# چگونگی ساخت چارچوب های وب توسط گوگل

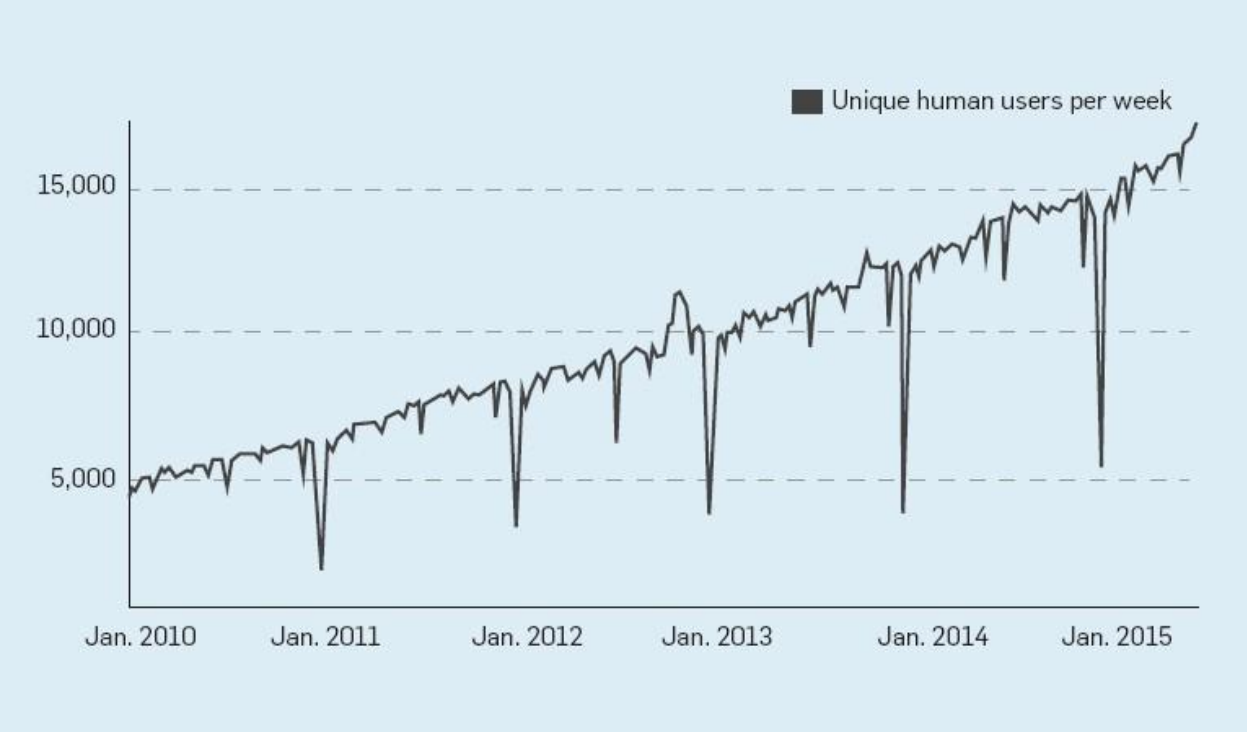
اغلب کاربران حرفه ای می دانند که گوگل با استفاده از یک Repository، تمام ۲ میلیارد خط کد را به اشتراک گذاشته و از الگوی توسعه مبتنی بر ترانک استفاده می کند.

| | |
|---|---|
| Total number of files | 1 billion |
| Number of source files | 9 million |
| Lines of source code | 2 billion |
| Depth of history | 35 million commits |
| Size of content | 86TB |
| Commits per workday | 40,000 |

از نظر بسیاری از برنامه نویسان خارج از این شرکت این روال غیرعادی و عجیب است اما به هرحال این روش جواب داده است.

بسترکدهای گوگل توسط بیش از ۲۵ هزار توسعه دهنده نرم افزار گوگل ازتعداد زیادی شرکت در کشورهای مختلف به اشتراک گذاشته شده است. این افراد در یک روز معمولی بیش از ۱۶ هزار تغییر را روی این بستر اعمال می کنند.

