Dart is All The Things

小德

DJI 高级工程师

Geekbang 极客邦科技 | InfoQ 极客传媒

InfoQ官网 全新改版上线

促进软件开发领域知识与创新的传播

关注InfoQ网站
第一时间浏览原创IT新闻资讯

免费下载迷你书
阅读一线开发者的技术干货

自我介绍

- DJI
- 《Flutter 完全手册》作者

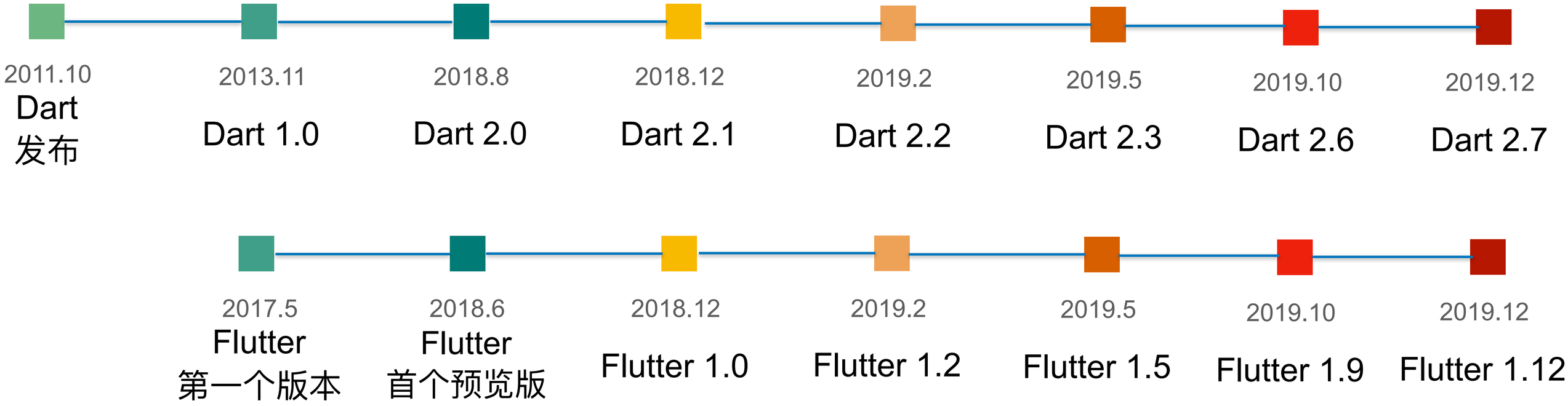
目录

Dart & Flutter

Dart 初印象

