特别提醒: 本幻灯片和活动举办时间为 2019 年，其中的内容无法保证最新和仍然可用。
如果有任何问题，请访问 flutter.cn 或其他方式与我们联系。

What makes Flutter fast
Flutter 为何快

Yuqian Li
李宇骞

Philosophy

Lazy programming

Dart AOT: 0.15s

```
import 'dart:typed_data';

const int T = 10000;
const int N = 1000;
```

C++: 0.28s

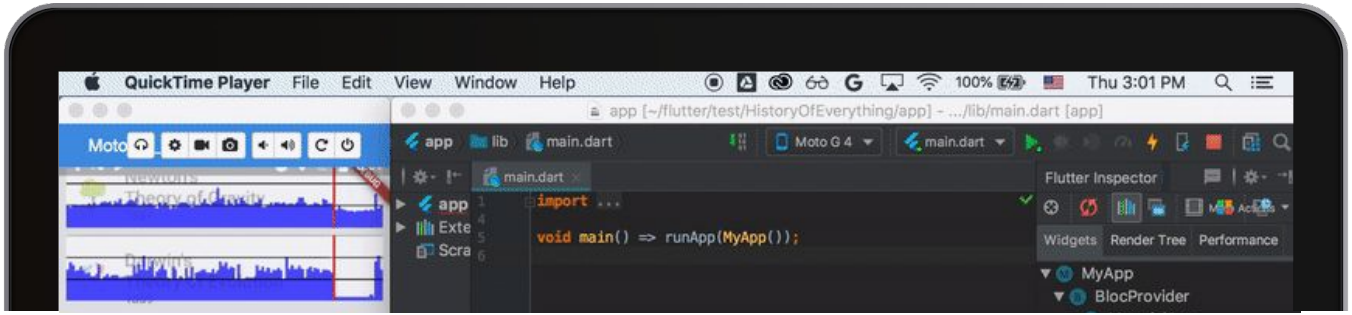
```
const int T = 10000;
const int N = 1000;

int main() {
 for(int t = 0; t < T; t++) {
```

Java: 0.60s

```
import java.util.ArrayList;

final class jstr {
 public static void main(String[] args) {
  final int T = 10000;
```



Build the best way to develop ~~for mobile~~ apps

Alibaba.com

Google Ads

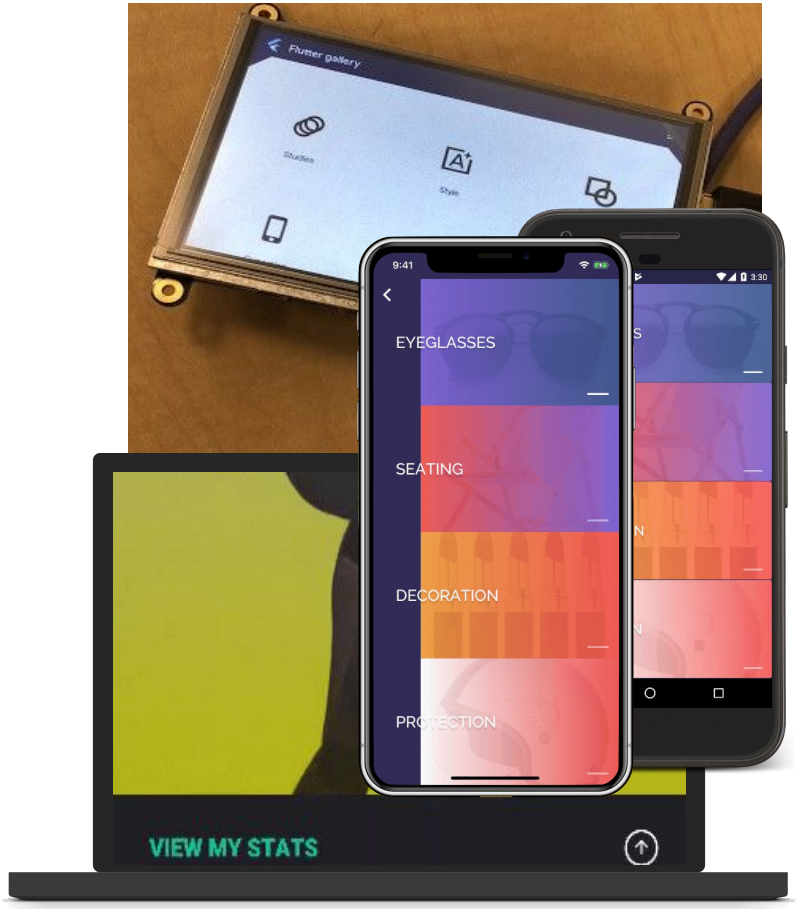
Tencent 腾讯

JD.COM 京东

PHILIPS

hue personal wireless lighting

.....



