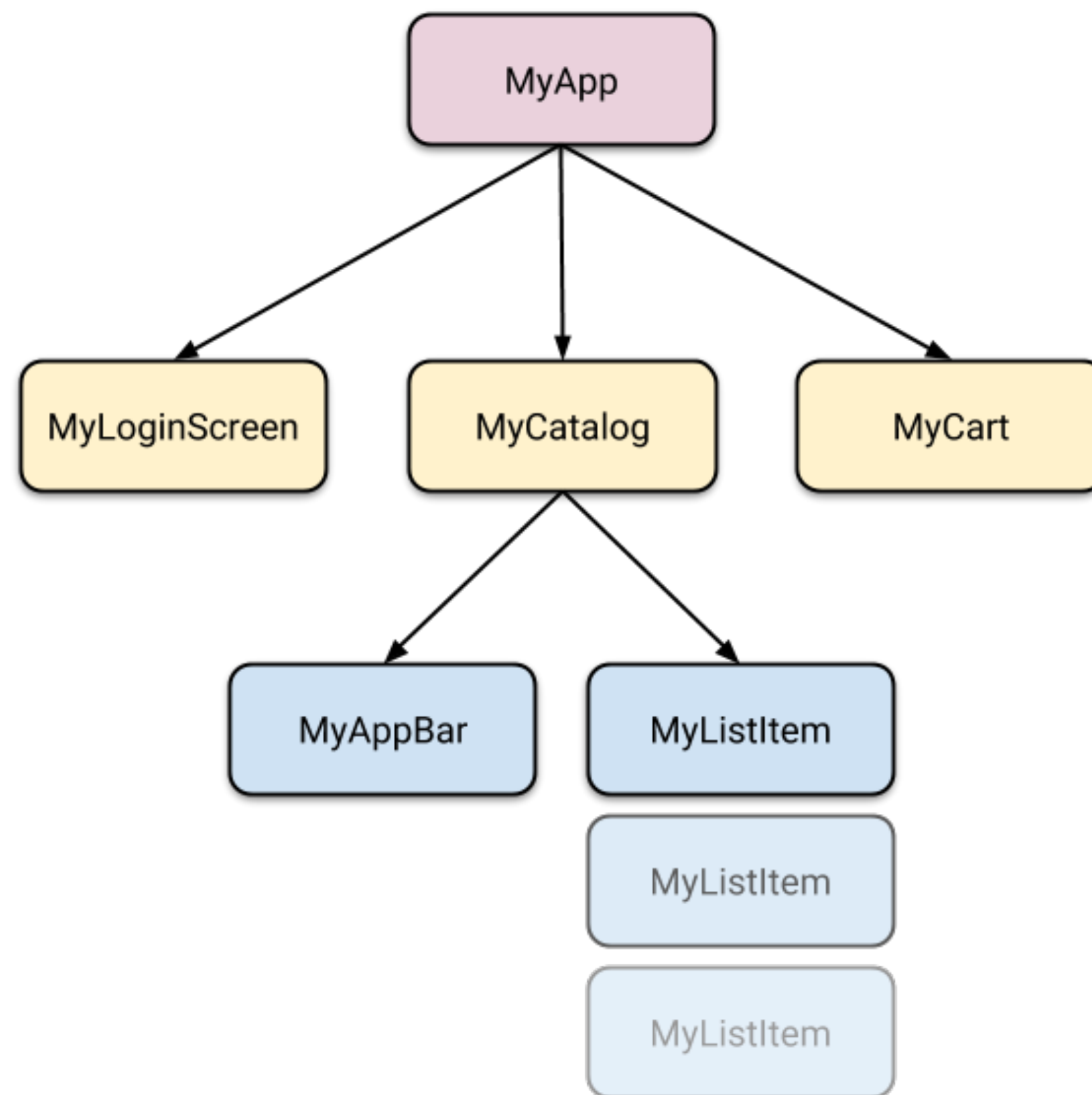