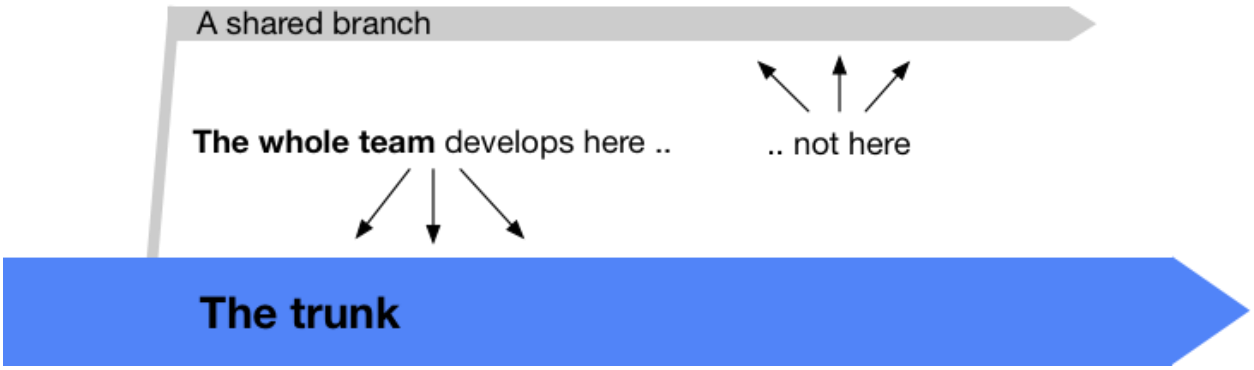
این مقاله به تعریف چگونگی ساخت یک چارچوب وب متن باز مرتبط می شود.




اصطلاح Human Users به معنی مهندسان نرم افزاری است که در گوگل کدنویسی می کنند (در برابر ابزارهای تولید منبع)

## تنها یک نسخه

هنگامی که شما از توسعه مبتنی بر ترانک در یک انبار بزرگ استفاده می کنید، تنها یک نسخه از هرچیز را در اختیار دارید. گفتنی است که در گوگل نمی توانید اپلیکیشنی مثل FooBar را داشته باشید که از AngularDart 2.2.1 استفاده کند و در عین حال اپلیکیشن دیگری مثل BarFoo را داشته باشید که از نسخه .2.3.0 آن استفاده کند چرا که هر دو اپلیکیشن باید از نسخه یکسانی استفاده کنند.