Beautiful

Productive

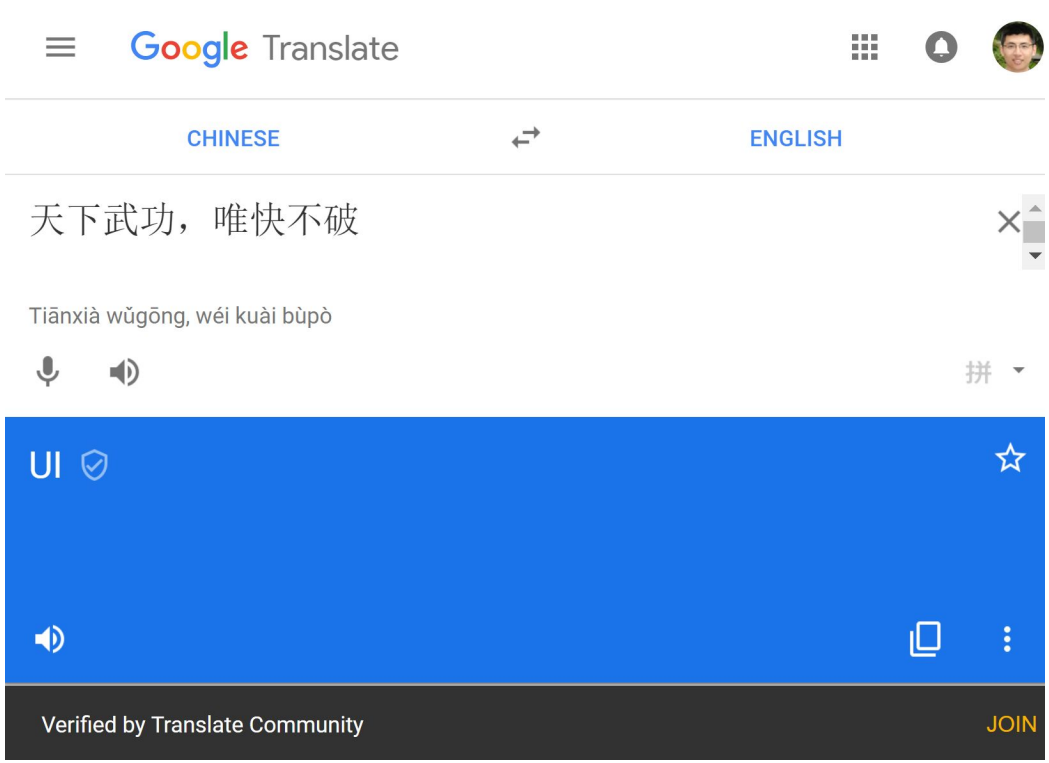
Fast Development

Flutter

Fast

Native Performance

Open



Screenshot taken from translate.google.com on 1/1/2019

Less work 干活少

Better incentive 激励好

Less work 干活少

Less compute work
Less engineer work
Less developer work

Less compute work: fewer malloc from Dart

Dart AOT: 0.15s

```
import 'dart:typed_data';

const int T = 10000;
const int N = 1000;

main() {
 for(int t = 0; t < T; t++) {
   var objs = new List<Uint8List>(N);
   for(int i = 0; i < N; i++) {
     objs[i] = new Uint8List(10);
   }
 }
}
```

C++: 0.28s

```
const int T = 10000;
const int N = 1000;

int main() {
 for(int t = 0; t < T; t++) {
   char* objs[N];
   for(int i = 0; i < N; i++) {
     objs[i] = new char[10];
   }
   for(int i = 0; i < N; i++) {
     delete [] objs[i];
   }
 }
}
```

Java: 0.60s

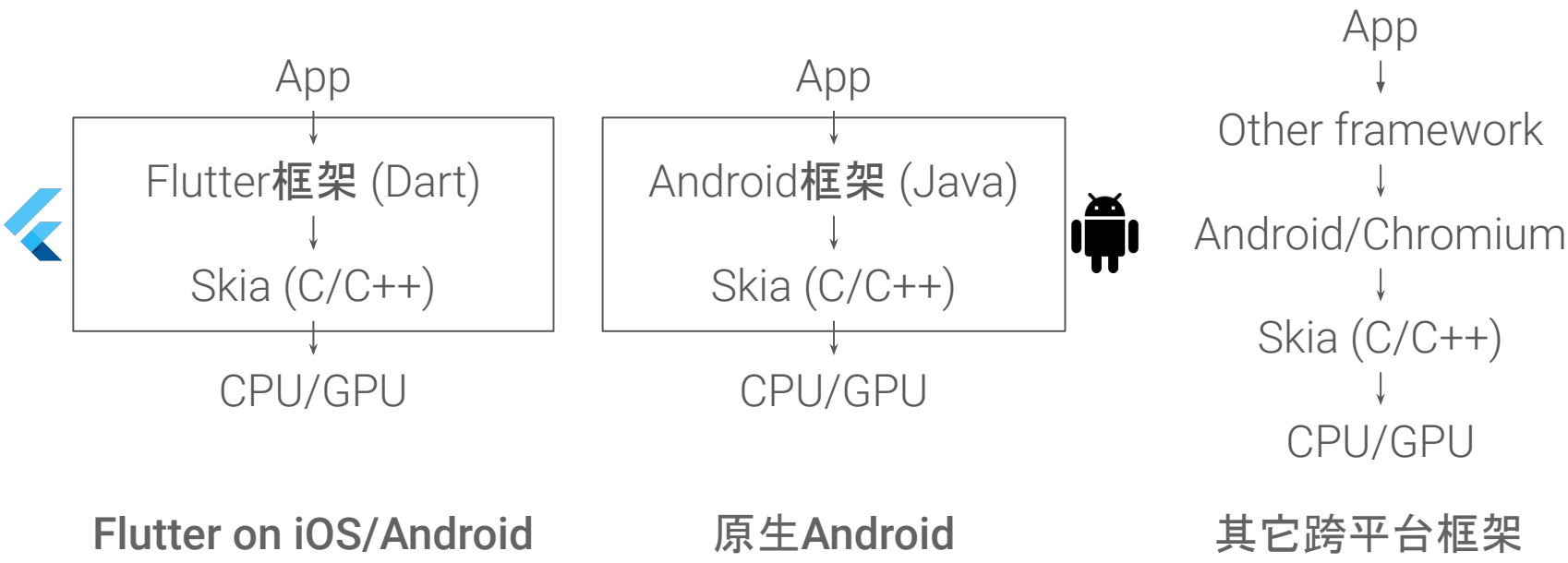
```
import java.util.ArrayList;

final class jstr {
 public static void main(String[] args) {
   final int T = 10000;
   final int N = 1000;

   for(int t = 0; t < T; t++) {
     ArrayList objs[] = new ArrayList[N];
     for(int i = 0; i < N; i++) {
       objs[i] = new ArrayList<Byte>(10);
     }
   }
 }
}
```