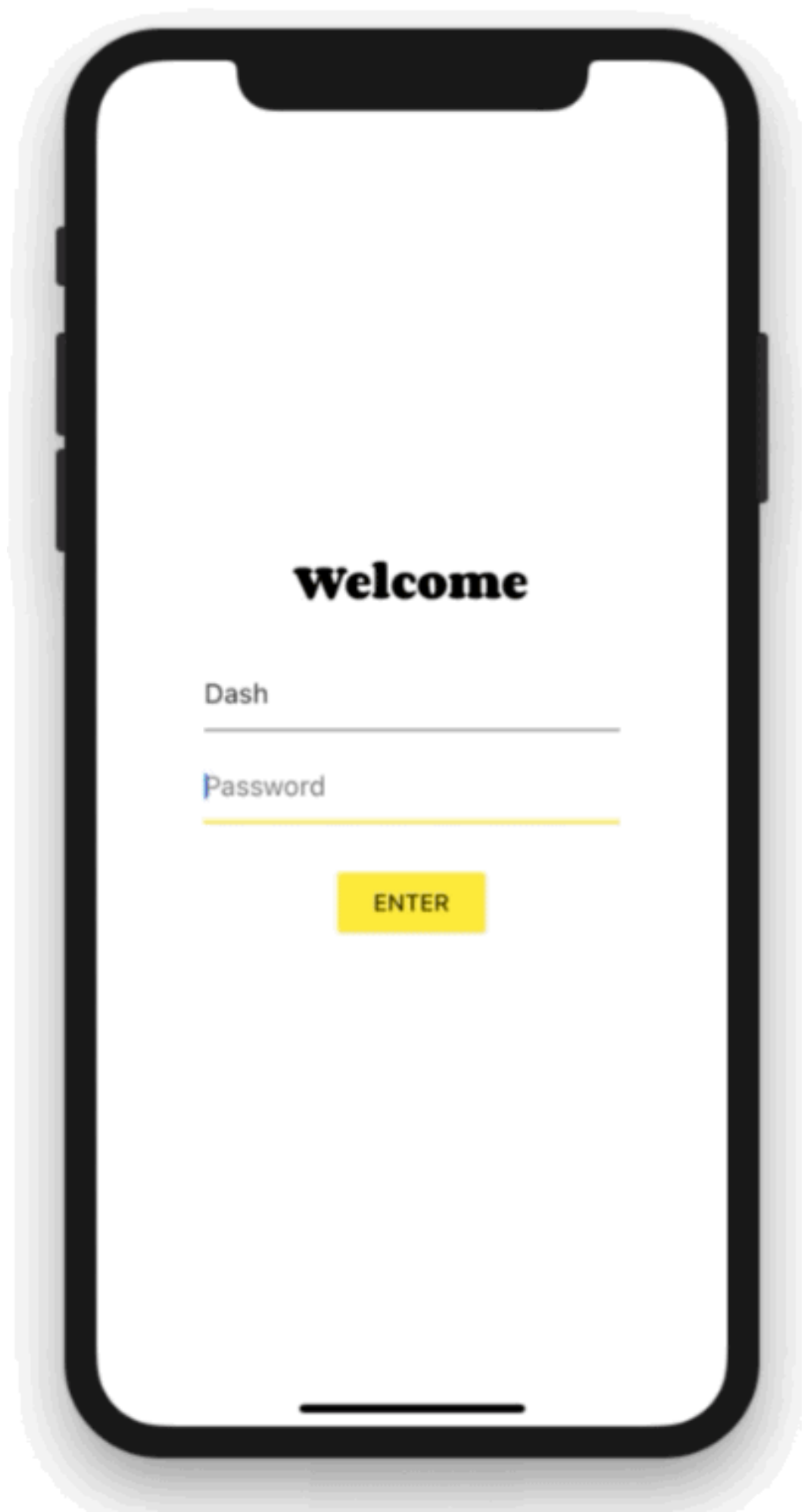
## Author

