

# Web的未来

新技术和新语言带来的改变

韩国恺

[hanguokai@gmail.com](mailto:hanguokai@gmail.com)

开发者9Y+ 技术爱好者 Google fans

韩国恺

@hanguokai

学习 动漫、电影 健身

上网



# DART



与 Dart 的缘分



# The Future

NEXT EXIT 

# 大纲

## 1. Web的演化

## 2. 语言篇

- a. 语言的发展

- b. Dart 介绍

## 3. 技术篇

- a. Web Components

- b. Polymer 库

# Web应用的优点

- 无需安装
- 增量式开发
- 自动升级
- 跨平台
- 天然的MVC (HTML+CSS+JavaScript)

# 当前的Web开发

- Web 开发越来越复杂：项目大，团队成员多
- 前端功能越来越多：CS→BS→CS
- 大型应用维护和协作难
- 性能弱，启动速度慢
- 工具支持弱
- 仔细选择技术方案和开发规范

**Web 开发中还有许多问题有待解决...**



**挫 折 禁 止**

# 未来的方向？





# Chrome OS 和 Chromebook

Chrome OS 基于浏览器和云端环境的 OS

Chromebook 搭载 Chrome OS 的笔记本

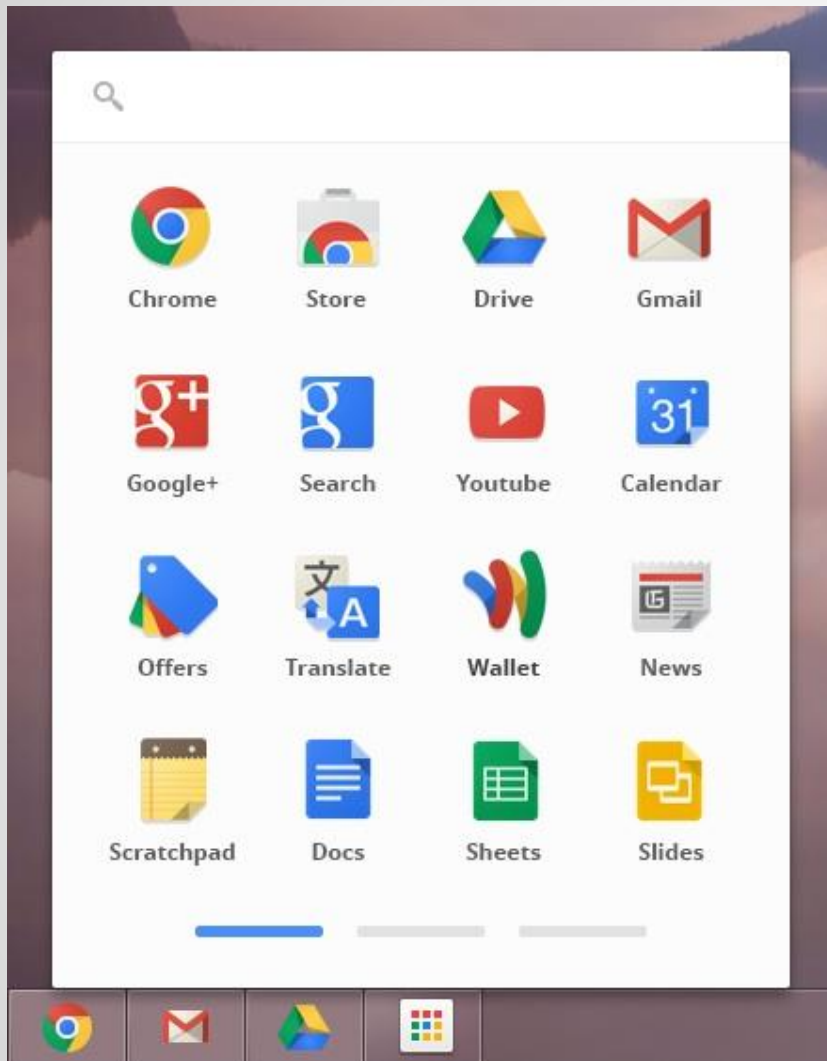
今天的讲义也是用 Google doc 做的



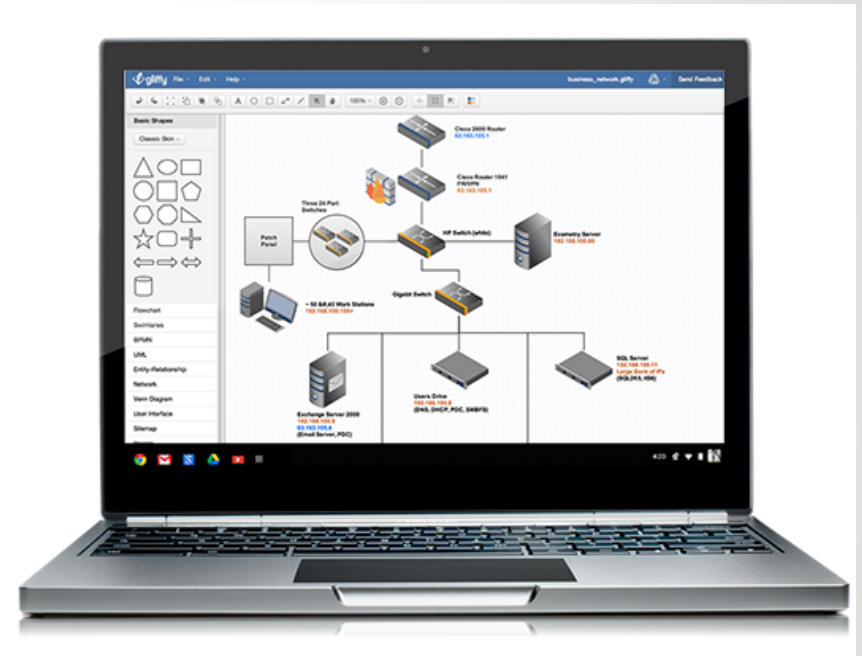
PC销量下降, 而  
Chromebook 销  
量增长

# Chrome App vs Native App

1. 传统的浏览器用来展示内容
2. 而作为 App 并不是人们最习惯的使用方式,  
Web App 应该给人以桌面 App 的“感觉”
3. Web App 应该更像 App 而不是一个浏览器  
Tab
4. 用户并不关心背后用的是什麼技术。
  - a. App做壳, Web做核
  - b. Web 会与 OS 更紧密集成, 直接提供 Web引擎支持



Chrome App Launcher



Chrome Apps

# Chrome App vs 桌面 App

技术上:

- 独立的窗口
- 离线使用
- 通知栏
- App Launcher
- 数据云同步: chrome sync 内置支持
- 平台访问: USB、蓝牙、Socket、Native Client

# 离线应用

利用离线技术将资源存在本地。如 Google Drive, 飞机上也可用继续查看和编辑今天的这个讲义。

	Web		Android		iOS	
	View	Edit	View	Edit	View	Edit
My Drive	✓		✓	✓	✓	✓
Documents	✓	✓	✓		✓	
Spreadsheets	✓		✓		✓	
Presentations	✓	✓	✓		✓	
Drawings	✓	✓	✓		✓	
Other Drive files			✓		✓	

