



# What makes Flutter fast Flutter 为何快

Yuqian Li

李宇骞

liyqian@google.com

# 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++) {
```

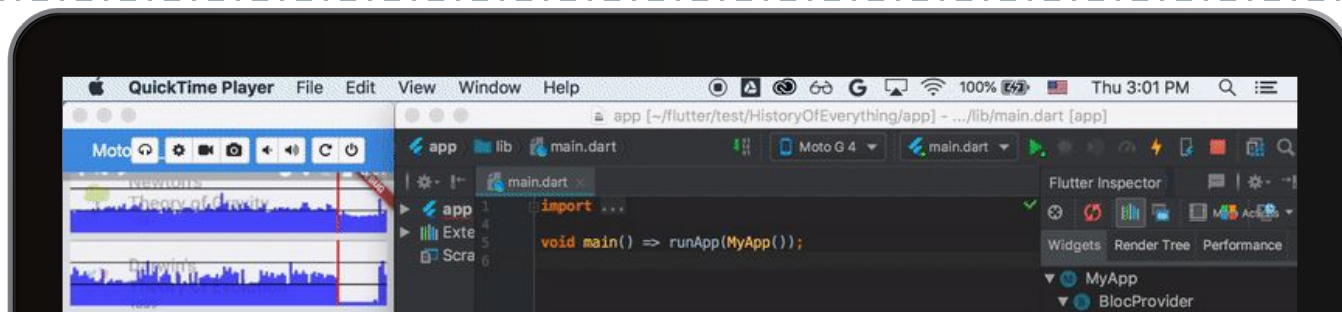
Java: 0.60s

```
import java.util.ArrayList;
```

```
final class jstr {
```

```
  public static void main(String[] args) {
```

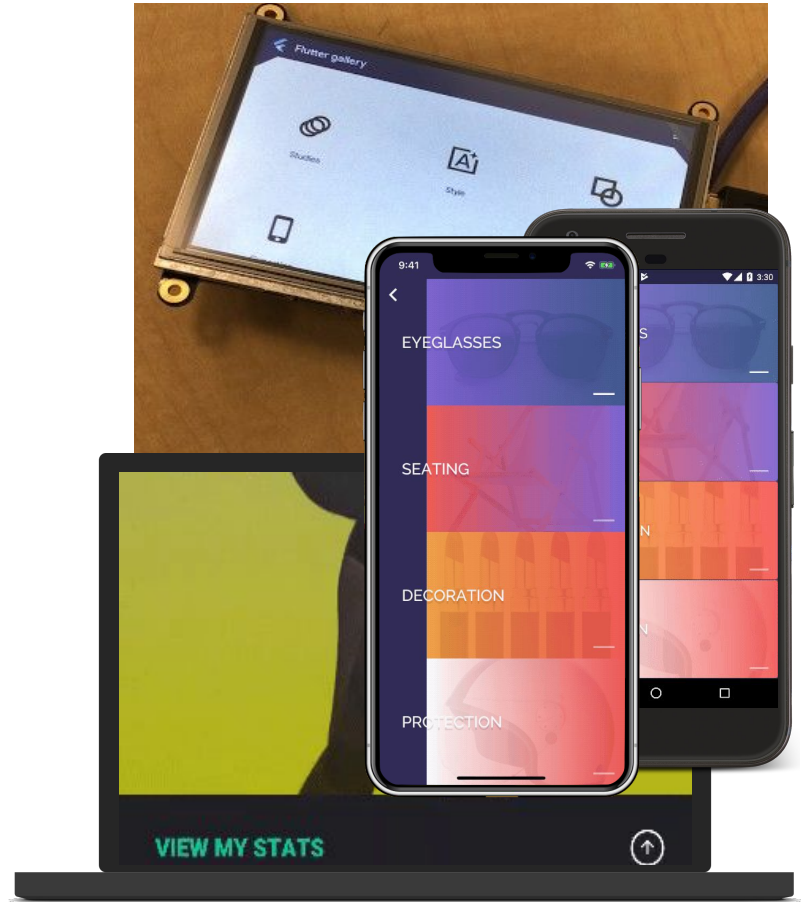
```
    final int T = 10000;
```



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



.....



Beautiful

Productive

Fast Development



Flutter

Fast

Native Performance

Open