✉ 🔍 Remi Rousselet

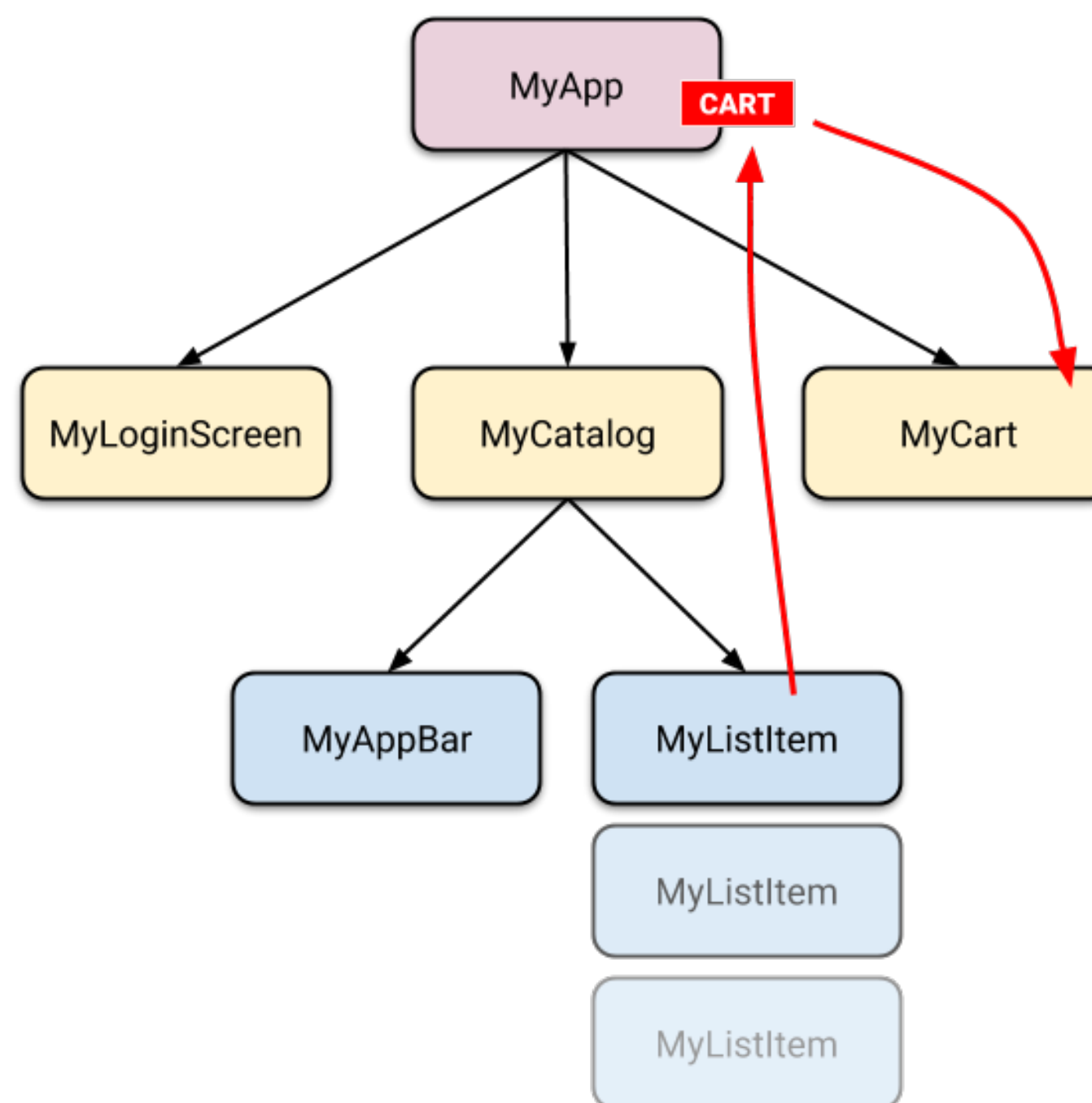
✉ 🔍 Flutter Team



# 状态管理要解决的一个基本问题



# 基本思路：把状态放到组件树的上层节点



# ChangeNotifier

ChangeNotifier 是 Flutter SDK 中的一个简单的类。它用于向监听器发送通知。

```
class CartModel extends ChangeNotifier {  
  /// Internal, private state of the cart.  
  final List<Item> _items = [];  
  
  /// An unmodifiable view of the items in the cart.  
  UnmodifiableListView<Item> get items => UnmodifiableListView(_items);  
  
  /// The current total price of all items (assuming all items cost $42).  
  int get totalPrice => _items.length * 42;  
  
  /// Adds [item] to cart. This is the only way to modify the cart from outside.  
  void add(Item item) {  
    _items.add(item);  
    // This call tells the widgets that are listening to this model to rebuild.  
    notifyListeners();  
  }  
}
```



# ChangeNotifierProvider

ChangeNotifierProvider widget 可以返回一个特定类别的 ChangeNotifier 实例并把它提供给下属的 widgets。这个类属于 provider package。

```
void main() {  
  runApp(  
    ChangeNotifierProvider(  
      builder: (context) => CartModel(),  
      child: MyApp(),  
    ),  
  );  
}
```