# 云+端——同步无处不在

我的设备：3个笔记本、1个台式机、iPhone 和 Android 平板

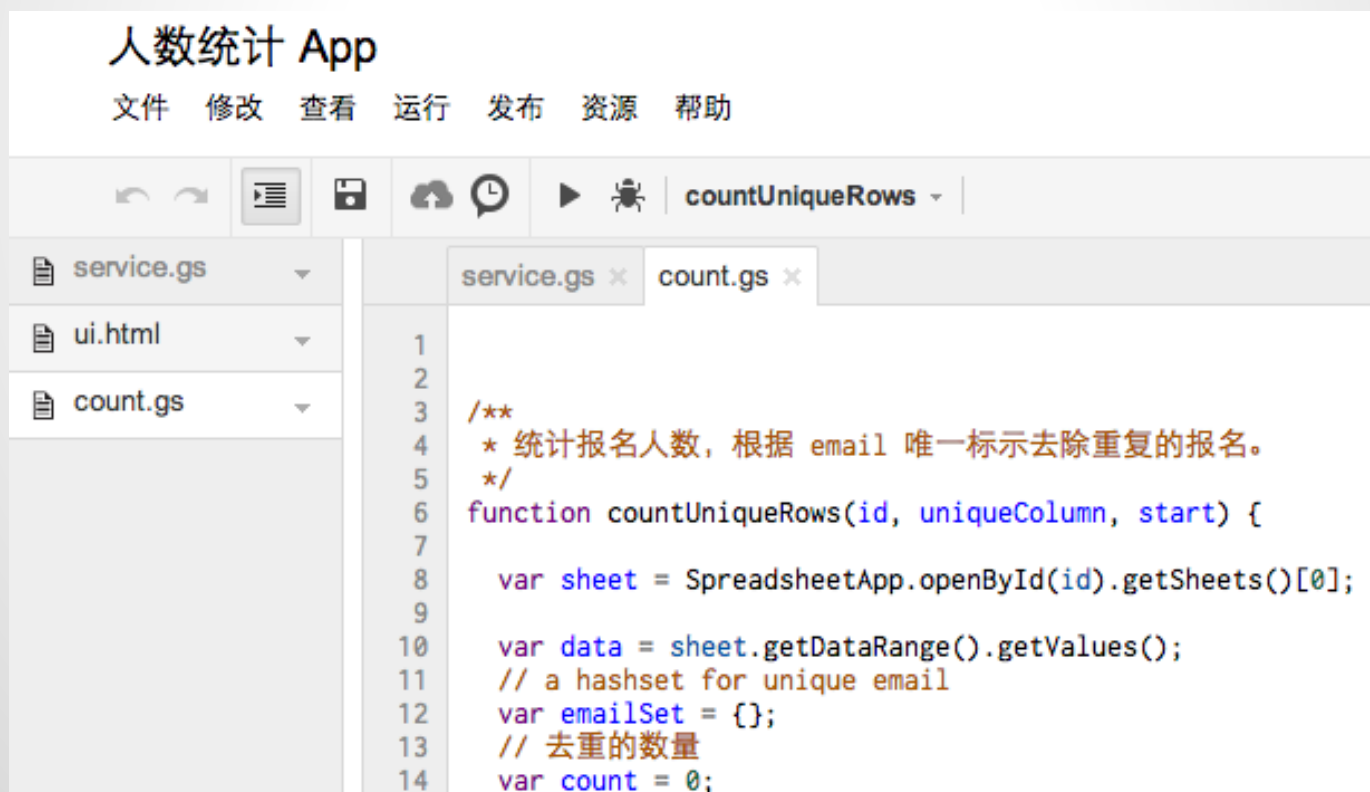


# 云+端——同步无处不在



# Google Apps Script

GAS 运行在云端的 JavaScript, 打通所有 Google 产品和服务, 自动化处理业务。





# 云服务

X-aaS 或 Web API，直接使用服务，这些都非常方便，但实现被隐藏了。

基础平台商提供现成的服务，开发者只需要懂软件、懂系统，但不需要懂硬件了。

# 安全

HTTPS 无处不在

Sandbox 模型

# 技术发展方向

- 演进中的各种新技术标准
- 编程语言的多元化

Web 是平台，能在上面做什么归根结底是技术标准和浏览器支持说了算。

技术可以被 hack 的方式玩出新花样，但能力本身并没有改变。

# 技术发展方向

1. 能力的扩展来自新技术: 如 WebRTC 和 Web Components。
2. 开发方式会有不同流派, 但基础技术的支撑很重要, 如编程语言。

# 大纲

1. Web的演化

2. 语言篇

a. 语言的发展

b. Dart 介绍

3. 技术篇 (Web Components)