Experiments conducted on Linux x64 4.18.10, Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz,
Dart 2.2.0-edge.0797d6ef695739aafd3759096f4143c7440f0213, gcc version 7.3.0 (Debian 7.3.0-5) with -O3, openjdk version 1.8.0_161

Less compute work: skip Android/Chromium



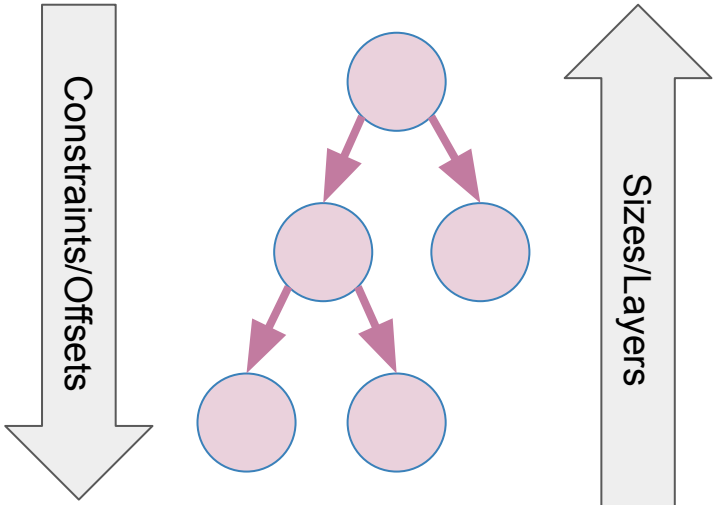
Flutter on iOS/Android

原生Android

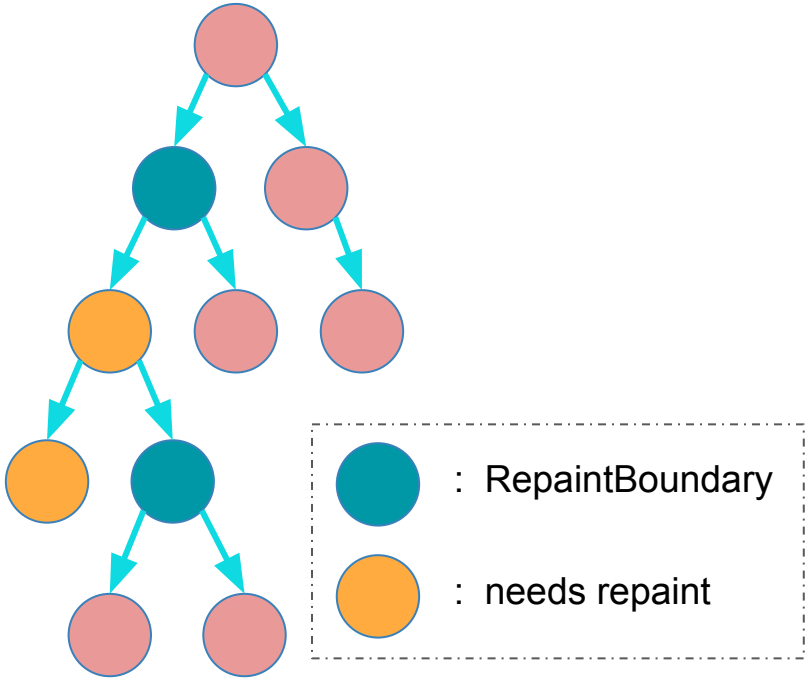
其它跨平台框架

From GDD China 2018

Less compute work: fewer tree walks



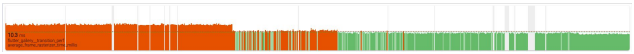
one-pass, linear time layout and painting
instead of $O(n^2)$



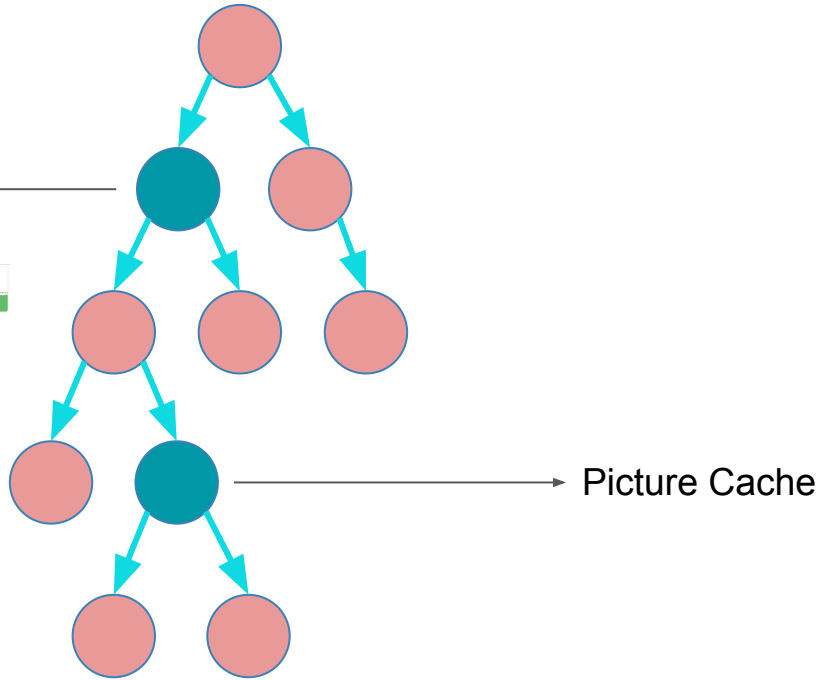
Tree diagrams copied from Xiao's GDD China 2018 slides.