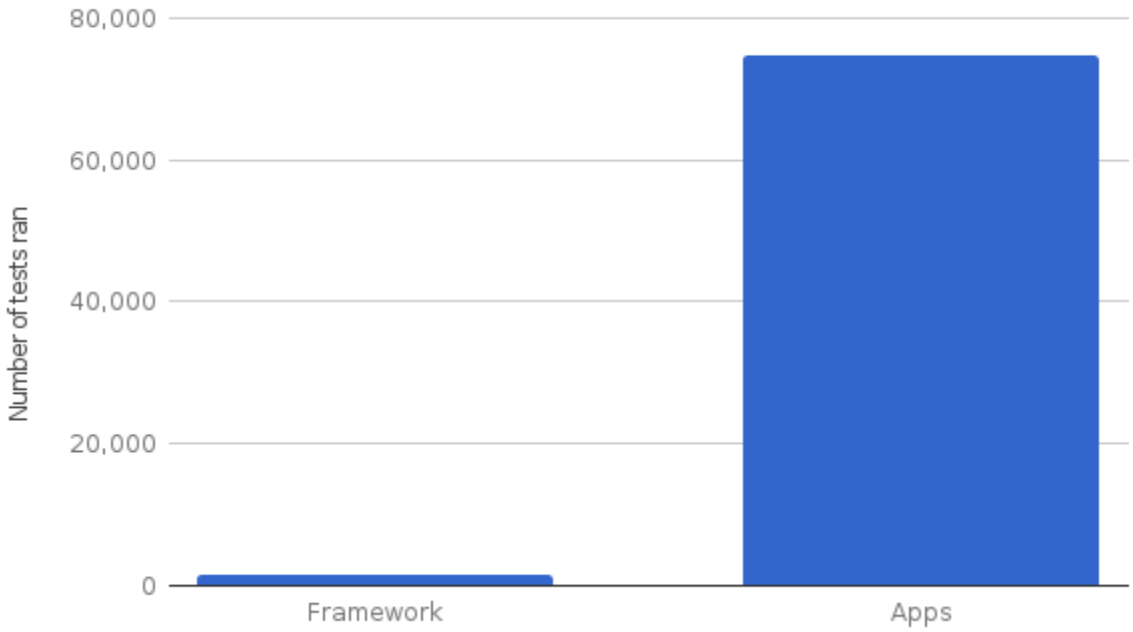



به همین دلیل است که کارکنان گوگل می گویند در این شرکت تنها از آخرین نسخه های هر نرم افزاری استفاده می شود.

اگر از نظر شما این کار خطرناک است، خب حق با شماست. با این حال گوگل فکر اینجا را هم کرده است.

## ۷۴ هزار آزمایش به ازای هر تغییر

AngularDart ۱۶۰۱ آزمایش را تعریف می کند اما هنگامی که شما در repository گوگل تغییری را در کد AngularDart ایجاد می کنید، این چارجوب تست ها را برای تمام کسانی که در گوگل از این چارچوب استفاده می کند اجرا می کند. در این صورت تعداد این تست ها به ۷۴ هزار عدد می رسد. (بسته به اینکه  این تغییر چقدر بزرگ است ممکن است تعدادی از این آزمایش ها اجرا نشوند)




به هر حال اجرای آزمایش های بیشتر بهتر است. من در این کد تغییری را ایجاد کرده ام که ۵ درصد مواقع خودش را نشان می دهد و چیزی مثل شرایط مسابقه را در الگوریتم تایید مجدد شناسایی تغییرات شبیه سازی می کند ( خط  && random.nextDouble() > .05 را به کدها اضافه کرده ام).  هنگام اجرای این کد در هیچ یک از ۱۶۰۱ آزمایش مشاهده نشده اما در تعدادی از آزمایش های کلاینت ظاهر شده است.

مزیت اصلی این تست ها این است که در اپ های واقعی اجرا می شوند. علاوه براین نه تنها تعداد آنها بسیار زیاد است بلکه چگونگی استفاده از چارچوب توسط توسعه دهندگان را نیز نمایش می دهد. این مساله اهمیت زیادی دارد چرا که مالکان چارچوب گاهی اوقات نحوه استفاده از چارچوبشان را به درستی برآورد نمی کنند.

از طرف دیگر این روند باعث می شود که میلیاردها دلار سرمایه به سمت برنامه های در حال توسعه سرازیر شوند. بین اپ های آزمایشی که نویسنده چارچوب در اوقات فراغت خود توسعه می

دهد و برنامه های واقعی که ده ها یا صدها نفر در آن سرمایه گذاری کرده اند تفاوت های زیادی وجود دارد . اگر قرار است وب در آینده بهبود پیدا کند باید پشتیبانی بیشتری صورت گیرد.

شما آن را با مشکل مواجه می کنید، خودتان هم آن را تعمیر می کنید

زمانی که نویسندگان AngularDart تغییری را ایجاد می کنند که کاربران دیگر را با مشکل مواجه می کند، باید آن را اصلاح کنند. از آنجایی که از گوگل در یک مخزن نگهداری می شود، پیدا کردن مواردی که تغییرات روی آنها اثر گذاشته ساده بوده و بلافاصله می توان فرایند اصلاح را آغاز کرد.