# Web语言的发展

1995~2013

# JavaScript

1995, 18年前  
Brendan Eich



# JavaScript

# JavaScript 的发展

多范式：函数式、OO、命令式

1995 年诞生于 Netscape

1998 年成为 ECMA 标准

2006 年 jQuery 发布

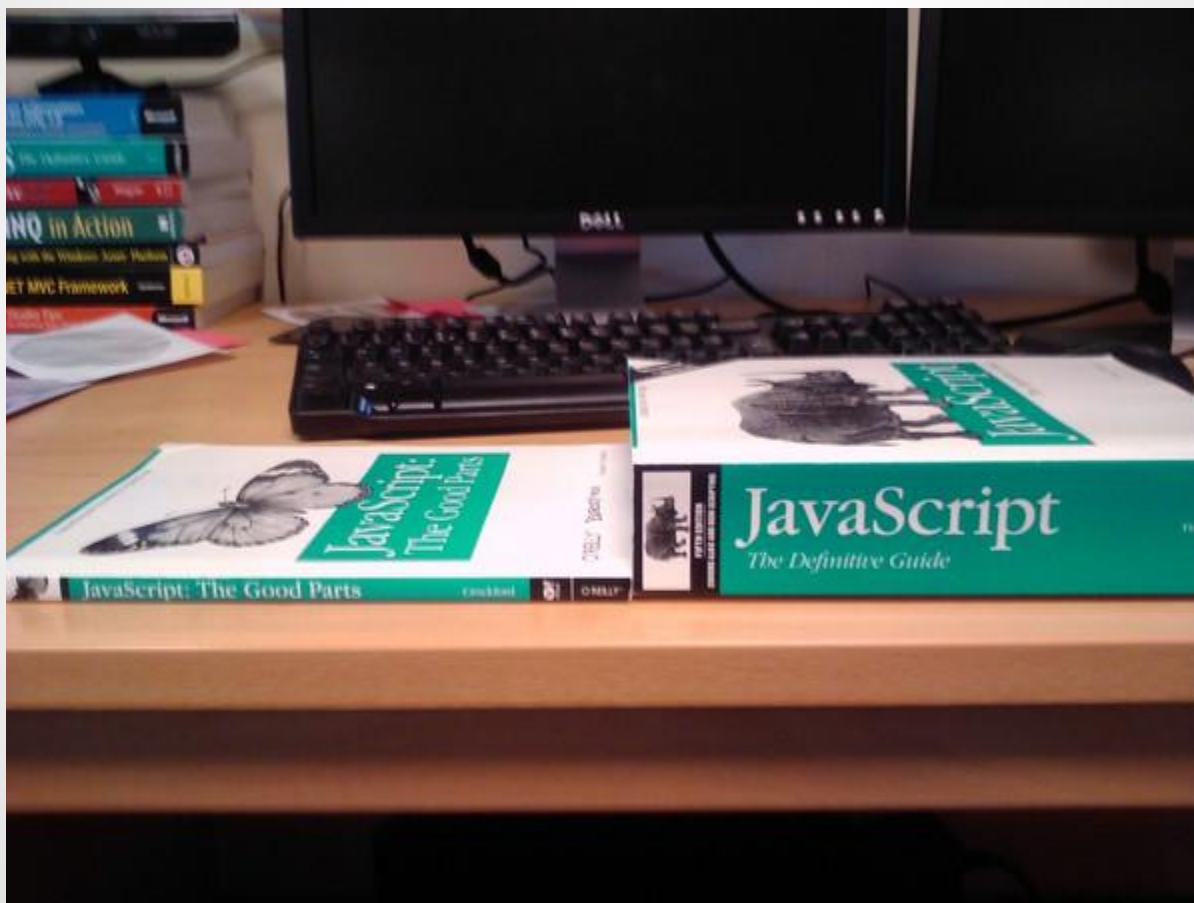
2009 年 Node.js: JavaScript 延伸到 Server 端

近两年: AMD、CommonJS 模块化标准

当前: 正在制定 ECMAScript 6 标准



# JavaScript 并不完美



# JavaScript 并不完美

- 程序结构不明显
- 并非为性能设计的语言
- 缺少一些基础功能的支持
- 缺少统一的基础特性，共享复用难

# 变量提升(hoisting)

```
var foo = 'top-level';
```

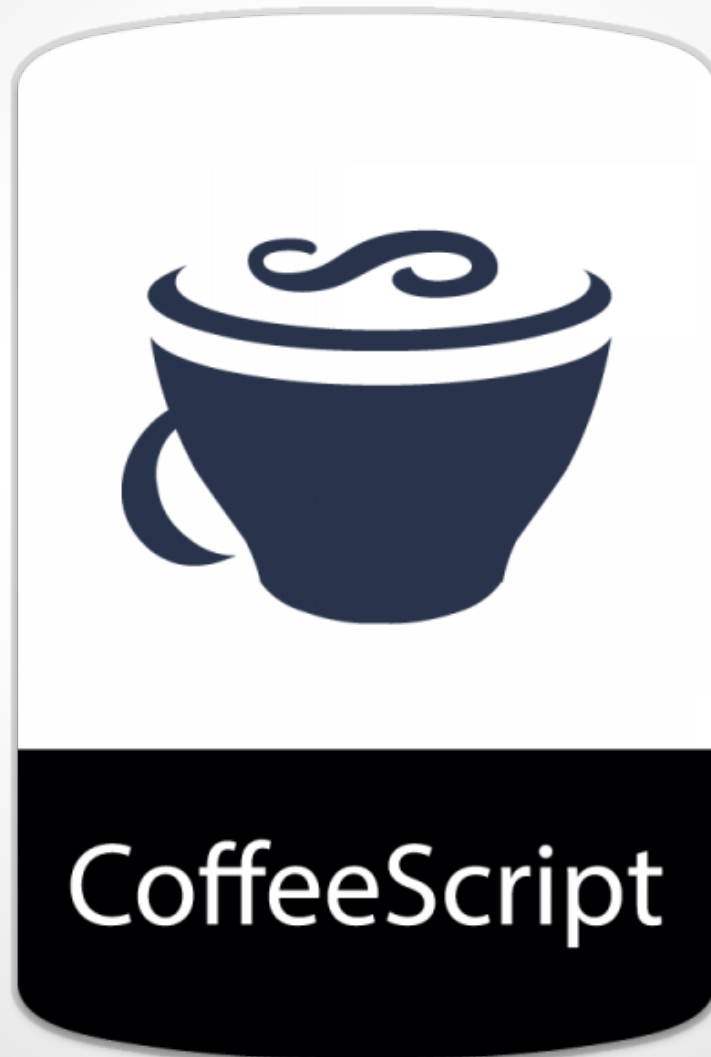
```
function bar() {  
  if (!true) { var foo = 'inside'; }  
  console.log(foo);  
}
```

in JavaScript 输出: undefined

in Dart            输出: 'top-level'

# CoffeeScript

2009, 4年前  
Jeremy Ashkenas



# CoffeeScript

受 Ruby 和 Python 影响较多

- 简洁、语法清晰
- 缩进表示
- 变量声明和作用域
- 支持类
- .....

# CoffeeScript

```
#coffee
```

```
alert "I knew it!" if elvis?
```

```
//JavaScript
```

```
if (typeof elvis !== "undefined" &&  
    elvis !== null) {  
    alert("I knew it!");  
}
```

# CoffeeScript

```
math =  
  root:    Math.sqrt  
  square:  square  
  cube:    (x) -> x * square x  
#Coffee
```

```