Less compute work: raster cache

(newly added) Layer Cache

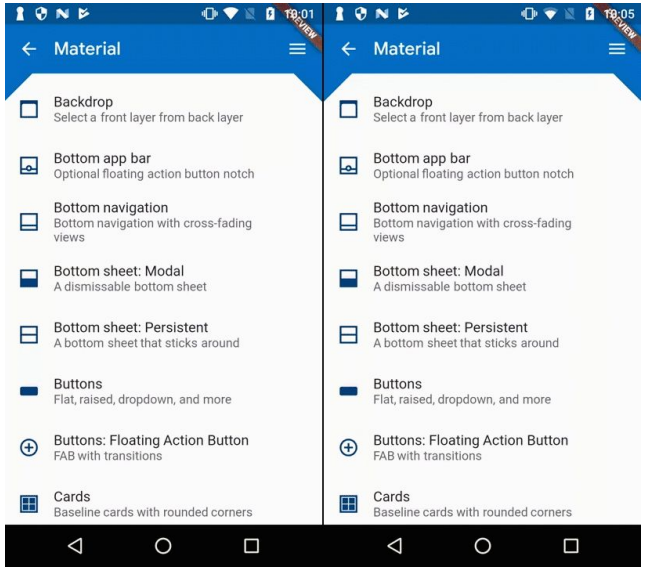


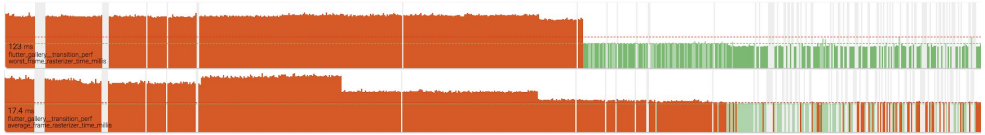
60% speedup in page transition animations.



Less compute work: fewer context switches

- Offscreen painting in Layer::Preroll instead of Layer::Paint
- Reduce the number of SkCanvas::saveLayer calls (& no clip by default)