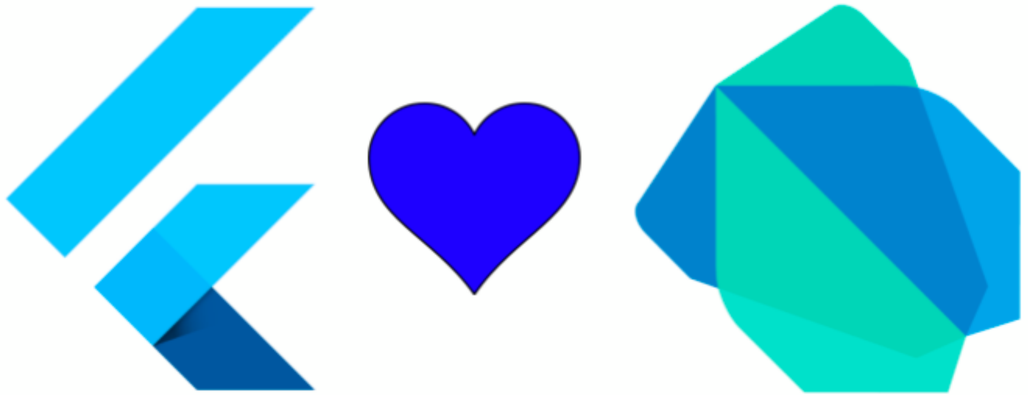
- Dart 完全由 Flutter 带起来

Dart & Flutter 的版本

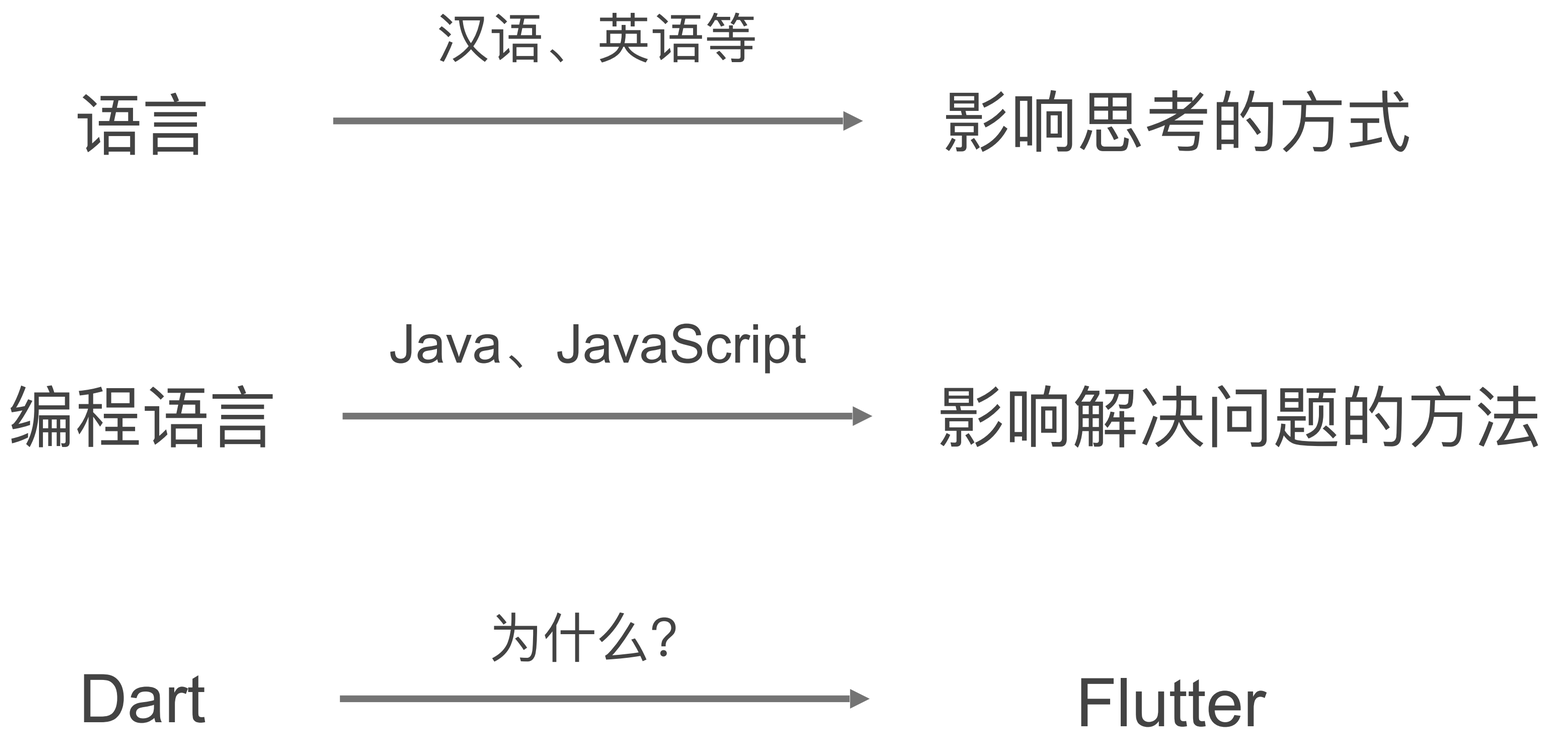


Flutter & Dart

- Dart 和 Flutter 的发版节奏基本同步
- Dart 已经和 Flutter 绑定



Flutter 为什么选择 Dart ?



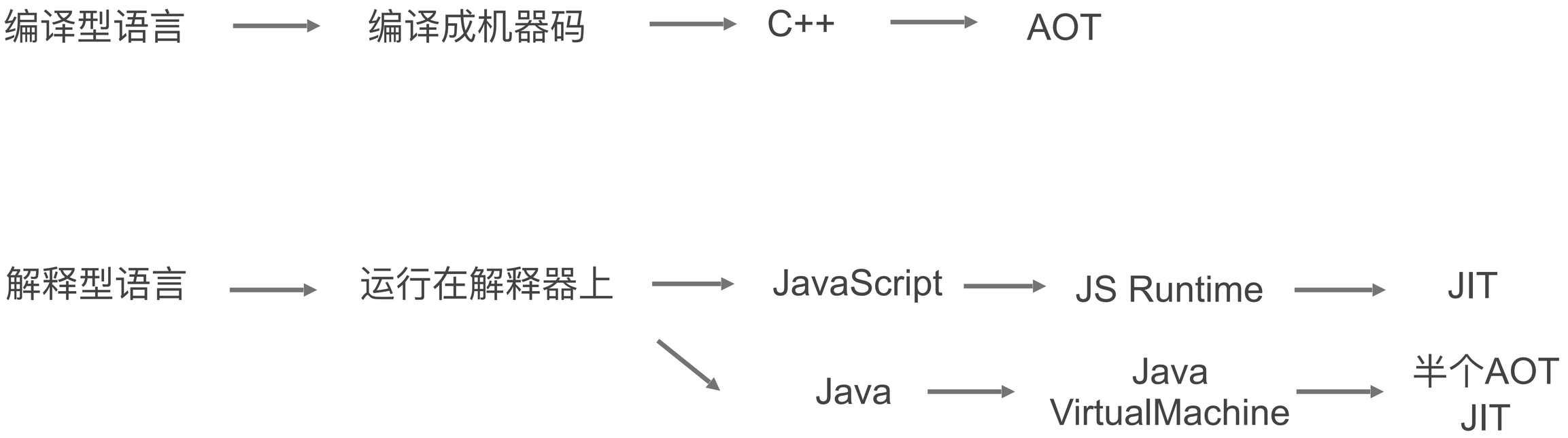
Flutter 为什么选择 Dart ?