CHINESE

↔

ENGLISH

天下武功，唯快不破

Tiānxià wǔgōng, wéi kuài bùpò

拼 ▾

UI 





Verified by Translate Community

JOIN



Screenshot taken from [translate.google.com](https://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;

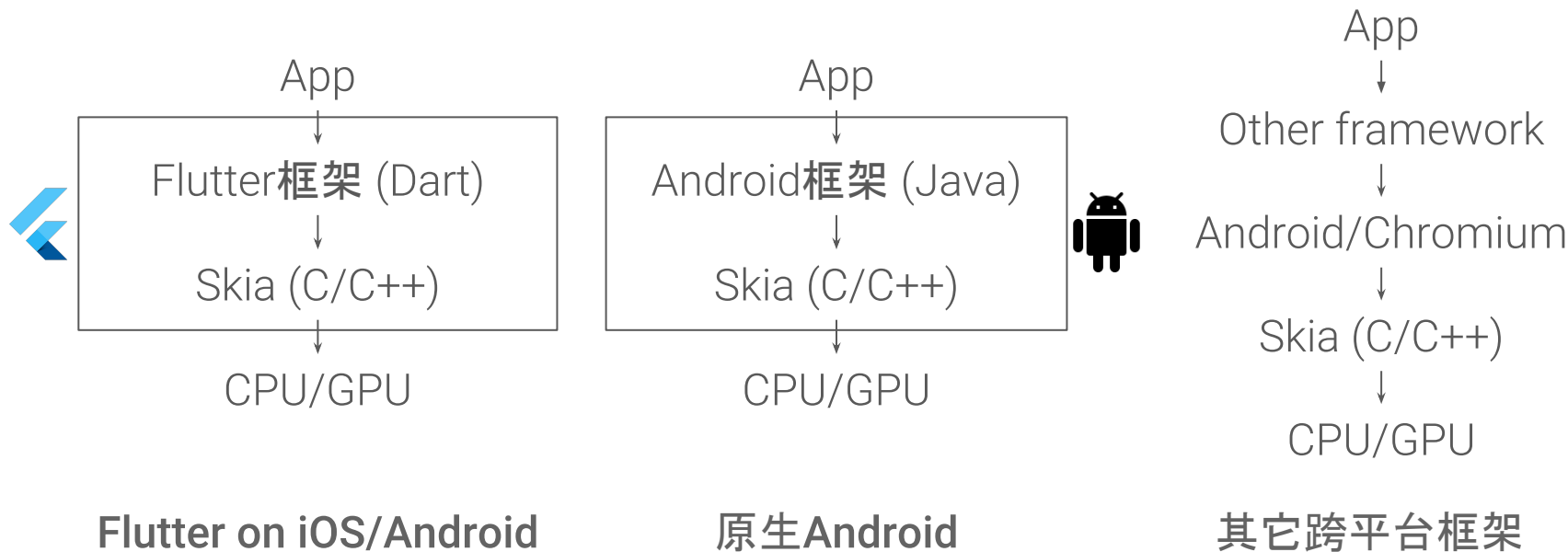
    for(int t = 0; t < T; t++) {
      ArrayList objs[] = new ArrayList[N];
      for(int i = 0; i < N; i++) {
        objs[i] = new ArrayList<Byte>(10);
      }
    }
  }
}
```

Experiments conducted on Linux x64 4.18.10, Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60GHz,