2x speedup

screenshot from GDD China 2018 slides

Less engineer work: lazy programming

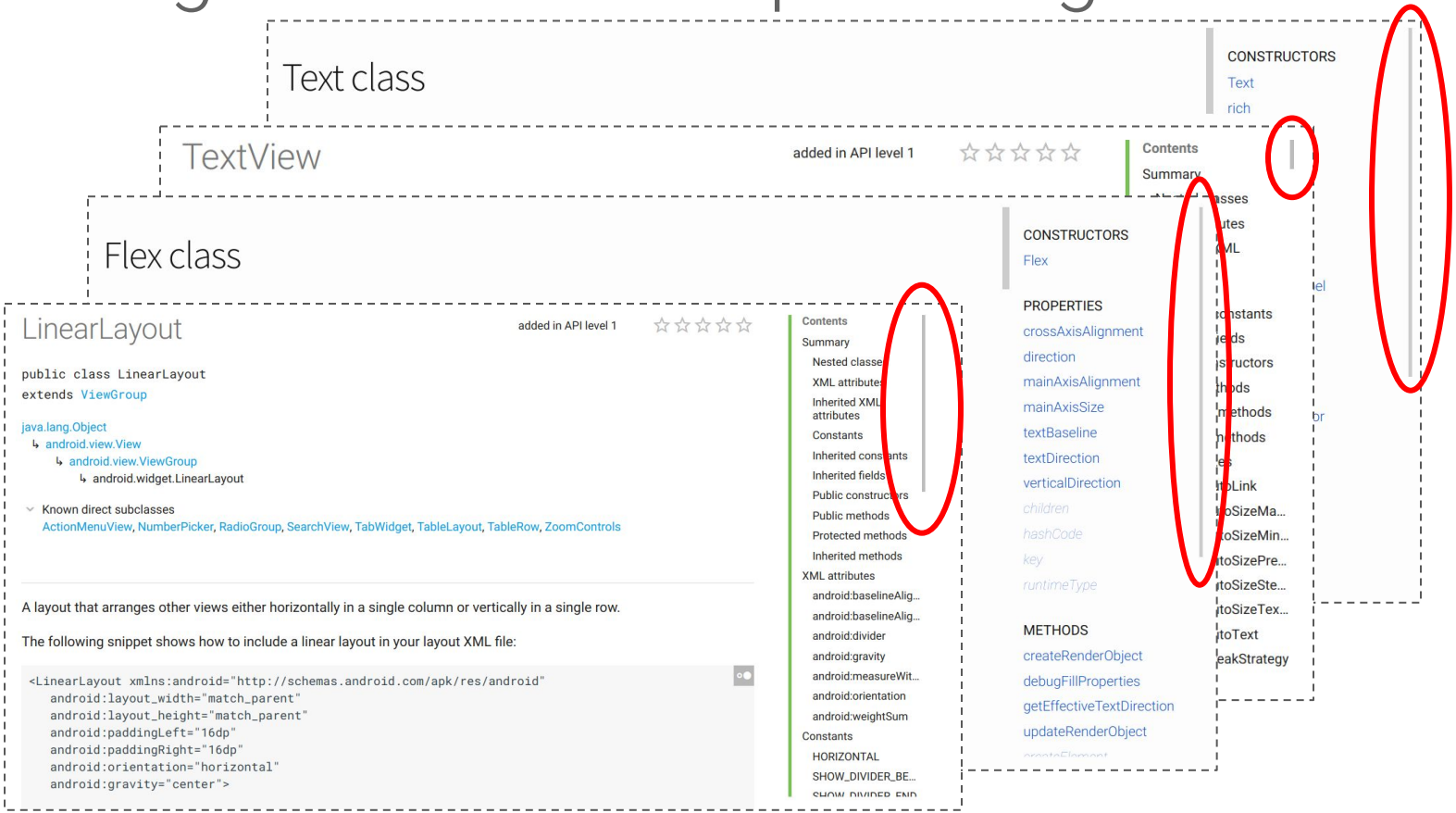
Philosophy

Lazy programming

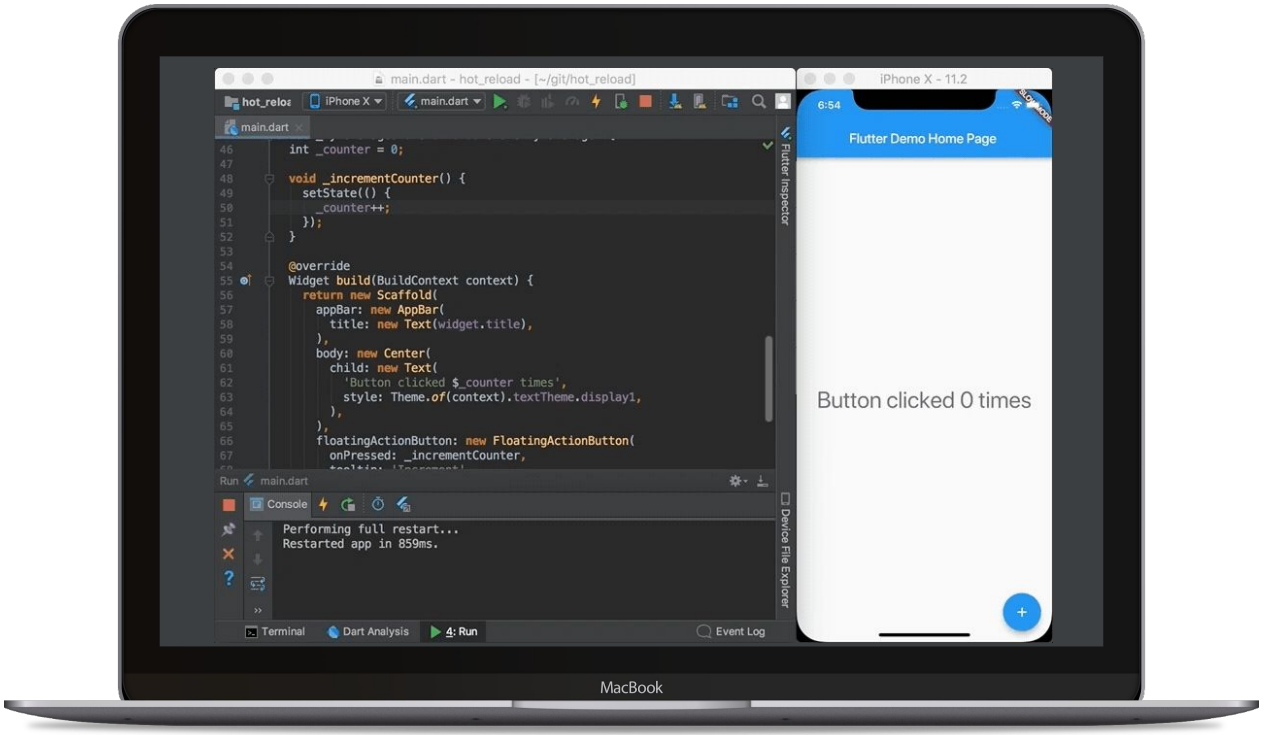
Write what you need and no more, but when you write it, do it right.

https://github.com/flutter/flutter/wiki/Style-guide-for-Flutter-repo#lazy-programming