هر تغییری که در AngularDart باعث مشکل شود شامل تمام اصلاحات در برنامه های تحت تاثیر قرار گرفته گوگل هم می شود، بنابرین کد مشکل دار و اصلاح آن به صورت همزمان در مخزن اجرا میشوند.

برای مثال زمانی که اعضای تیم AngularDart تغییری را ایجاد می کند که روی اپلیکیشن AdWords تاثیر می گذارد، آنها به سراغ کد منبع اپلیکیشن رفته و آن را اصلاح می کنند. این افراد می توانند تست های موجود AdWords را در فرایند اجرا کرده و کدهای جدید را به آن اضافه کنند. آنها تمام موارد مذکور را در لیست تغییرات قرار داده و از دیگران می خواهند آنها را بازبینی کنند. از آنجایی که این تغییرات شامل مخزن AngularDart و مخزن AdWords می باشد، سیستم به صورت خودکار مسلتزم پذیرس بررسی کد از سوی هردو تیم بوده و تنها در این صورت تغیرات اعمال می شوند.

این مساله باعث توسعه چارچوب به صورت مشترک می شود. توسعه دهندگان چارچوب AngularDart به میلیون ها خط کدی دسترسی دارند که با استفاده از پلتفرم آنها توسعه داده شده است. آنها نیازی ندارد که به چگونگی استفاده از چارچوبشان فکر کنند.( تنها مشکلی که می تواند وجود داشته باشد این است که آنها تنها کد گوگل را در نظر گرفته و و کدهای Workivas، Wrikes و StableKernels که از AngularDart استفاده می کنند را در نظر نگیرند).

الزام به بروزرسانی کد کاربری فرایند توسعه را کند می کند اما نه آنقدر که فکرکنید (به فرایند پیشرفت AngularDart از ماه اکتبر نگاه کنید). این کند شدن روند بسته به اینکه از یک چارچوب چه انتظاراتی دارید، مزایا و معایب خاص خود را دارد. بعدا به این مساله می پردازیم.

به هرحال دفعه بعدی که یکی از کارکنان گوگل به شما گفت که نسخه آلفای یک کتابخانه پایدار بوده و در حال تولید است دلیل آن را می دانید.

## تغییرات در مقیاس بزرگ

فرض کنید که یک تغییر بزرگ در AngularDart رخ داده و مثلا از نسخه ۲ به ۳ اپدیت شود، اگر این تغییر در تمام ۷۴ هزار آزمایش دچار مشکل شود چه اتفاقی رخ می دهد؟ آیا اعضای تیم به سراغ این ۷۴ هزار مورد رفته و همه آنها را اصلاح می کنند؟ آیا آنها در هزاران فایل منبع که در توسعه اکثرآنها نقشی نداشته اند، تغییراتی ایجاد می کنند؟

بله

یکی از مزایای داشتن یک سیستم صوتی این است که از ابزارهای قدرتمندی برخودار خواهید بود. برای مثال در Dart صوتی، در صورت نیاز به بازسازی بسیاری از تغییرات به صورت خودکار انجام شده و نیازی به تائید از سوی توسعه دهنده وجود ندارد.

زمانی که در کلاس Foo یک روش از ()bar به ()baz تغییر پیدا می کند، می توانید ابزاری را ایجاد کنید که تمام مثال های کلاس و زیرکلاس های Foo را در مخزن پیدا کرده و تغییرات لازم را اعمال کند.

با استفاه از Dart صوتی می توانید مطمئن باشید که مشکل خاصی پیش نخواهد آمد اما بدون استفاده از آن حتی چنین تغییر کوچکی هم می تواند مشکلات بزرگی را ایجاد کند.

بر اساس راهنمای Dart در صورت دستکاری یک کلید ممکن است تمامی کدهای شما پاک شوند. در واقع این راهنما تاکید می کند که قواعد کنترل whitespace-handling برای Dart آن چیزهایی هستند که dart_style تولید می کند.