math = {  
  root: Math.sqrt,  
  square: square,  
  cube: function(x) {  
    return x * square(x);  
  }  
};  
//JavaScript
```

# Dart

The Google logo, featuring the word "Google" in its characteristic multi-colored font (blue, red, yellow, blue, green, red).

# DART

2011, 2年前

Lars Bak等



# TypeScript



2012, 1年前  
Anders Hejlsberg

# TypeScript

和 Dart 类似的初衷，但更兼容 JavaScript

- 类
- 类型
- 模块
- .....

# TypeScript

```
class Greeter {  
    greeting: string;  
    constructor(message: string) {  
        this.greeting = message;  
    }  
    greet() {  
        return "Hello, " + this.greeting;  
    }  
}  
  
var greeter = new Greeter("world");
```

# 在Web上试用其它编程语言

- Dart <http://try.dartlang.org/>
- TypeScript <http://www.typescriptlang.org/Playground/>
- CoffeeScript <http://coffeescript.org/>

CoffeeScript等编译器就是 JS 实现的, 可以在线直接编译。  
Dart 有所不同, dart2js 编译器 是 Dart 语言实现的, 然后 dart2js 自己把自己编译成了 JS。

甚至这些非编译为 JavaScript 的语言:

Go <http://play.golang.org/>

但原理是将代码上传到服务器(沙盒中)执行, 并返回输出

# 其它数十种编译到JS或JS的扩展语言

- ClojureScript
- LiveScript
- RubyJS
- GWT
- Asm.js
- .....