Less engineer work: simpler widgets & APIs

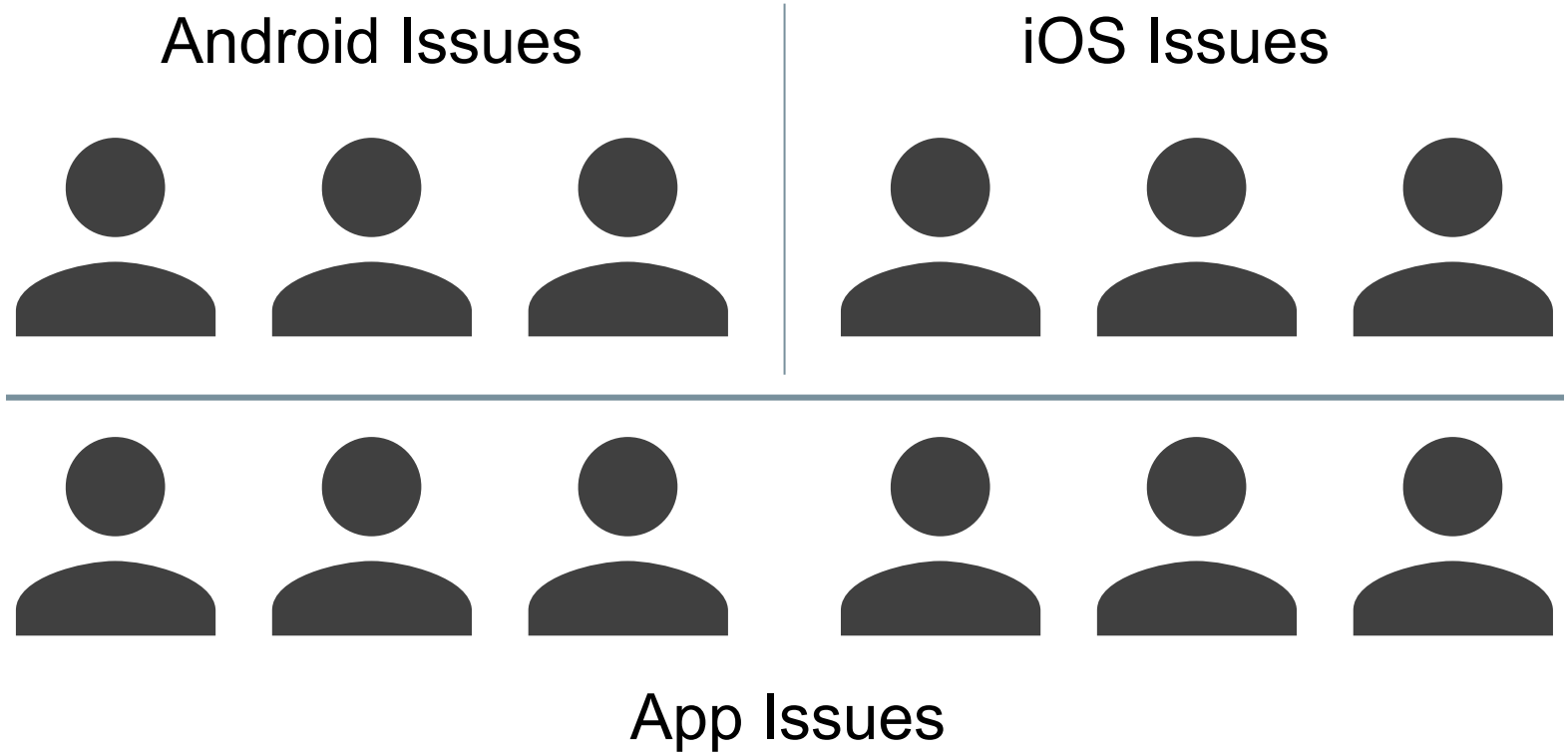


Less developer work: hot reload



Screenshot from Flutter 1.0 slides

Less developer work: single codebase

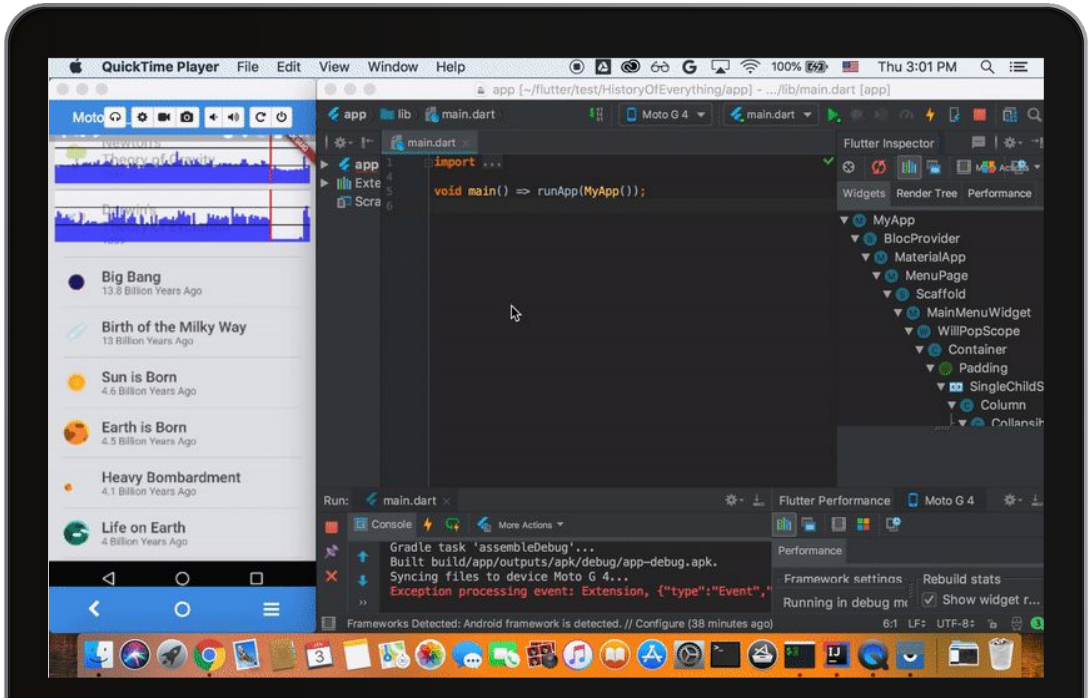


Less developer work: everything is Dart

- Fewer to learn

- Fewer cognitive context or tool switches

Less developer work: jump to code



Full video here: https://www.bilibili.com/video/av40362171/

Better incentive 激励好

Better API
Better metrics
Better feedback

Better API: cost awareness

No synchronous slow work

There should be no APIs that require synchronously completing an expensive operation (e.g. computing a full app layout outside of the layout phase). Expensive work should be asynchronous.

Getters feel faster than methods

Property getters should be efficient (e.g. just returning a cached value, or an O(1) table lookup). If an operation is inefficient, it should be a method instead. (Looking at the Web again: we would have `document.getForms()`, not `document.forms`, since it walks the entire tree).

```
image = await renderRepaintBoundary.toImage();
```

https://github.com/flutter/flutter/wiki/Style-guide-for-Flutter-repo#getters-feel-faster-than-methods
https://github.com/flutter/flutter/wiki/Style-guide-for-Flutter-repo#no-synchronous-slow-work