- Dart 同时支持 AOT 和 JIT

AOT → Ahead-Of-Time(预先编译)

JIT → Just-In-Time(实时编译)

AOT 与 JIT 的发展


编译型语言
编译成机器码
C++
AOT
解释型语言
运行在解释器上
JavaScript
JS Runtime
JIT
Java
Java VirtualMachine
半个AOT JIT

AOT & JIT

			开发速度	跨平台	性能
C++	→	AOT	慢	高	高
Java	→	半个AOT & JIT	中	中	中
JavaScript	→	JIT	快	高	低

AOT & JIT

- AOT 性能高、跨平台好，但是开发速度慢
- JIT 开发速度快、跨平台好，但是性能差
- 现代语言多采用混合编译方式
- 以达到 性能、跨平台、开发速度 的最佳结合
- Dart 就是这样的语言，所以 Flutter 选择了 Dart

AOT & JIT

AOT → Dart 编译成机器码 → 运行在 ARM & X64 → 性能
跨平台

JIT → Dart 编译成 Kernel binary
运行在 Dart VM 上 → Hot Reload
代码即时生效
不需要再编译 → 开发速度

- Dart 是唯一一个支持严格的 AOT 和 JIT 的编程语言

Flutter & Dart

Flutter 对 Dart 的影响

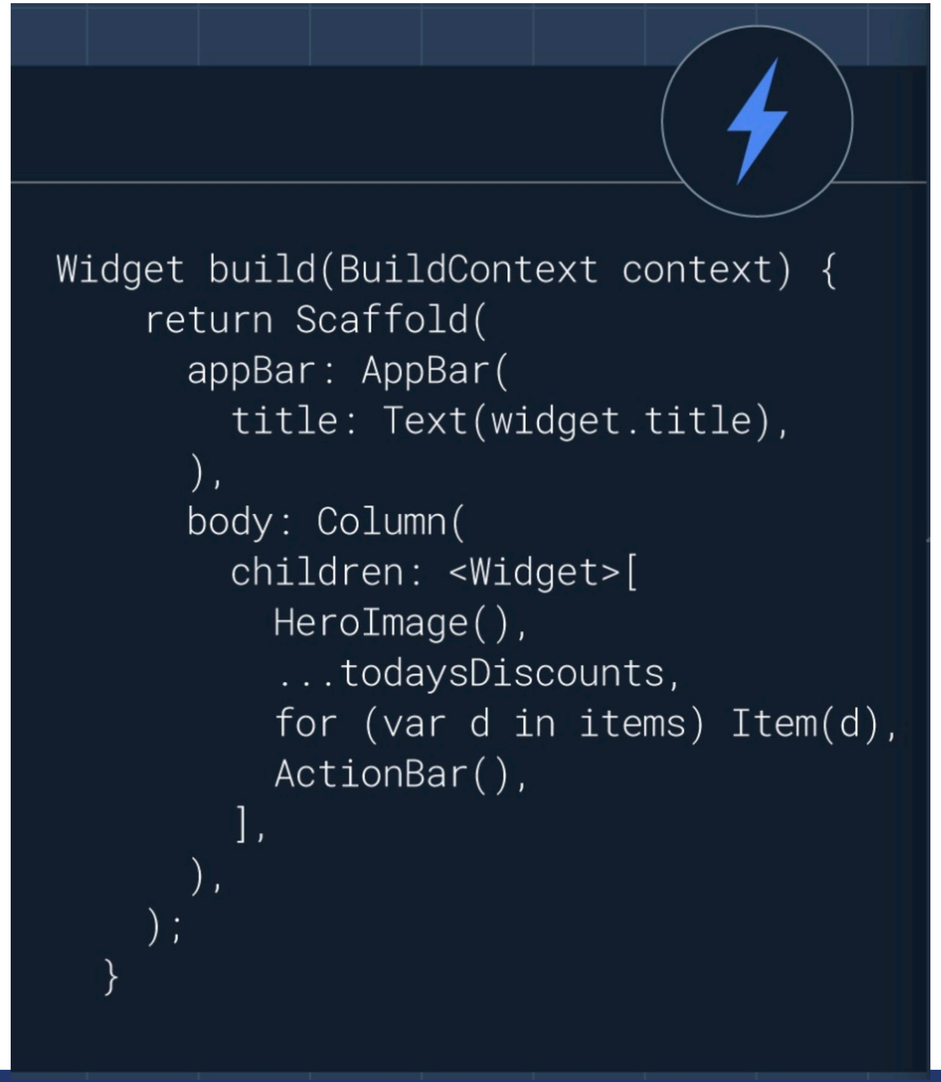
- Flutter 也在 促进 Dart 不断发展

Flutter 对 Dart 的影响

- 使 Dart 成为对 UI 最友好的语言
- Dart is a client-optimized language
- Optimized for UI

写 UI 的流程

声明式 UI



```
Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Column(
        children: <Widget>[
          HeroImage(),
          ...todaysDiscounts,
          for (var d in items) Item(d),
          ActionBar(),
        ],
      ),
    );
  }
```