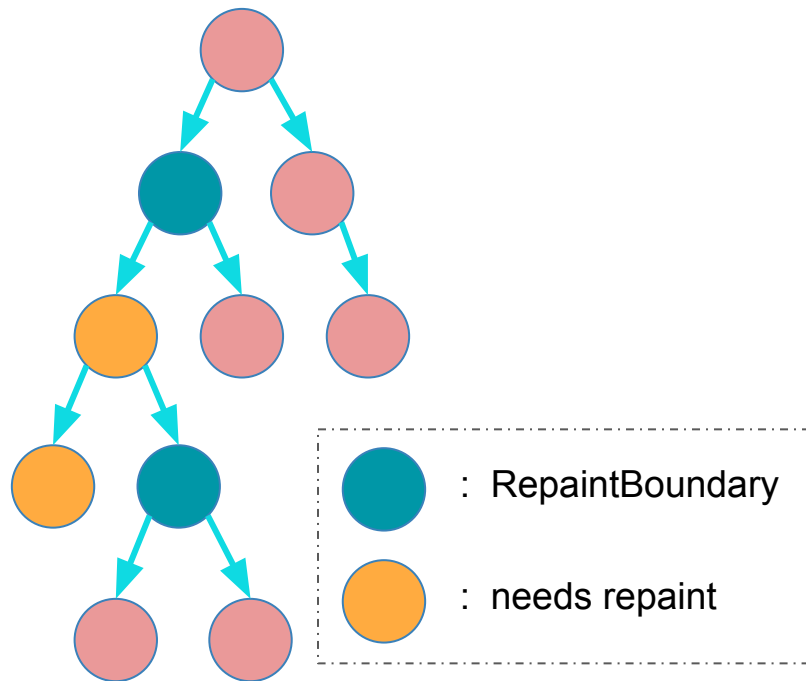
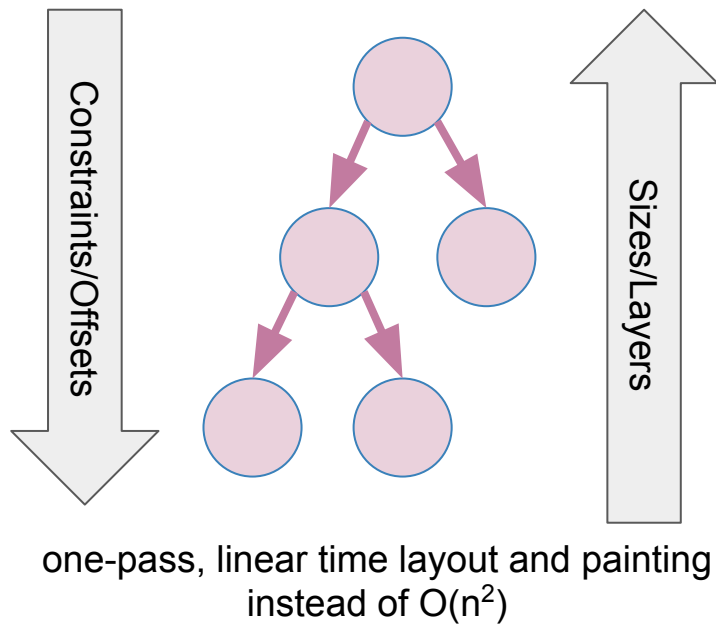
Dart 2.2.0-edge.0797d6ef695739aafd3759096f4143c7440f0213, gcc version 7.3.0 (Debian 7.3.0-5) with -O3, openjdk version 1.8.0\_161