Better API: avoid some APIs

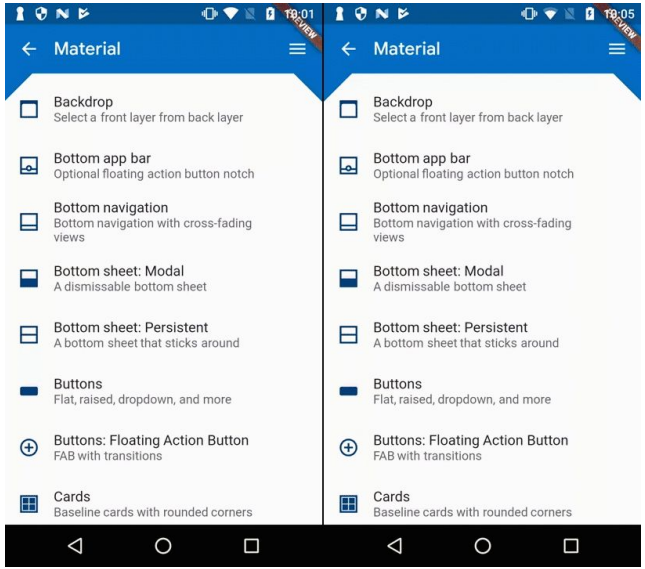
Avoid APIs that encourage bad practices

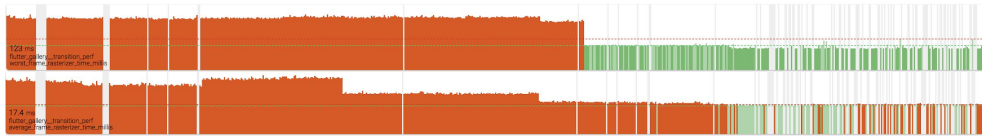
For example, don’t provide APIs that walk entire trees, or that encourage O(N^2) algorithms, or that encourage sequential long-lived operations where the operations could be run concurrently.

https://github.com/flutter/flutter/wiki/Style-guide-for-Flutter-repo#avoid-apis-that-encourage-bad-practices

Better API: fast by default

- clipBehavior: Clip.antiAliasWithSaveLayer -> Clip.none



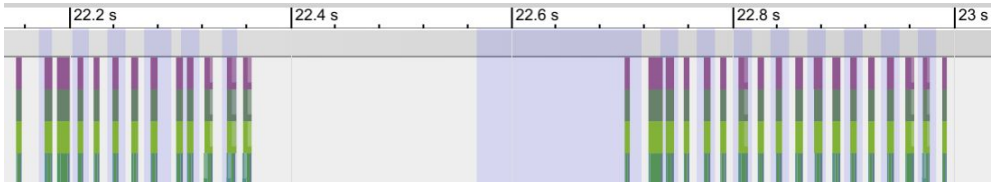


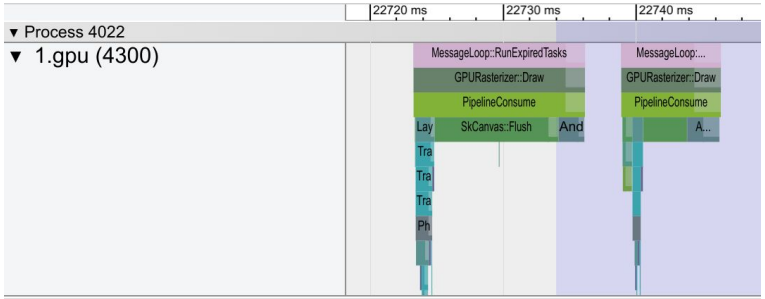
2x speedup

screenshot from GDD China 2018 slides

Better metrics: FPS -> average frame time

https://commons.wikimedia.org/wiki/File:Nuclear_Dawn_-_Silo_FPS_01.png





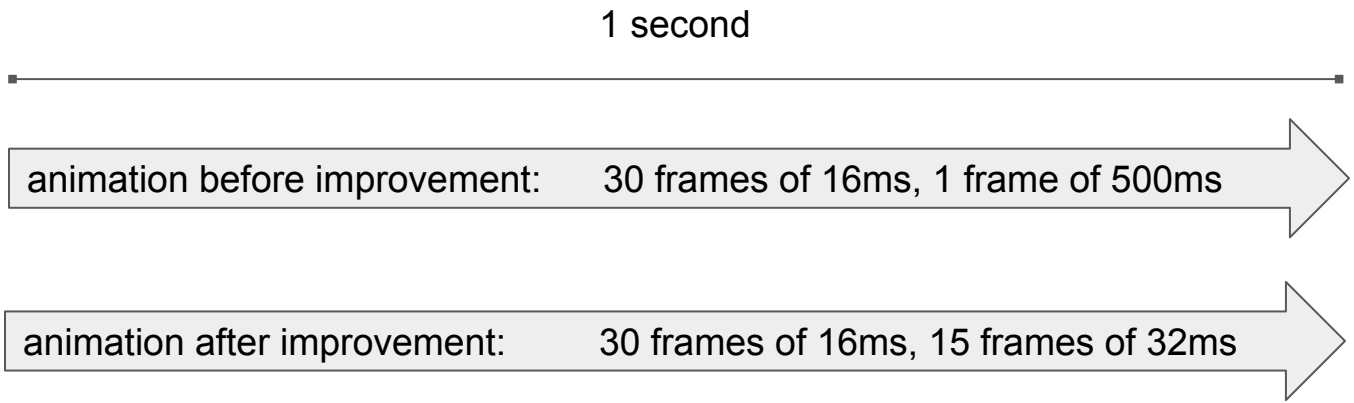
2 items selected: Slices (2)

Name ∇	Wall Duration ∇	CPU Duration ∇
MessageLoop::RunExpiredTasks	20.491 ms	17.012 ms