more languages search by “[List of languages that compile to JS](#)”

# 各种技术所做的改进

	Structure	Syntax	Semantics	Tools	Core Libs	Requires Compilation for Development	Performance
<i>Vanilla JS</i>	----	----	----	----	----	No	----
<b><i>Dart</i></b>						No	
<i>Closure</i>			----			Yes	----
<i>CoffeeScript</i>			----	----	----	Yes	----
<i>TypeScript</i>			----		----	Yes	----
<i>GWT</i>						Yes	----

# 大纲

1. Web的演化

2. 语言篇

a. 语言的发展

b. Dart 介绍

3. 技术篇 (Web Components)

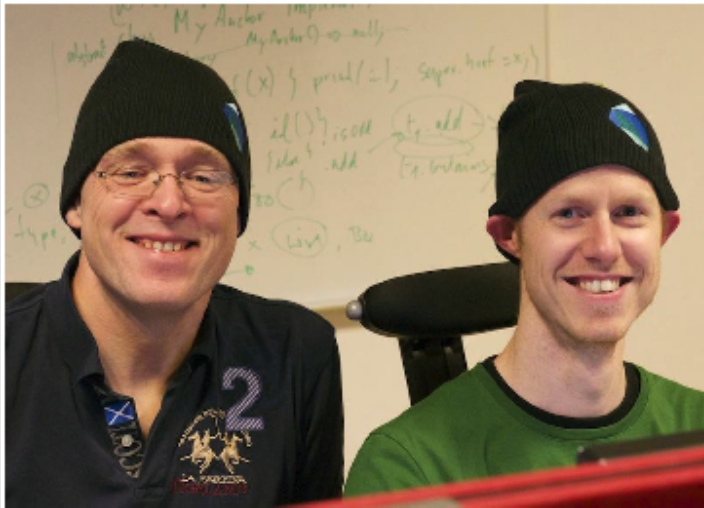
# Dart 语言的诞生

- 2011年10月对外宣布
- Chrome V8 团队打造
- 主要成员包括:Lars Bak、Kasper Lund、Gilad Bracha等
- 以大量编程语言实现的经验为基础





# Dart 并非颠覆, 而为实用



*Hacking on VMs side by side the last 13 years...*

Smalltalk  
Hotspot  
Crankshaft  
Beta CLDC JVM  
OOVM Dart V8  
Self

# Dart 代码

```
class Cookie {  
    var number_of_chips;  
    Cookie(num) {  
        number_of_chips = num;  
    }  
}
```

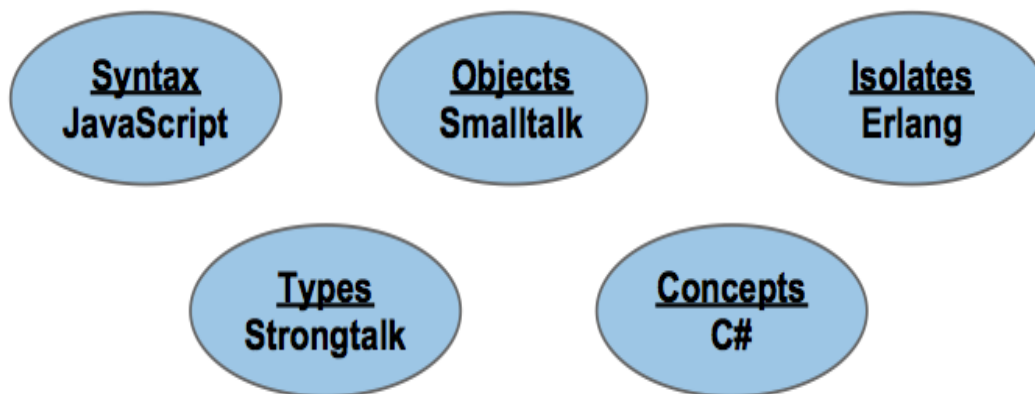
Dart 像 Java ?

有点像，但差太多

# Dart 是什么样语言

- 简单、熟悉的面向对象语言
- 基于类, 单继承、多实现
- 熟悉的语法和恰当的词法作用域
- 可选的静态类型
- 单线程和基于Isolate的并发

**Dart =**



# 为什么喜欢 Dart

1. 开源
2. 有 Google 当靠山
3. 执行效率高
4. 有机会成为 Android 内置的虚拟机，用来写手机 App
5. 可以用来写服务器后台程序
6. 可以写 Web 前端程序，在浏览器内运行

@蔡学镛



# Dart 语言的目标

- 结构化并且灵活的Web语言
- 简单、有生产力
- 适合从小型到大型的项目
- 高性能、快速启动
- 适合各种设备的Web环境



# Dart 语言的组成

- 语言规范
- Dart VM
- 丰富的类库
- 工具：
  - Dartium: Chrome + Dart VM
  - 包管理 Pub
  - Dart Editor
- Dart → JavaScript 编译器: dart2js