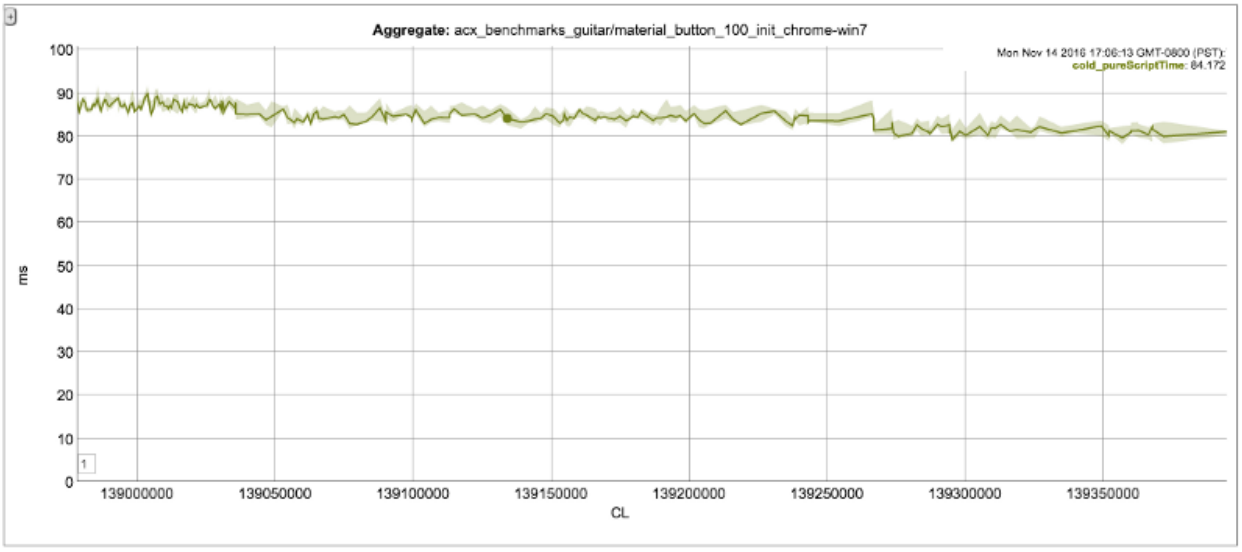
```dart
List<IntegerPair> numbers = [new IntegerPair(1, 82), new IntegerPair(2, 79)];

void addInput() {
  numbers.add(new IntegerPair(numbers.length + 1, 123));
}

compute() async {
  if (_targetsChanged()) {
    _sendPort.send({
      "type": "register-targets",
      "value": numbers.map((pair) => pair.value).toList()
    });
    foundIntegers.clear();
  }
  _sendPort.send({"type": "start"});
  isolateReady = false;
  isolateComputing = true;
}

void deleteByKey(int key) {
  numbers.removeWhere((pair) => pair.key == key);
  recomputeKeys();
}

void recomputeKeys() {
  for (int i = 0; i < numbers.length; i++) {
    numbers[i].key = i + 1;
  }
}

String _buildTargetsFingerprint() =>
    (numbers.map((pair) => pair.value).toList()..sort()).join(',');

void forceStop() {
  _sendPort.send({"type": "force-stop"});
}
```

یکی دیگر از نکاتی که امکان اجرای تغییرات گسترده در dart_style را فراهم می کند، فرمت کننده پیش فرض آن است. تمام کد Dart در گوگل را می توان با استفاده از این ابزار فرمت کرد. زمانی که کد شما به بررسی کنندگان می رسد، با استفاده از dart_style فرمت می شود بنابرین نیازی به بحث درباره جایگذاری کدها وجود نداشته و بازسازی های لازم نیز صورت می گیرند.

## معیارهای کارایی

همانطور که قبلا گفتیم یکی از مزایای AngularDart تست های ان است. اما این ماجرا به همین جا ختم نمی شود. گوگل نسبت به ارزیابی کارایی برنامه هایش حساسیت زیادی دارد و به همین دلیل بسیاری از اپ های آن Benchmarkهای دقیقی دارند.