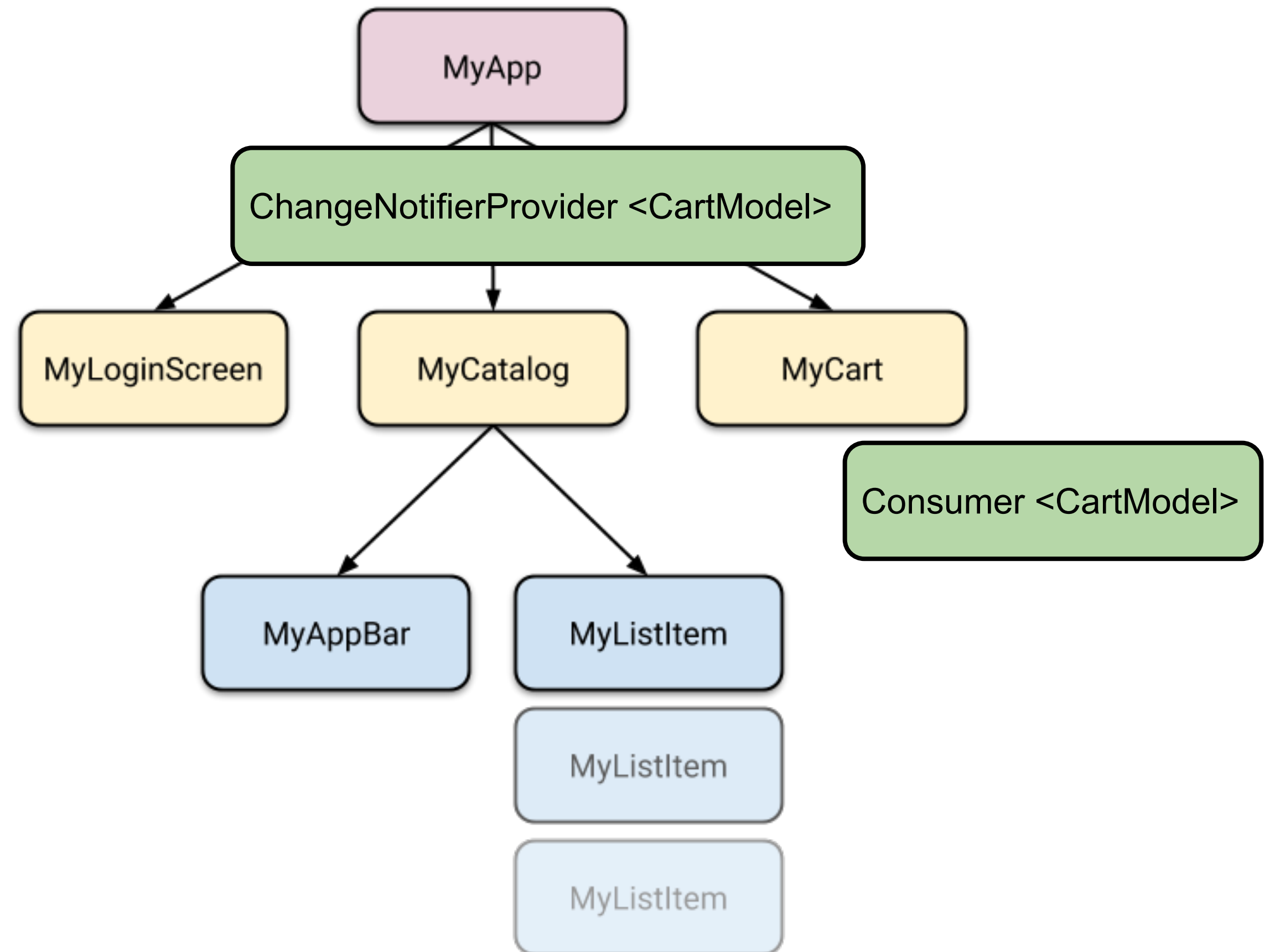