# Less compute work: skip Android/Chromium



# Less compute work: fewer tree walks

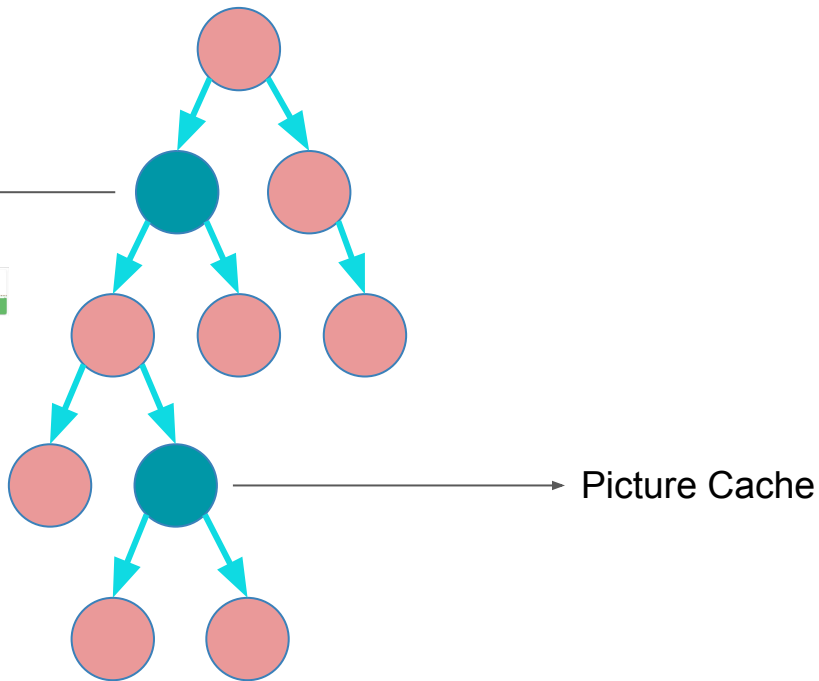


# Less compute work: raster cache

(newly added) Layer Cache

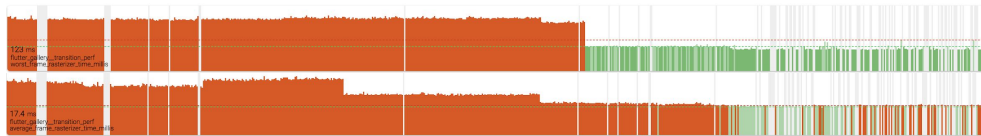
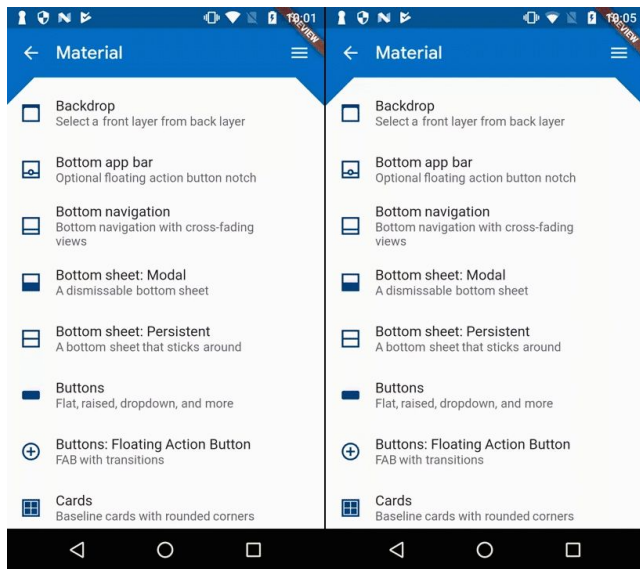


60% speedup in page  
transition animations.



# 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

The image shows a screenshot of the Android Studio documentation interface, highlighting four classes: **Text**, **TextView**, **Flex**, and **LinearLayout**. Red dashed boxes and red circles are used to highlight specific features and annotations.

- Text class:** A dashed box highlights the **Text** class header and its **CONSTRUCTORS** section, which includes `Text` and `rich`. A red circle highlights the vertical scrollbar on the right side of the **Text** class page.
- TextView:** A dashed box highlights the **TextView** class header. A red circle highlights the vertical scrollbar on the right side of the **TextView** class page.
- Flex class:** A dashed box highlights the **Flex** class header and its **CONSTRUCTORS** section, which includes `Flex`. A red circle highlights the vertical scrollbar on the right side of the **Flex** class page.
- LinearLayout:** A dashed box highlights the **LinearLayout** class header and its **CONSTRUCTORS** section, which includes `LinearLayout`. A red circle highlights the vertical scrollbar on the right side of the **LinearLayout** class page.

Annotations include:

- Red dashed boxes around the class headers and their respective **CONSTRUCTORS** sections.
- Red circles around the vertical scrollbars on the right side of the **Text**, **TextView**, **Flex**, and **LinearLayout** class pages.