- 用代码写 UI
- 代码的嵌套表示 UI 的层级
- 命名参数和数组

spread operator

```
List<Widget> northAmericanCountries = [
    Text('USA'),
    Text('Canada')
];

List<Widget> asianCountries = [
    Text('India'),
    Text('China')
];

List<Widget> europeanCountries = [
    Text('France'),
    Text('Germany'),
    Text('Italy')
];
```

spread operator

```
Row(
  children: northAmericanCountries
              ..addAll(asianCountries)
              ..addAll(europeanCountries),
)
```

```
Row(
  children: asianCountries
              ..addAll(northAmericanCountries)
              ..addAll(europeanCountries),
)
```

```
Row(
  children: [
              ...northAmericanCountries,
              ...asianCountries,
              ...europeanCountries
            ],
)
```

collection if

```
var countriesToShow = northAmericanCountries;
countriesToShow.addAll(europeanCountries);

if (isAsian)
   countriesToShow.addAll(asianCountries);
```

```
var countriesToShow = [
     ...northAmericanCountries,
     ...europeanCountries,
     if (isAsian)
          ...asianCountries
];
```

collection for

```
Row(
    children: [
        europeanCountries
          .map((country) => Text("New $country"))
          .toList()
    ]
)
```

```
Row(
    children: [
        for (var country in europeanCountries) Text("New ${country.data}")
    ]
)
```

spread operator & collection if & collection for

觉得没有用?

- UI 上可以实现简单逻辑
- 实现了 UI 和 Dart 语法完美融合

获取数据

Isolate

Dart 的 Isolate 是单线程  但是单线程如何实现高性能?