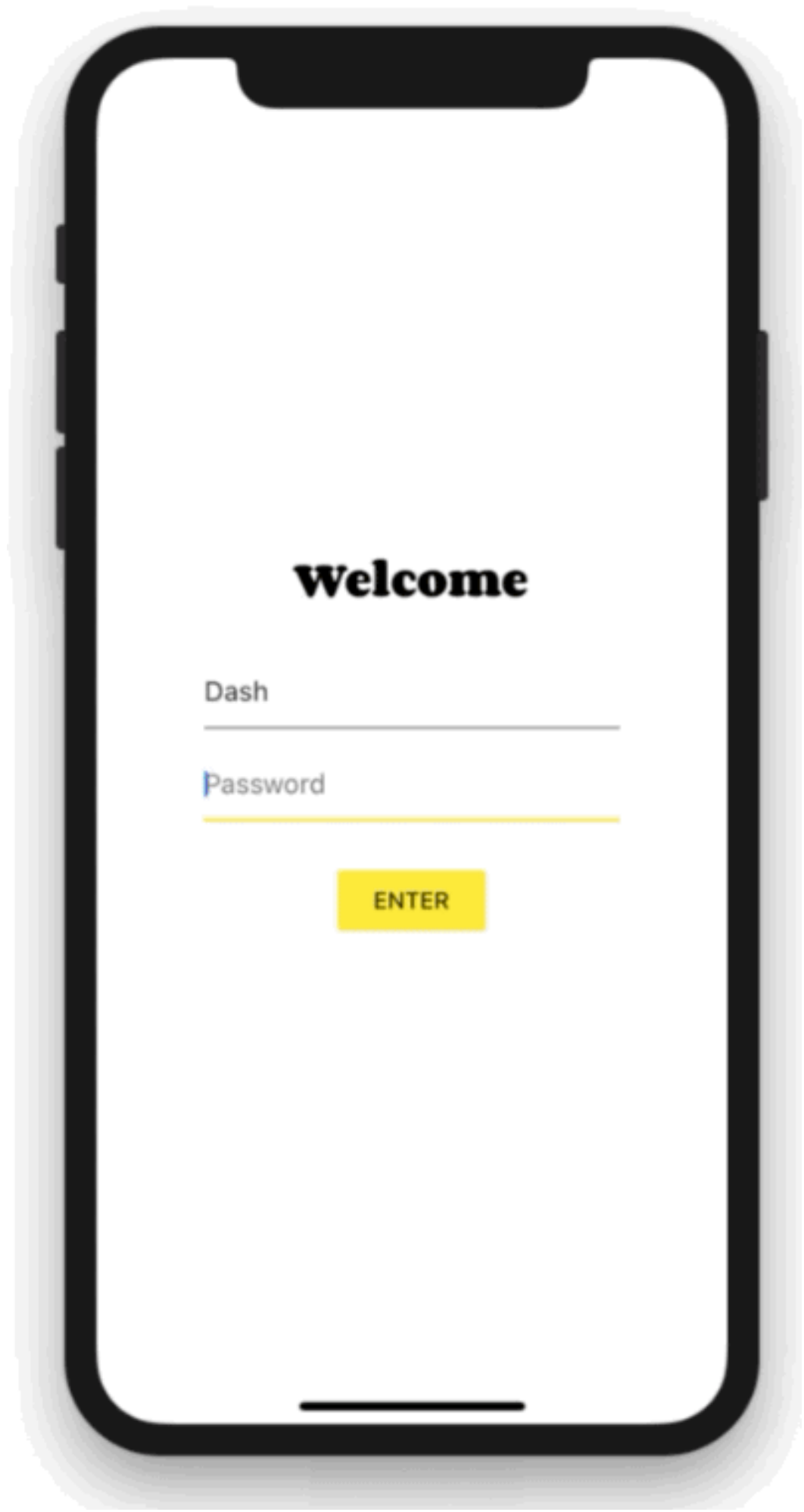
# Consumer