بنابرین زمانی که تیم توسعه تغییری را در AngularDart ایجاد می کنند که یک برنامه را ۱ درصد کندتر می کند آنها قبل از اعمال تغییر از آن خبردارند. زمانی که این تیم در ماه اکتبر اعلام کرد که اپ های AngularDart ۴۰ درصد کوچکتر و ۱۰ سریعتر از ماه اگوست شده اند، آنها از اپ های مهم و واقعی با چندین مگابایت کد تجاری و میلیون ها کاربر صحبت می کردند.



## ابزار توسعه جادویی

شاید شما هم از خود بپرسید که هنگام ایجاد یک باگ در AngularDart باید چه آزمایش هایی را در این مخزن عظیم انتخاب کنیم؟ مطمئنا نیازی به دست چین کردن ۷۴ هزار آزمایش یا اجرای همه آنها وجود ندارد و پاسخ در ابزاری به نام Bazel نهفته است.

در چنین مقیاسی نمی توانید از اسکریپت های ساده استفاده کنید چرا که سرعت عمل تا حد زیادی پایین می آید اما آنچه به آن نیاز دارید یک ابزار جادویی است.

جادویی در این قسمت تا حد زیادی به واژه خالص در توابع شباهت دارد. مراحل توسعه برنامه نباید اثرات جانبی داشته باشد (مثل فایل های موقتی یا تغییر PATH و غیره)، و از طرف دیگر باید قطعی باشند ( یک ورودی یکسان همیشه به یک خروجی یکسان ختم شود). در این صورت هنگام اجرای ان در هر دستگاهی و در هر زمانی خروجی های پایداری خواهید داشت. نیازی نیست که نسخه های شما کامل و جامع باشند و می توانید نسخه های آزمایشی را به سرورهای تولید بفرستید.

گوگل چندین سال را صرف توسعه چنین ابزاری کرده و از سال گذشته آن را با نام Bazel به صورت متن باز در اختیار کاربران قرار داده است.

حالا به لطف این قابلیت، ابزارهای آزمایش داخلی توانایی تعیین اثر تغییرات بر نسخه های آزمایشی را داشته و می توانند آنها را در زمان مناسب اجرا کنند.

## این موارد به چه معنی هستند؟

هدف اصلی AngularDart بهترین بودن در تولید، کارایی و قابلیت اطمینان در ایجاد اپلیکیشن های وب است. این مقاله به آخرین گزینه یعنی قابلیت اطمینان و اهمیت استفاده اپلیکیشن های مهم گوگل از قبیل AdWords و AdSense از این چارچوب می پردازد.

همانطور که در بالا توضیح دادیم اهمیت AngularDart تنها در تیم توسعه آن خلاصه نمی شود، بلکه برخورداری از تعداد انبوهی از کاربران احتمال ایجا تغییرات سطحی در این چارچوب را به حداقل رسانده و آن را قابل اعتماد ساخته است.

```
main() {
  bootstrap(PrimeFinder);
}
```

اگر به دنبال چارچوبی هستید که تغییرات اساسی به همراه داشته و هر چند ماه یکبار قابلیت های مهمی به ان اضافه شود، باید دور AngularDart را خط بکشید حتی اگر تیم توسعه دهنده AngularDart به دنبال توسعه چارچوبی به این طریق بود، با توجه به موارد بالا مشخص است که امکان آن را نداشت.

از نظر من بهترین پیش بینی برای پشتیبانی بلندمدت از یک پشته فناوری متن باز، این است که بخش مهمی از کسب و کار توسعه دهنگان باشد. برای مثال MySQL را در نظر بگیرید که کتابخانه اندروید به شمار می رود.

به همین دلیل خوشحالم که Dart در نهایت از یک چارچوب وب (AngularDart)، یک کتابخانه اجزا (AngularDart Components) و یک چارچوب موبایل (Flutter) برخوردار شده و از تمام آنها می توان برای توسعه اپلیکیشن های حیاتی گوگل استفاده کرد.