Below the **LinearLayout** class header, the following text is visible:

```
public class LinearLayout
extends ViewGroup

java.lang.Object
↳ android.view.View
↳ android.view.ViewGroup
↳ android.widget.LinearLayout

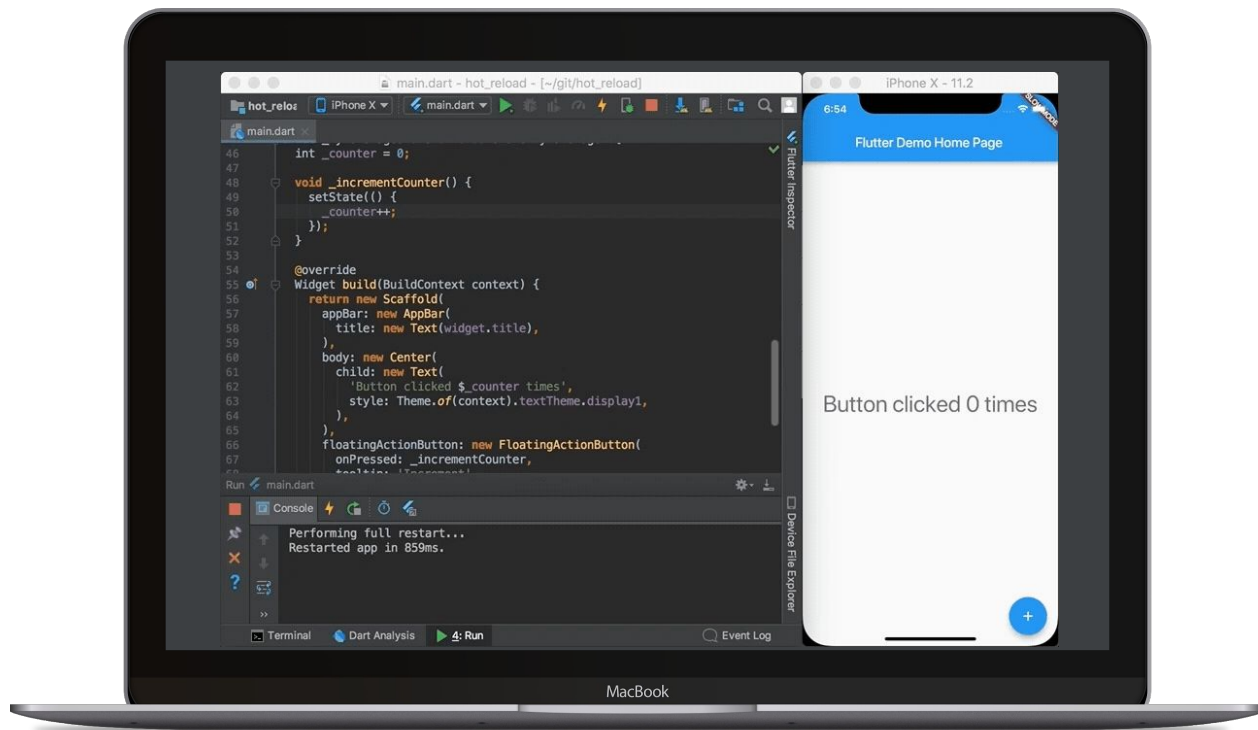
Known direct subclasses
ActionMenuView, NumberPicker, RadioGroup, SearchView, TabWidget, TableLayout, TableRow, ZoomControls
```

A layout that arranges other views either horizontally in a single column or vertically in a single row.