现在 `CartItem` 已经通过 `ChangeNotifierProvider` 提供给 UI 组件树的下层节点。我们可以用 `Consumer` 这个类来调用它。

```
return Consumer<CartItem>(
  builder: (context, cart, child) {
    return Text("Total price: ${cart.totalPrice}");
  },
);
```



# Provider 状态管理方案小结



<https://flutter-io.cn/docs/development/data-and-backend/state-mgmt/simple>

## 4. Flutter 开发体验: “UI as Code”

- UI-as-Code 是旨在提高 Flutter UI 代码可读性的一系列尝试
- 包括 Dart 语法的改进和代码编辑器 UI 的改进

# 用 Dart 代码编写 Flutter UI 的优点和改进空间

优点:

- Flutter 的声明式 UI 编写方法能够直观地描述 UI 结构
- 不需要为 UI 布局学习额外的语法
- 不需要维护代码之外的 UI 定义文件

改进空间:

- 复杂 UI 逻辑会使用命令式语法, 打破代码结构和 UI 视觉结构的一致性
- 多层嵌套之后不容易对 UI 的组成部分做到一目了然

```
@override
Widget build(BuildContext context) {
  return Container(
    height: 56.0, // in logical pixels
    padding: const EdgeInsets.symmetric(horizontal: 8.0),
    decoration: BoxDecoration(color: Colors.blue[500]),
    // Row is a horizontal, linear layout.
    child: Row(
      // <Widget> is the type of items in the list.
      children: <Widget>[
        IconButton(
          icon: Icon(Icons.menu),
          tooltip: 'Navigation menu',
          onPressed: null, // null disables the button
        ),
        // Expanded expands its child to fill the available space.
        Expanded(
          child: title,
        ),
        IconButton(
          icon: Icon(Icons.search),
          tooltip: 'Search',
          onPressed: null,
        ),
      ],
    ),
  );
}
```



# UI-as-Code: 目前比较成功的改进

## Dart 2.3 新语法元素

```
Widget build(BuildContext context) {  
  return Column(children: [  
    Text(mainText),  
    for (var section in sections)  
      HeadingAction(section.heading),  
  ]);  
}
```

## IDE 新功能 : Editor UI Guides

```
return Scaffold(  
  backgroundColor: kShrinePink50,  
  body: SafeArea(  
    child: Container(  
      child: ScopedModelDescendant<AppStateModel>(  
        builder: (context, child, model) {  
          return Stack(  
            children: [  
              ListView(  
                children: [  
                  Row(  
                    children: [  
                      SizedBox(  
                        width: _leftColumnWidth,  
                        child: IconButton(  
                          icon: const Icon(Icons.keyboard_arrow_down),  
                          onPressed: () =>  
                            ExpandingBottomSheet.of(context).close(),  
                        ),  
                      Text(  
                        'CART',  
                        style: localTheme.textTheme.subhead  
                          .copyWith(fontWeight: FontWeight.w600),  
                      ),  
                      const SizedBox(width: 16.0),  
                      Text('${model.totalCartQuantity} ITEMS'),  
                    ],  
                  ),  
                ],  
              ),  
            ],  
          ),  
        ),  
      ),  
    ),  
  ),  
);
```

# Dart 2.3 为 UI 代码优化的新语法元素

- **Control Flow Elements in Collections**  
在集合数据类型的定义中使用 **if** 和 **for** 这样的流程控制元素
- **Spread Operator ‘...’**  
在一个集合变量的定义中，扩展另一个集合变量并把它的元素插入到当前所在的位置

```
Widget build(BuildContext context) {  
  return Row(  
    children: [  
      IconButton(icon: Icon(Icons.menu)),  
      if (!isAndroid) ... [  
        Expanded(child: title),  
        IconButton(icon: Icon(Icons.search)),  
      ],  
    ],  
  );  
}
```

```
Widget build(BuildContext context) {  
  var items = [Text('目录')];  
  // 把章节按分卷放到目录里  
  for (var volume in volumes) {  
    items.addAll(volume.chapters);  
  }  
  
  if (page != pages.last) items.add(Text('下一页'));  
  
  items.add(Text('索引'));  
  
  return Column(children: items);  
}
```

```
Widget build(BuildContext context) =>
  Column(children: [
    Text('目录'),
    for (var volume in volumes)
      ...volume.chapters,
    if (page != pages.last) Text('下一页'),
    Text('索引'),
  ];
```



# // without

```
Widget build(BuildContext context) {  
  var items = [Text('目录')];  
  
  for (var volume in volumes) {  
    items.addAll(volume.chapters);  
  }  
  
  if (page != pages.last)  
    items.add(Text('下一页'));  
  
  items.add(Text('索引'));  
  
  return Column(children: items);  
}
```