# Dart 两种运行模式

## 1) 检查模式 (checked)

检查类型匹配, 及早发现问题, 但性能差

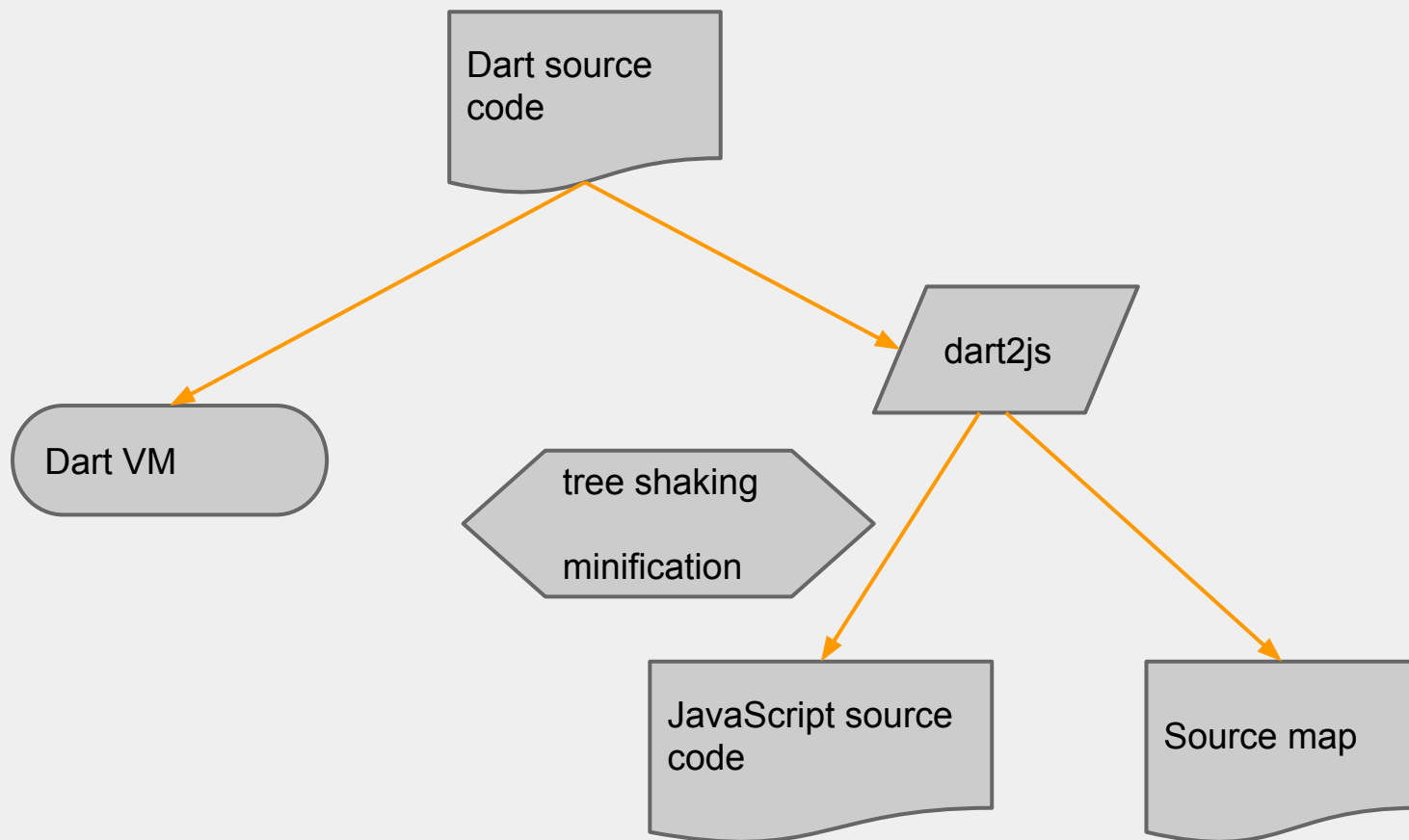
`T x = o` 等价于 `assert(o == null || o is T)`