The following snippet shows how to include a linear layout in your layout XML file:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:orientation="horizontal"
    android:gravity="center">
```

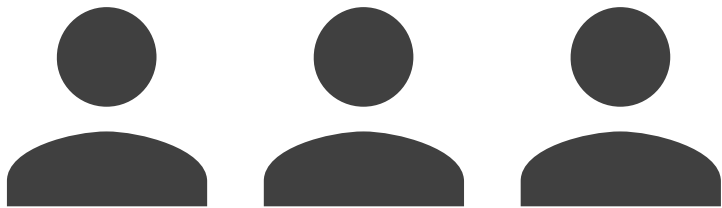
# Less developer work: hot reload



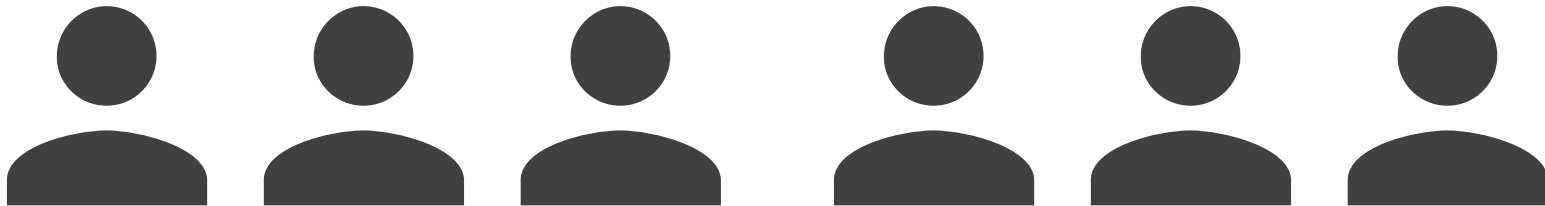
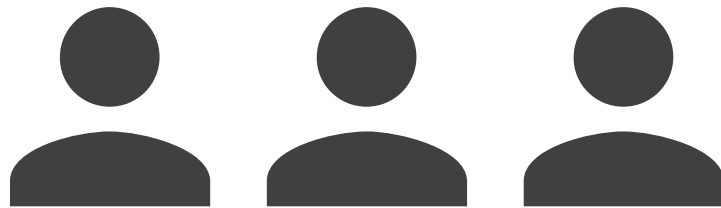
Screenshot from Flutter 1.0 slides

# Less developer work: single codebase

Android Issues



iOS Issues



App Issues



# Less developer work: everything is Dart

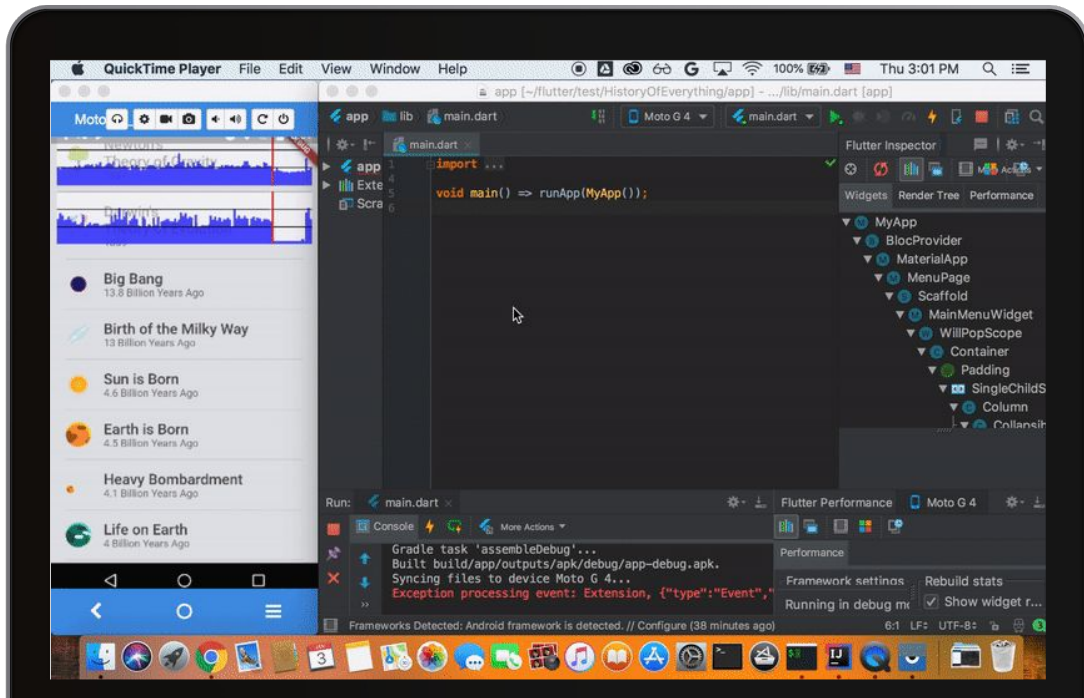
- Fewer to learn



Dart