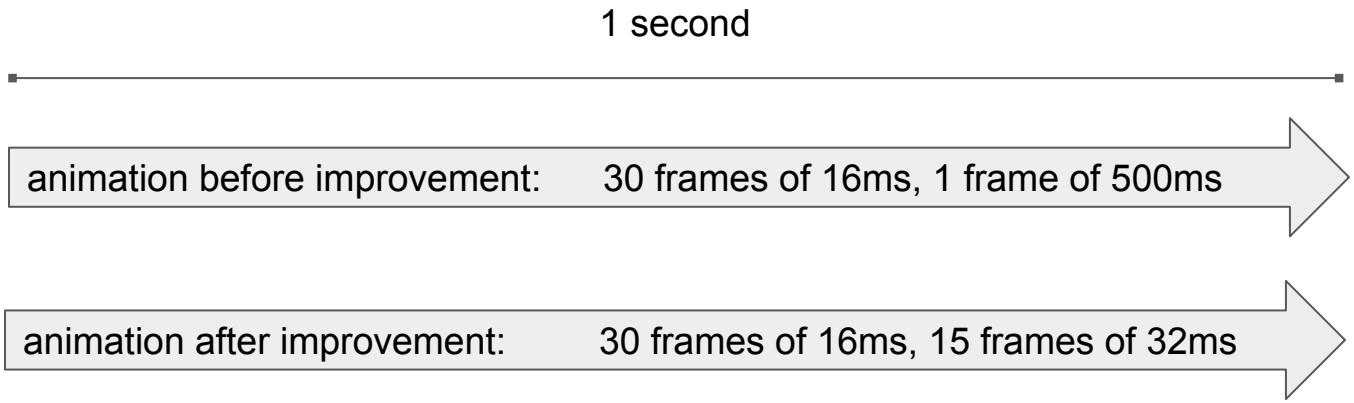
Selection start	**22,723.207 ms**
Selection extent	**23.191 ms**

Title	**MessageLoop::RunExpiredTasks**	
Start ▲	**Wall Duration (ms) ∇**	**CPU Duratic**
22,723.207 ms	12.950 ms	11.482 ms
22,738.857 ms	7.541 ms	5.530 ms

Better metrics: frame missed -> percentile time

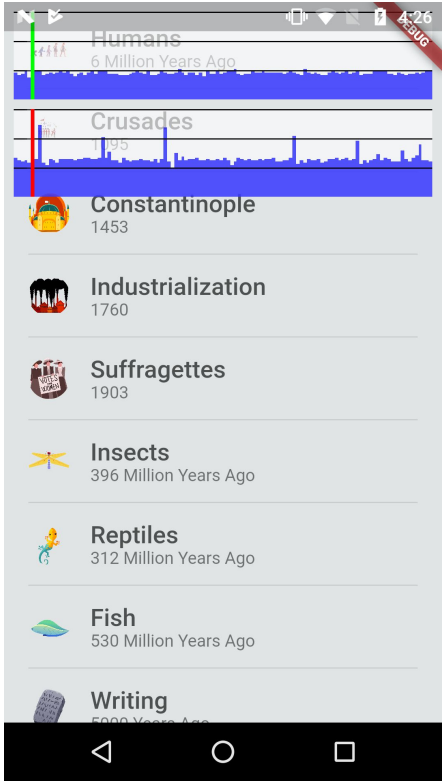


Better metrics: frame missed -> percentile time

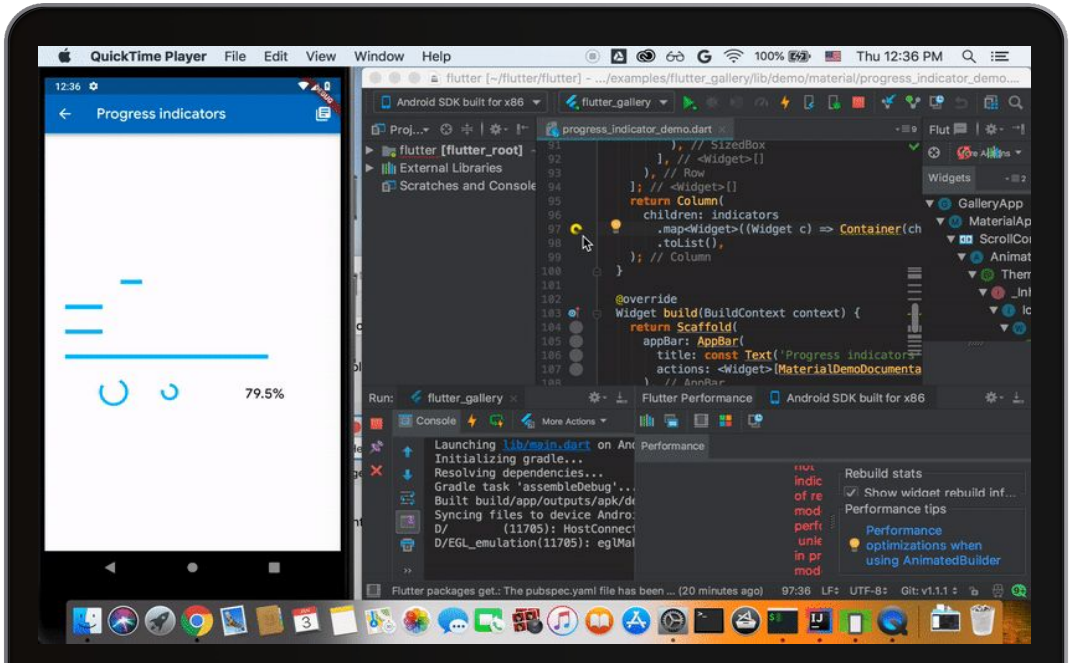


90th percentile, 99th percentile,
100th percentile (worst) frame time

Better feedback: performance overlay



Better feedback: rebuild count



See the GIF here.

Better feedback: measuring debt

- Remove files
- Shorten code
- Reduce dependencies

Summary

- Less work 干活少
  - Less compute work
  - Less engineer work
  - Less developer work
- Better incentive 激励好
  - Better API
  - Better metrics
  - Better feedback

	Contribute	Don’t contribute
Contribute	10, 10	-1, 0
Don’t contribute	0, -1	0, 0

谢谢！Thank you!

github.com/flutter