# // with ui-as-code

```
Widget build(BuildContext context) =>  
  Column(children: [  
    Text('目录'),  
    for (var volume in volumes)  
      ...volume.chapters,  
    if (page != pages.last)  
      Text('下一页'),  
    Text('索引'),  
  ]);
```

# IDE 对呈现 UI 代码的优化

在不改变代码的情况下，我们能否让开发者对 UI 代码的结构做到一目了然？

```
return SingleChildScrollView(  
  child— SafeArea(  
    top: false,  
    child— Container(  
      color: Theme.of(context).colorScheme.surface,  
      child— Column(  
        children: <Widget>[  
          Container(  
            constraints: const BoxConstraints.expand(height: 248),  
            child— Image.asset(  
              'food/fruits.png',  
              package: 'flutter_gallery_assets',  
              fit: BoxFit.fitWidth,  
            ), // Image.asset  
          ), // Container  
          const SizedBox(height: 17),  
        ],  
      ),  
    ),  
  ),  
);
```

# Editor UI Guides

```
return Row(  
  children: [  
    SizedBox(width: _leftColumnWidth),  
    Expanded(  
      child: Padding(  
        padding: const EdgeInsets.only(right: 16.0),  
        child: Column(  
          children: [  
            Row(  
              crossAxisAlignment: CrossAxisAlignment.center,  
              children: [  
                const Expanded(  
                  child: Text('TOTAL'),  
                ),  
                Text(  

```

# 其他的增强 UI 代码可读性的尝试

淡化 “child:” 和  
“children:”

```
return Row(  
  children: [  
    SizedBox(width: _leftColumnWidth),  
    Expanded(  
      child: Padding(  
        padding: const EdgeInsets.only(right: 16.0),
```

用字体中的 ligatures  
缩写 “child:” 和  
“children:”

```
return Row(  
  ↪ [  
    SizedBox(width: _leftColumnWidth),  
    Expanded(  
      ↪ Padding(  
        padding: const EdgeInsets.only(right: 16.0),
```



# 小结：为开发者体验作持续的优化

## Dart 1.x

## Dart 2.0: Optional 'new'

## Dart 2.3: UI as Code

## Editor UI Guides

```
// Dart 1.x
@override
Widget build(BuildContext context) {
  var items = [new Text('目录')];
  // 把章节按分卷放到目录里
  for (var volume in volumes) {
    items.addAll(volume.chapters);
  }

  if (page != pages.last) items.add(

  items.add(new Text('索引'));

  return new Scaffold(
    appBar: new AppBar(
      title: new Text(widget.title),
    ), // AppBar
    body: new Center(
      child: new Column(
        mainAxisAlignment: MainAxisAlignmentA
        children: items,
      ), // Column
    ), // Center
  ); // Scaffold
}

//Dart 2.0
@override
Widget build(BuildContext context)
Widget build(BuildContext context) {
  var items = [Text('目录')];
  // 把章节按分卷放到目录里
  for (var volume in volumes) {
    items.addAll(volume.chapters);
  }

  if (page != pages.last) items.addo

  items.add(Text('索引'));

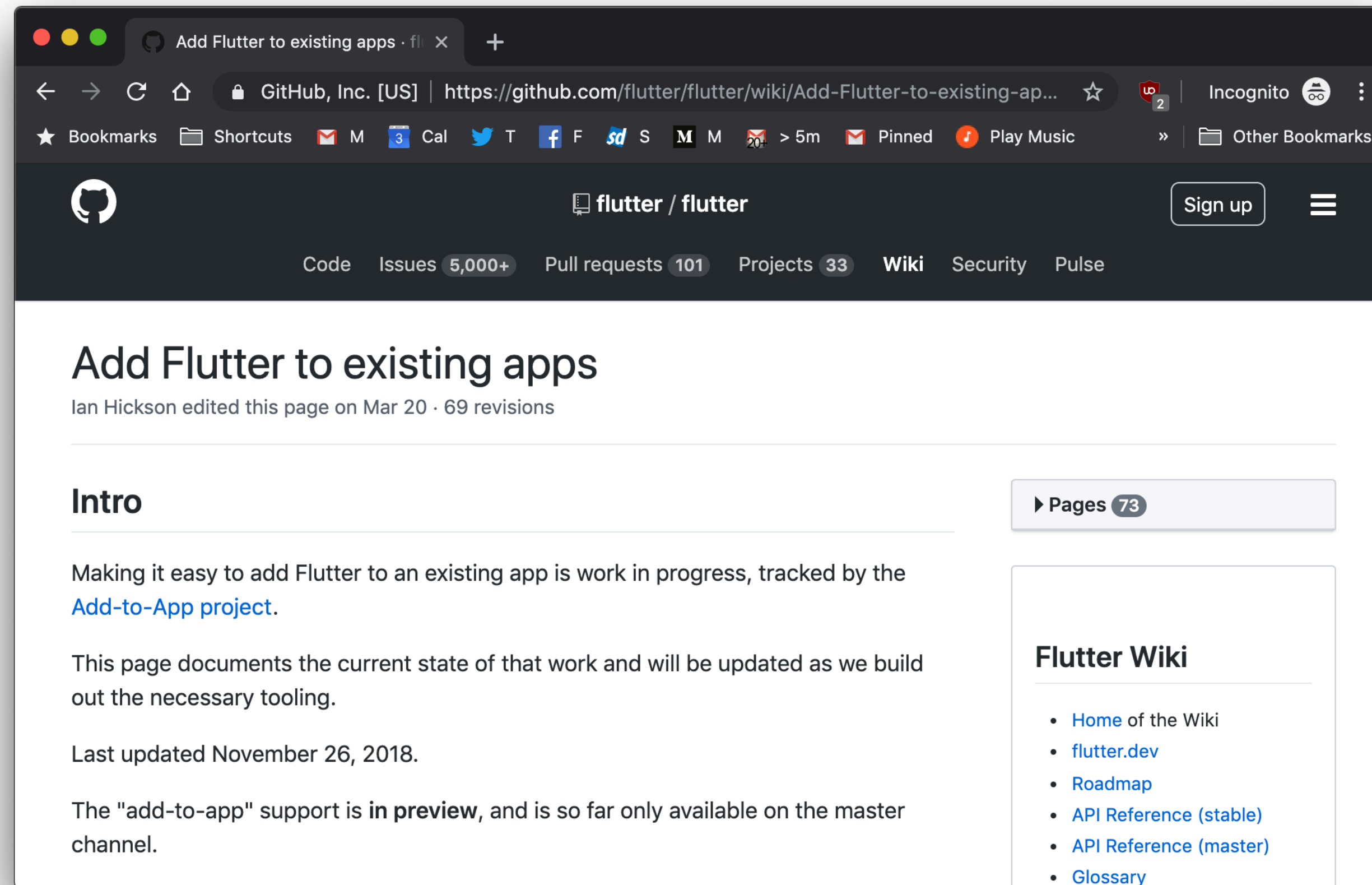
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ), // AppBar
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisis
        children: items,
      ), // Column
    ), // Center
  ); // Scaffold
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ), // AppBar
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignr
        children: [
          Text('目录'),
          for (var volume in volumes)
            ...volume.chapters,
          if (page != pages.last)
            Text('下一页'),
          Text('索引'),
        ],
      ), // Column
    ), // Center
  ); // Scaffold
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment
        children: [
          Text('目录'),
          for (var volume in volumes)
            ...volume.chapters,
          if (page != pages.last)
            Text('下一页'),
          Text('索引'),
        ],
      ),
    ),
  );
}
```