- 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

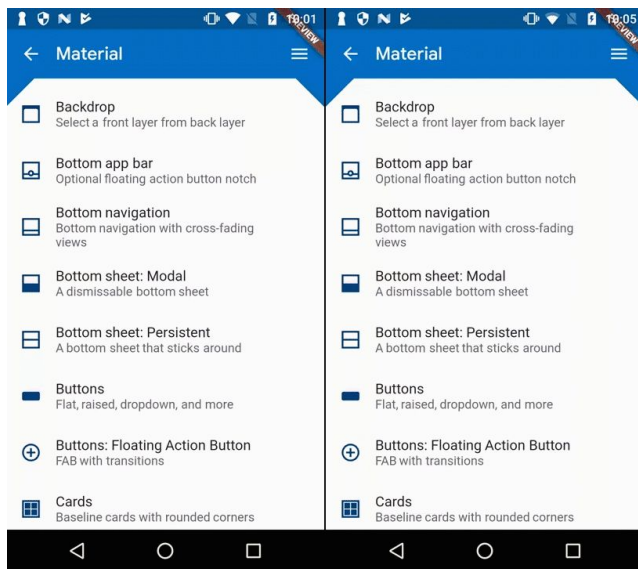
## **Avoid APIs that encourage bad practices**

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



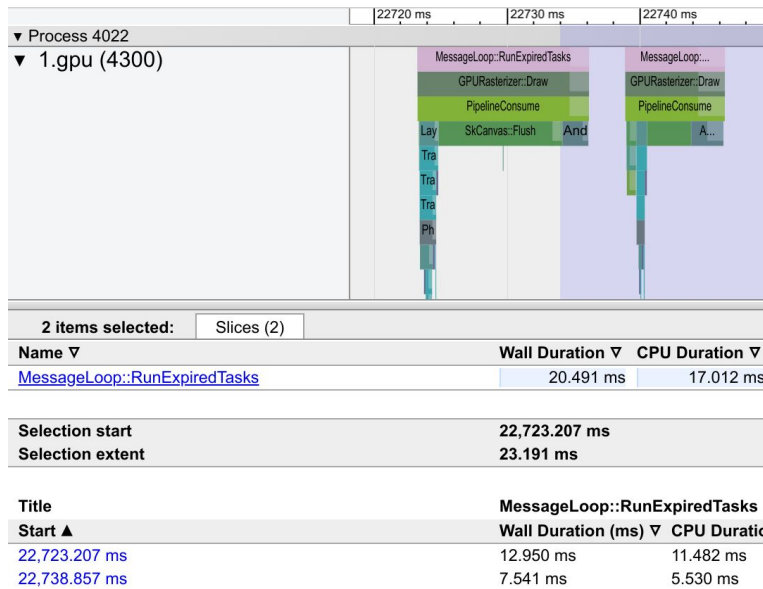
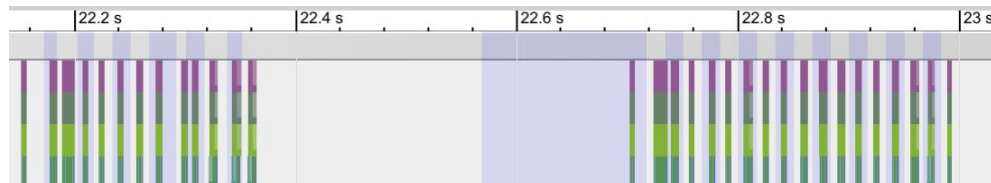
2x speedup

screenshot from GDD China 2018 slides

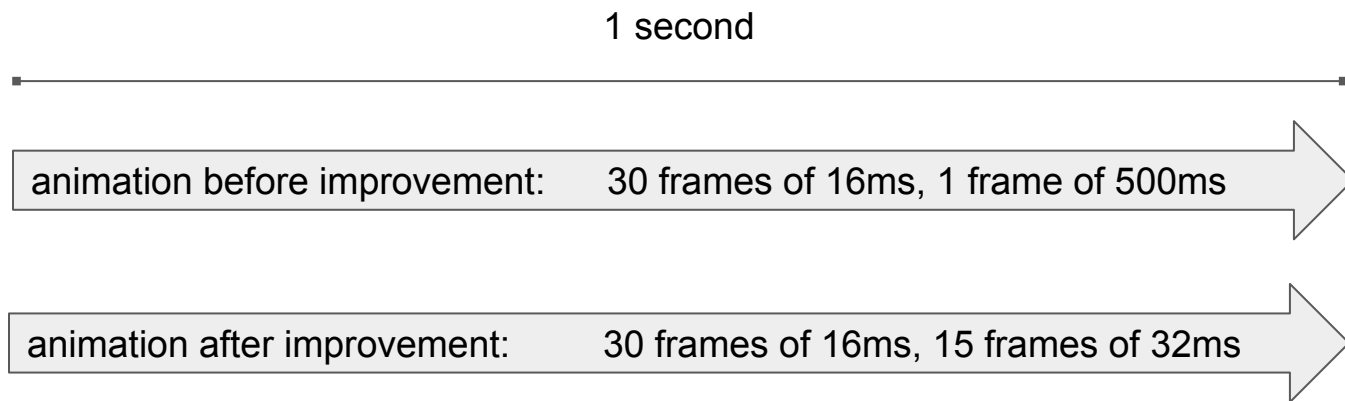
# Better metrics: FPS -> average frame time



[https://commons.wikimedia.org/wiki/File:Nuclear\\_Dawn\\_-\\_Silo\\_FPS\\_01.png](https://commons.wikimedia.org/wiki/File:Nuclear_Dawn_-_Silo_FPS_01.png)