Isolate 的组成

Isolate

→ 私有内存 → 没有锁的问题 → GC比较快

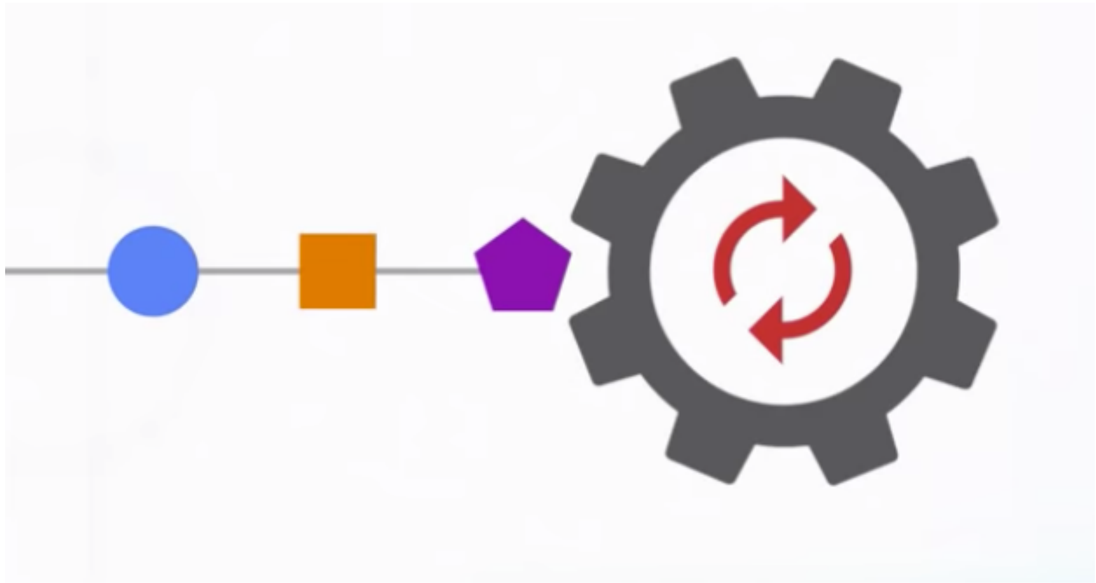
+

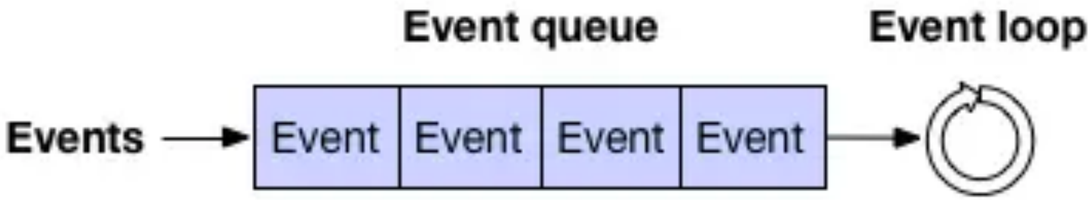
→ 一个线程

+

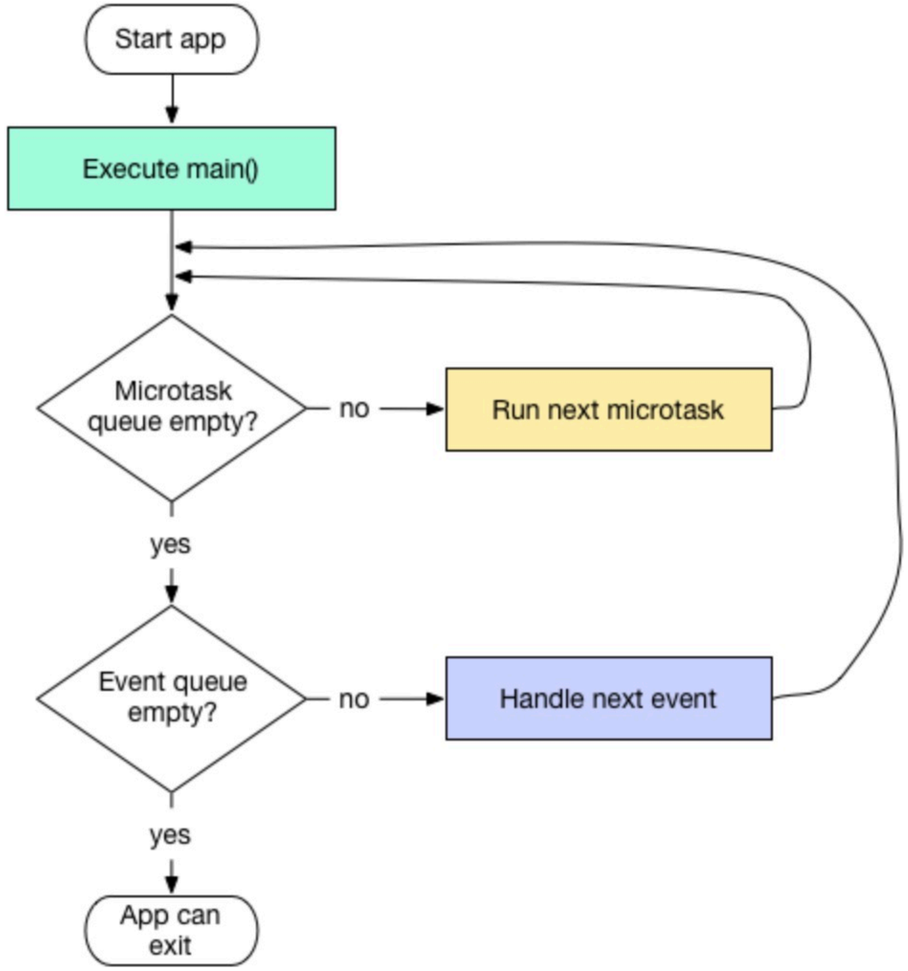
→ Event Loop

Event Loop





Event-loop



- Microtask queue

内部事件：

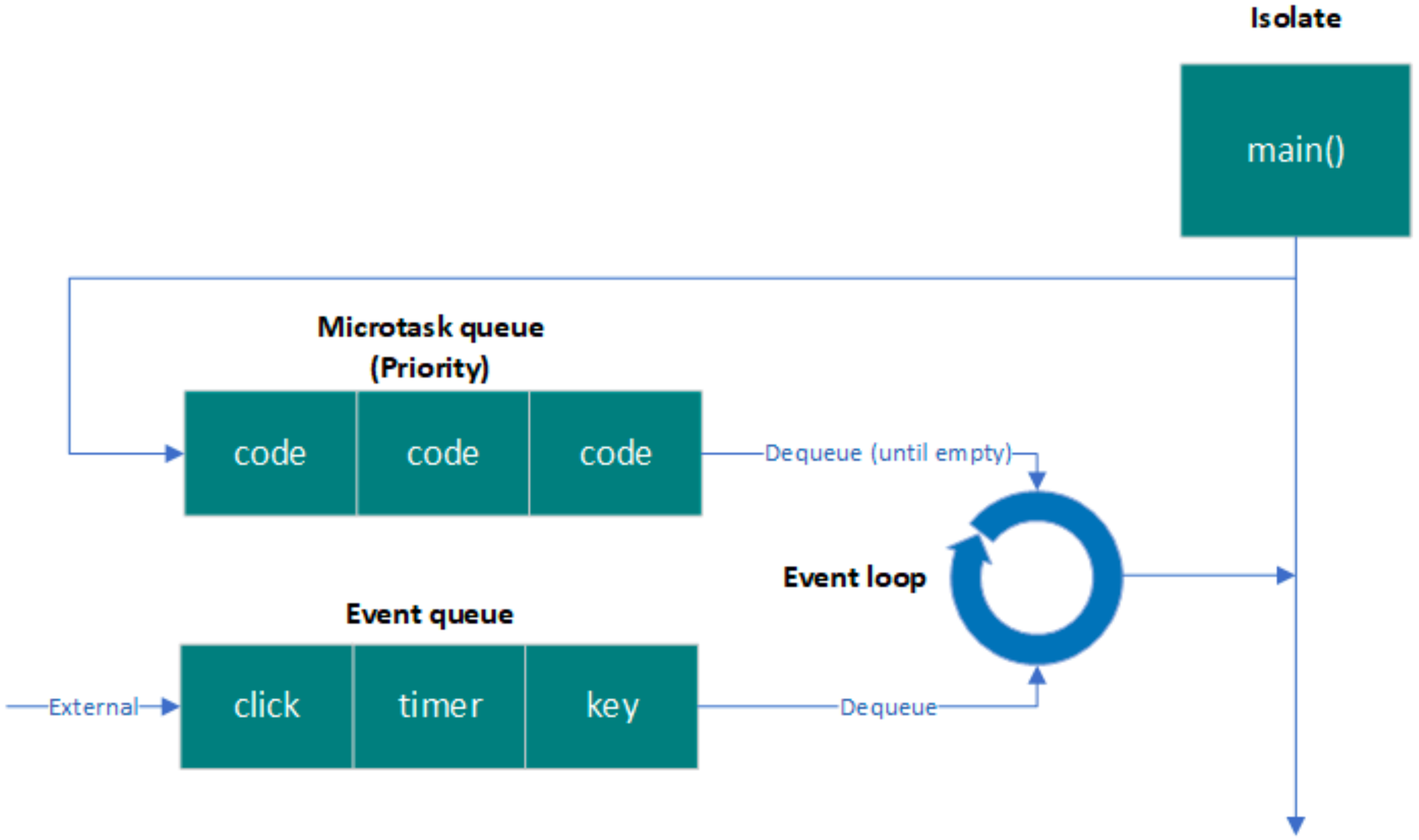
- 系统内部
- Future.microtask
- …

- Event queue

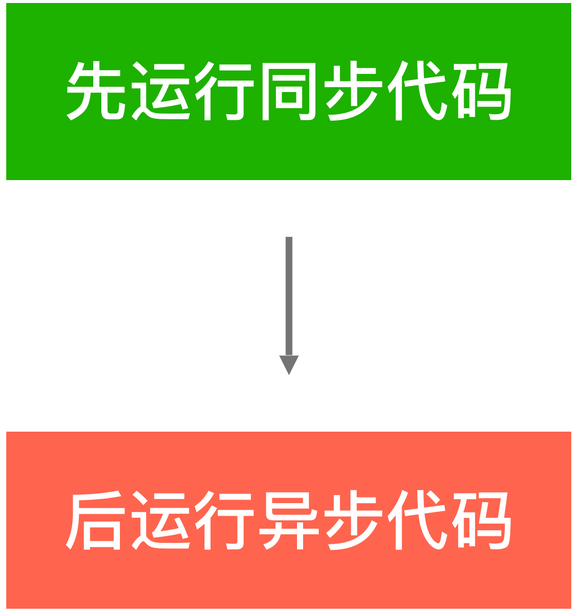
外部事件：

- I/O
- gesture
- drawing
- timers
- futures
- async
- …

Event-loop


Isolate
main()
Microtask queue
(Priority)
code
code
code
Dequeue (until empty)
Event loop
Event queue
External
click
timer
key
Dequeue

Event-loop



Future

Future ⟶ 代表需要延迟计算的对象

```
import 'dart:async';

Future myFunction() => new Future.value('Hello');

void main() {
  myFunction().then((value) => debugPrint(value));
  myFunction().then((value) => debugPrint(value));
  someOtherFunction();

}
```

async/await

```
void mainTest() async {
  debugPrint(await myFunction()); // Waits for completion
  debugPrint(await myFunction()); // Waits for completion
  someOtherFunction(); // Runs this code
}
```

- 把异步代码写成同步代码的样子，更好理解

Isolate & async

```
Future<int> findPrimeCount(int num) async {
  int primeCount = 0;
  for (int i = 1; i <= num; i++) {
    int divideCount = 0;
    for (int j = 1; j <= i; j++) {
      if (i % j == 0) {
        divideCount++;
      }
    }
    if (divideCount <= 2) {
        primeCount++;
    }
  }
  return primeCount;
}
```