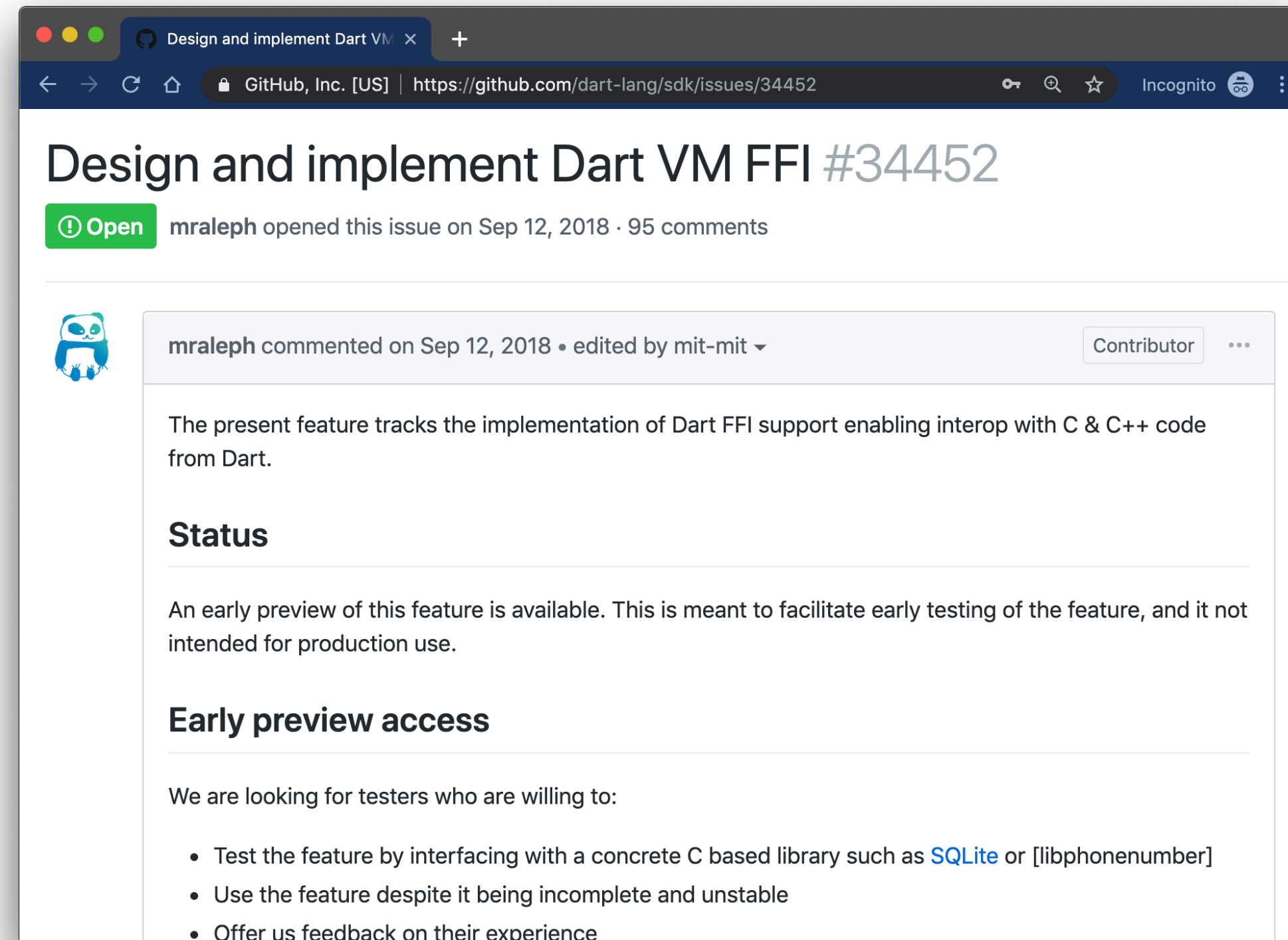
# 5. Flutter 和 Dart 近期展望

- 改进 Flutter 和原生混编（add-to-app）的机制和开发体验



# 5. Flutter 和 Dart 近期展望

- 改进 Flutter 和原生混编（add-to-app）的机制和开发体验
- Dart 语言增加 non-nullable (by default) types 和 extension methods, Dart runtime 会增加 FFI (C/C++ interop)





## 5. Flutter 和 Dart 近期展望

- 改进 Flutter 和原生混编（add-to-app）的机制和开发体验
- Dart 语言增加 non-nullable (by default) types 和 extension methods, Dart runtime 会增加 FFI (C/C++ interop)
- 更容易理解的报错信息
- Flutter for Web 性能和可用性优化
- 完善桌面 UI 的基本交互方式



# Flutter 2019 路线图

flutter / flutter

Watch 2,314 Star 67,287 Fork 7,668

Code Issues 5,000+ Pull requests 103 Projects 33 Wiki Security Insights

## Roadmap

Ray Rischpater, KF6GPE edited this page 11 days ago · 6 revisions

### 2019

Pages 73

With the [release of Flutter 1.0](#), we've made a good start but we've still got lots of work to do!

In the interest of transparency, we want to share high-level details of our roadmap, so that others can see our priorities and make plans based off the work we are doing.

We've established some broad themes that we want to focus on over the coming year:

- Fundamentals
- Ease of adoption
- Ecosystem
- Support for platforms beyond mobile
- Tooling

Our plans will of course evolve over time based on customer feedback and new market opportunities. The list here shouldn't be viewed either as exhaustive, nor a promise that we will complete all this work. If you have feedback about what you think we should be working on, we encourage you to get in touch (e.g. by [filing an issue](#) or [emailing the flutter-dev mailing list](#)). And

#### Flutter Wiki

- [Home of the Wiki](#)
- [flutter.dev](#)
- [Roadmap](#)
- [API Reference \(stable\)](#)
- [API Reference \(master\)](#)
- [Glossary](#)
- [Contributor Guide](#)
- [Code of Conduct](#)

#### Process

- [Our Values](#)
- [Tree hygiene](#)
- [Issue hygiene and Triage](#)

# Dart 语言设计文档和研发计划

dart-lang / language

Watch 79 Star 376 Fork 29

Code Issues 205 Pull requests 12 Projects 1 Wiki Security Insights

Design of the Dart language

dart-language language-changes specification

269 commits 17 branches 0 releases 17 contributors View license

Branch: master New pull request Create new file Upload files Find File Clone or download

ernstg Revert "Removed --enable-experiment flag for non-function type aliases" Latest commit b3fe5c5 10 days ago

accepted	Revert "Removed --enable-experiment flag for non-function type aliases"	10 days ago
doc	Clarify file names (#56)	8 months ago
inactive	Add the old inactive proposal to reserve "async" and "await". (#174)	5 months ago
minutes	Add minutes for 2018.08.13 language meeting	10 months ago
resources	Write up summary of optional semicolon prototypes.	4 months ago
specification	Tweaks	12 days ago
templates	Clarify implementation step for validating codegen (#325)	2 months ago
working	Move nnbd roadmap to accepted	18 days ago

工程实施阶段



设计阶段





# TGO 鲲鹏会

## 汇聚全球科技领导者的高端社群

🏠 全球12大城市

👤 850+ 高端科技领导者

使命

Mission

为社会输送更多优秀的  
科技领导者

愿景

Vision

构建全球领先的有技术背景  
优秀人才的学习成长平台



扫描二维码，了解更多内容



# THANKS



官方网站  
[flutter.dev](https://flutter.dev)

中文社区站  
[flutter.cn](https://flutter.cn)



# Flutter