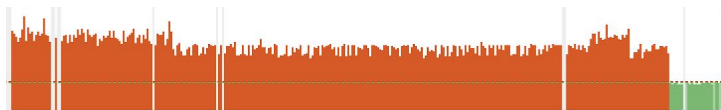
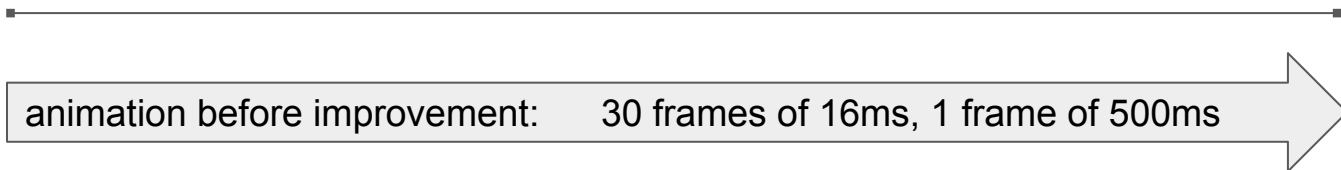
# Better metrics: frame missed -> percentile time





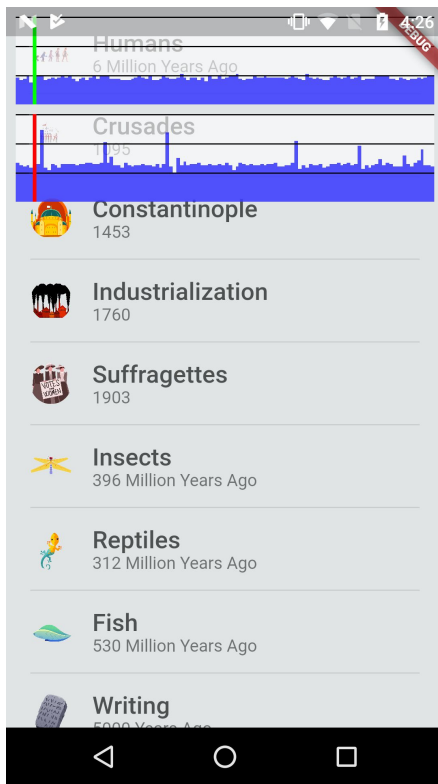
# Better metrics: frame missed -> percentile time

1 second

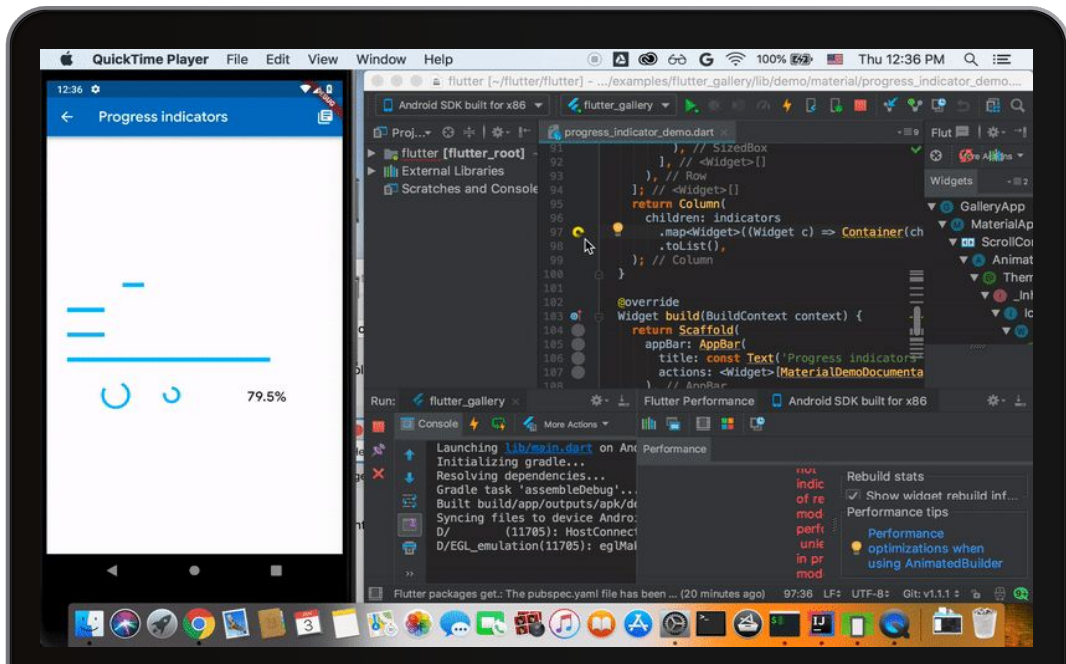


90th percentile, 99th percentile,  
100th percentile (worst) frame time

# 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!

[liyuqian@google.com](mailto:liyuqian@google.com)

[github.com/flutter](https://github.com/flutter)

# 反馈 & 了解更多

关注谷歌官方“谷歌开发者”微信公众号了解更多 Flutter 动态



有任何关于 Flutter 的问题和反馈？请填写下面表格与我们沟通