- CPU Bound
- IO Bound

Isolate & async

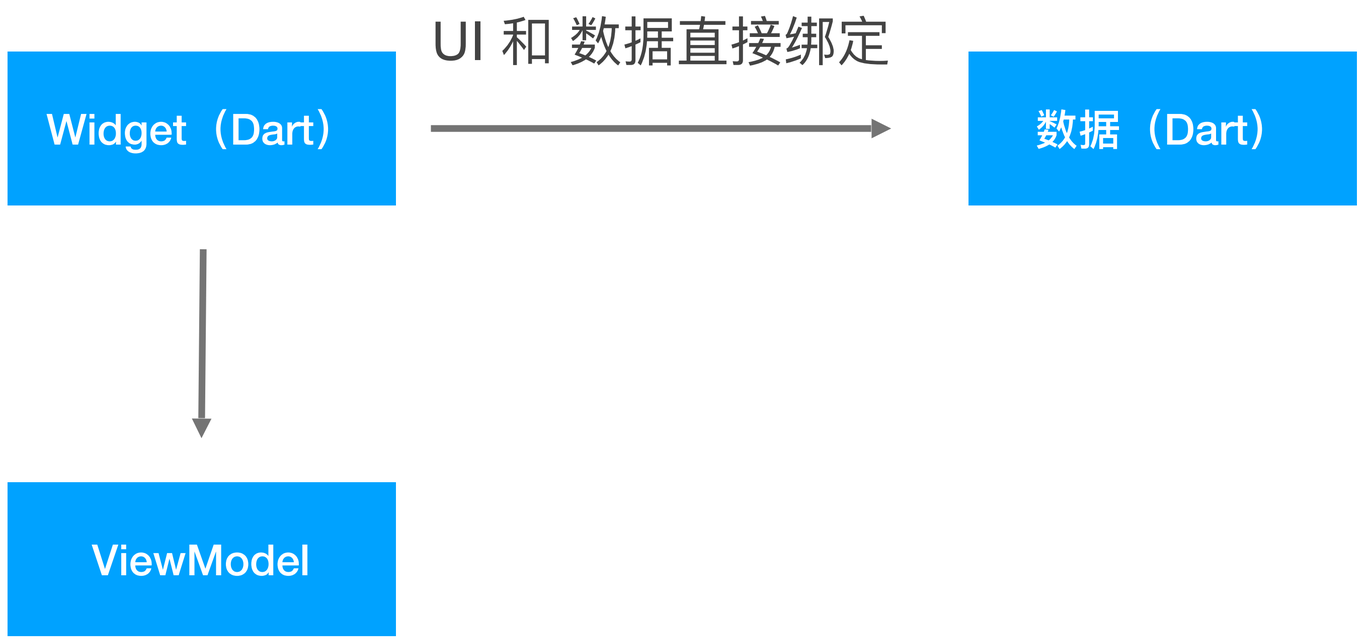
CPU 密集型 → Isolate

IO 密集型 → async

数据与UI绑定

Widget

```
Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Column(
        children: <Widget>[
          HeroImage(),
          ...todaysDiscounts,
          for (var d in items) Item(d),
          ActionBar(),
        ],
      ),
    );
  }
```



Dart 全平台

Dart 的另一面

WEB

PC

后台

Dart for backend

- http_server
- 支持 HTTP 的 dart:io 库
- 异步编程

Dart for backend

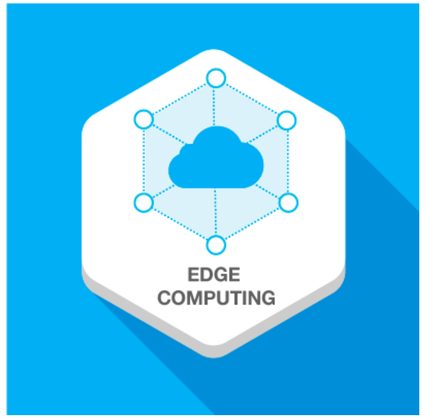
- 所有端的语言一致
- 但是，毕竟客户端和后台解决的问题是不一样的
- 思路也会差很大
- 很有潜力，但是实行起来会比较困难

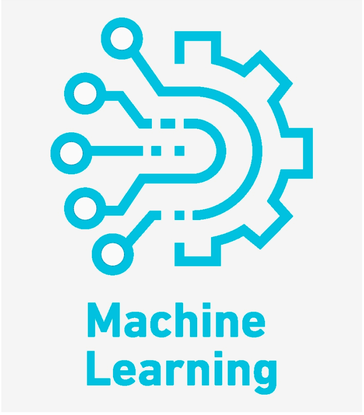
Dart 的另一个亮点

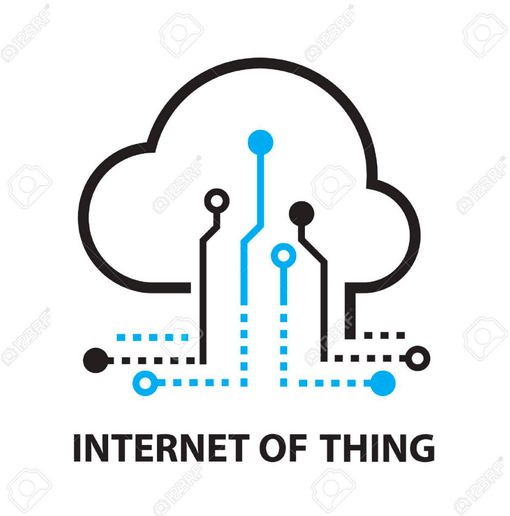
dart:ffi 与 C 交互

```
import 'dart:ffi' as ffi;

typedef hello_world_func = ffi.Void Function();

typedef HelloWorld = void Function();

final dylib = ffi.DynamicLibrary.open(‘hello_world.dylib');

final HelloWorld hello = dylib
    .lookup<ffi.NativeFunction<hello_world_func>>('hello_world')
    .asFunction();
```

为什么需要极致性能?







总结

- 声明式 UI
- spread operator & collection if & collection for
- 异步支持：Isolate & EventLoop & Future & async & await
- Dart for backend
- dart:ffi

展望

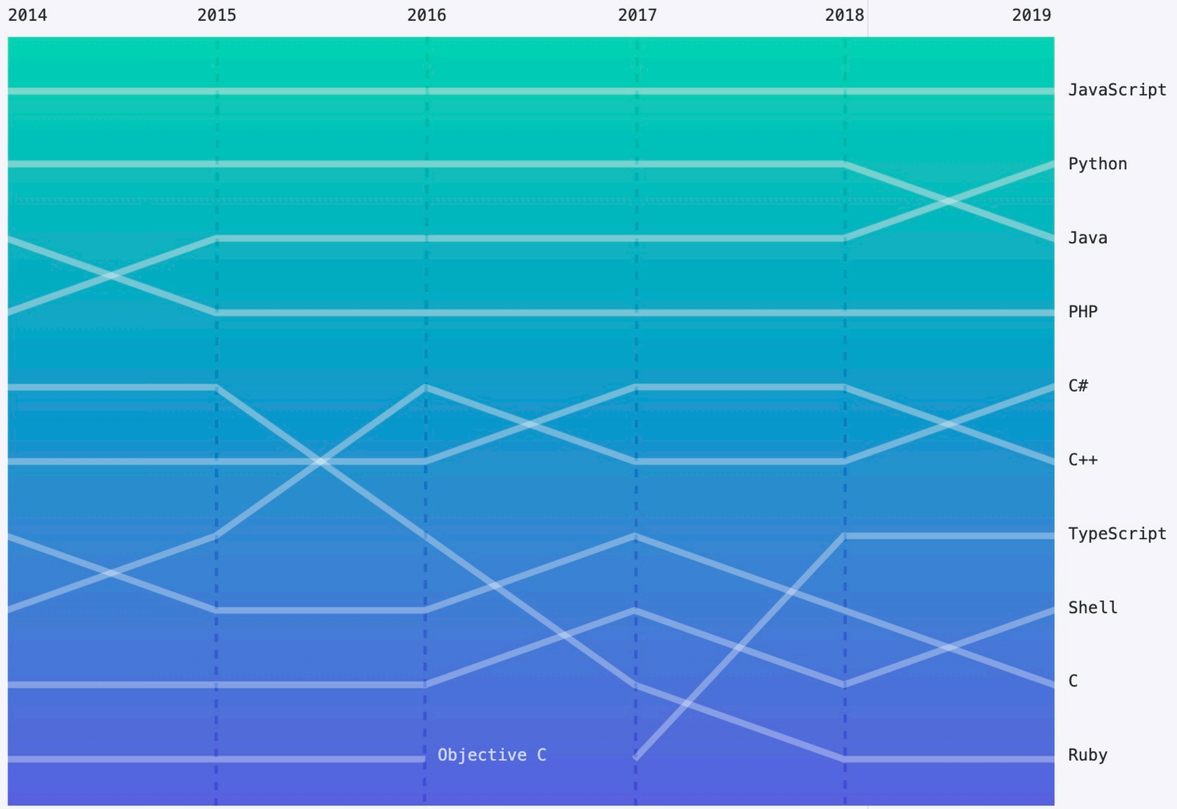
2019 Github 语言排行榜

Top languages

In the last year, developers collaborated in more than 370 primary languages on GitHub.

Top languages over time

This year, C# and Shell climbed the list. And for the first time, Python outranked Java as the second most popular language on GitHub by repository contributors.*



Dart 展望

我也无法预测未来会怎样

但是我建议大家都去学一门新语言,比如 Dart

Geekbang. 极客邦科技 | TGO 鲲鹏會

TGO鲲鹏会

汇聚全球科技领导者的高端社群

全球12大城市

900+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

极客时间

一起领「敲代码神器」

Git 快捷口令超大鼠标垫

常看常记，运指如飞，不卡壳

扫码免费领取

仅限200份，先到先得

THANKS

GMTC
全球大前端技术大会