## 2) 生产模式 (production)

不检查类型, 性能好

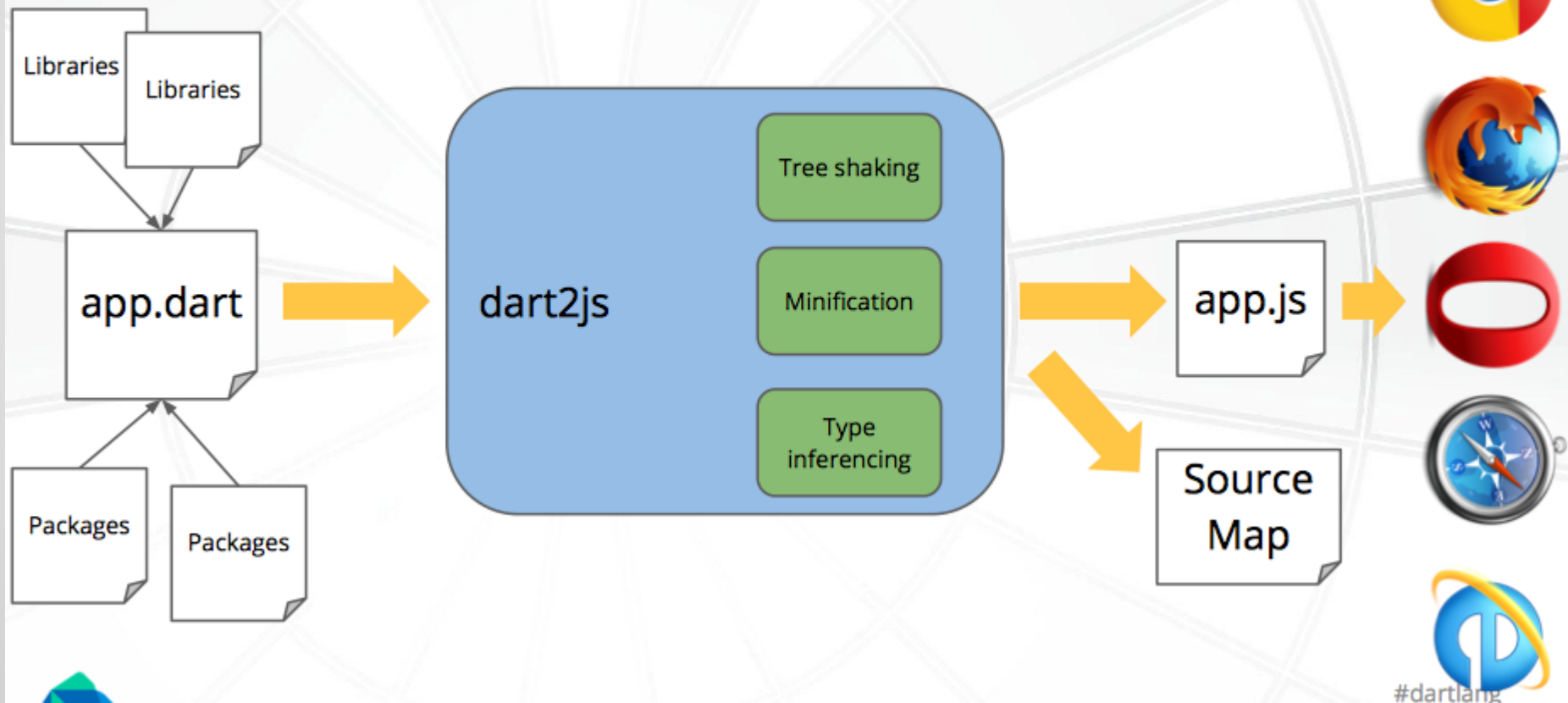


# Dart 运行环境

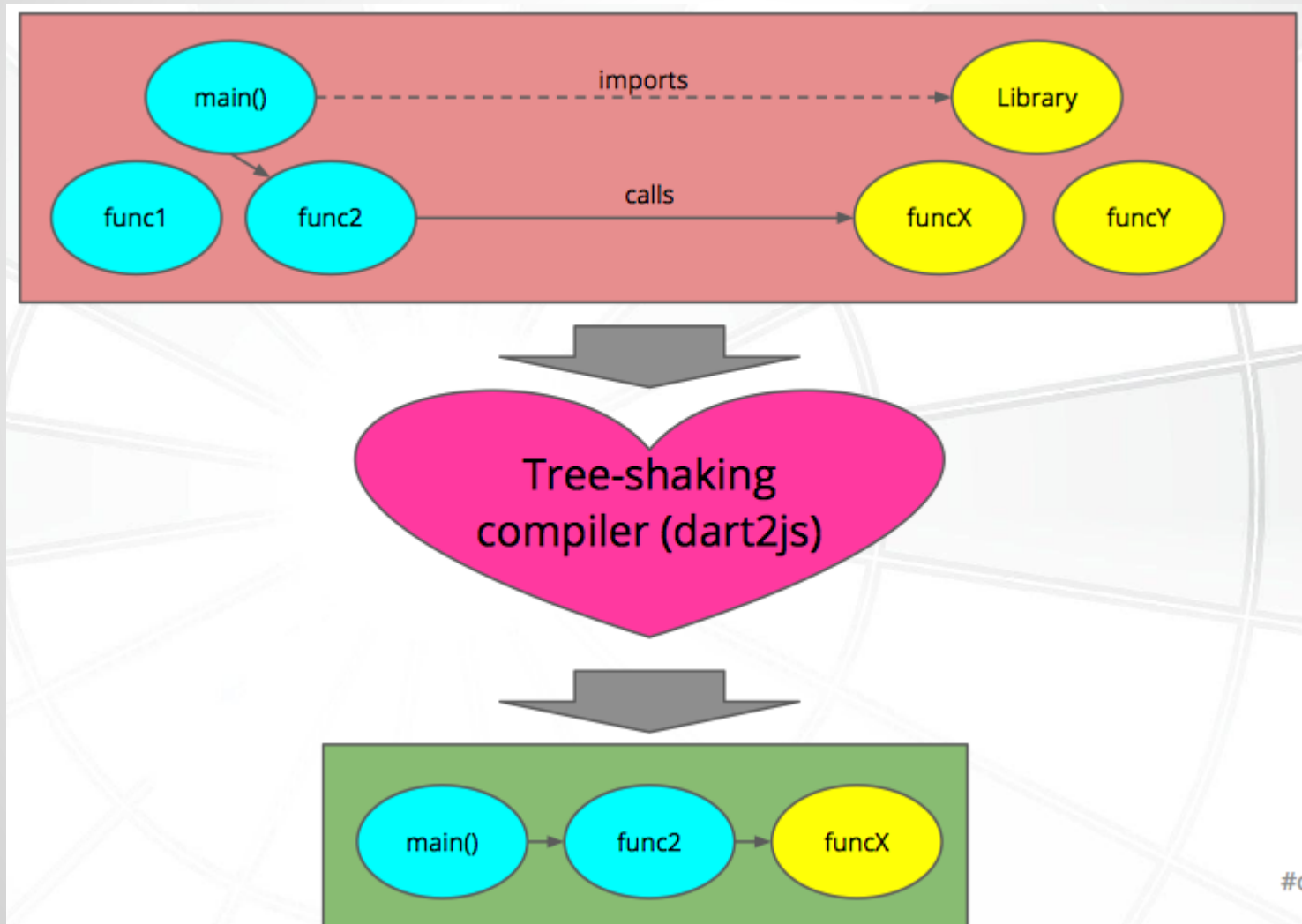


# Dart2js

## Generating smaller JavaScript



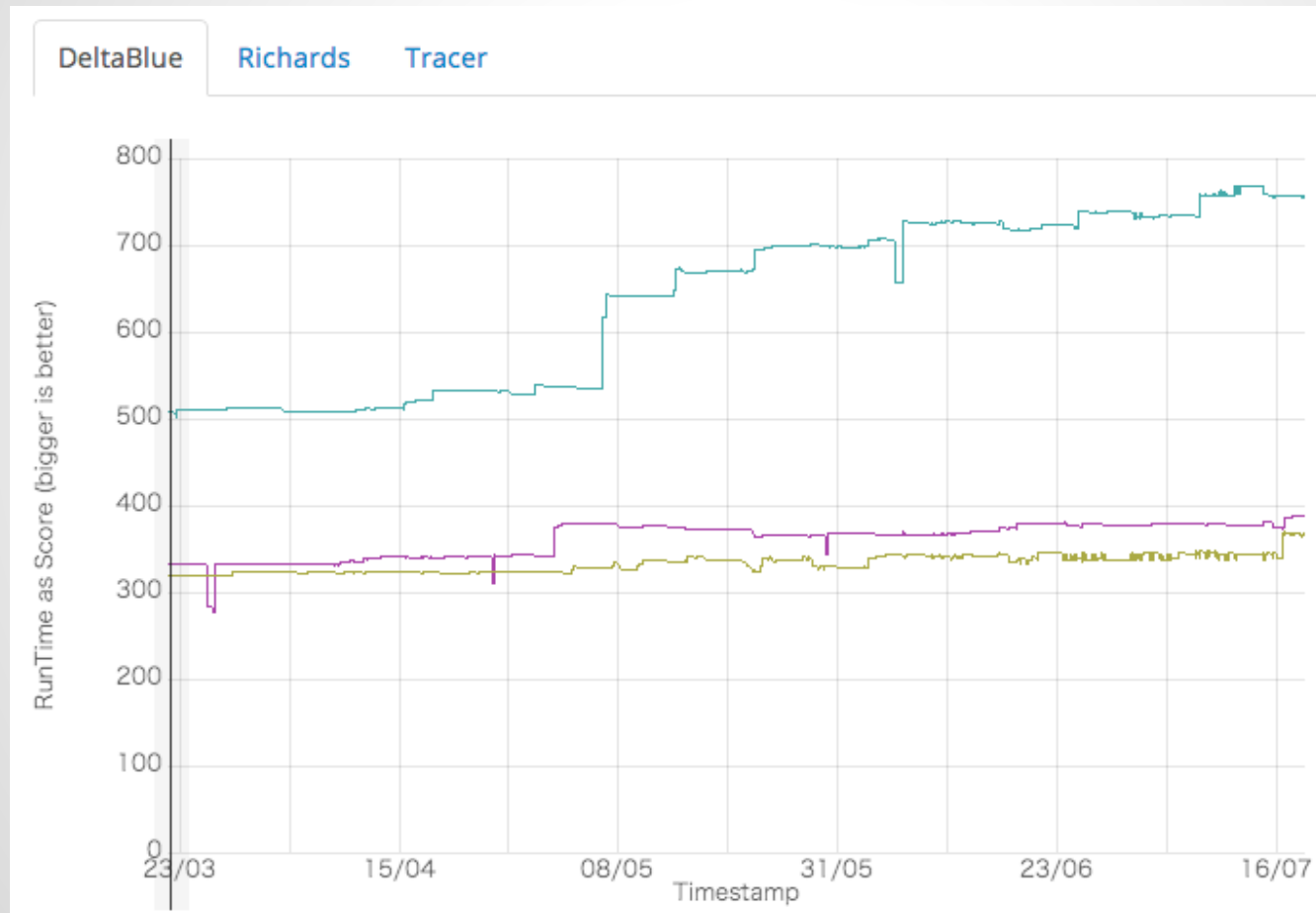
# Tree-shaking



# 高性能

- 语言的设计影响性能
- 使用snapshot启动速度快10倍
- 运行时不能改变对象结构有利于VM优化
- Dart VM 性能已超越 JavaScript V8
- .....

# 性能测试



<http://www.dartlang.org/performance/>

# Dart 语法特性

- 纯面向对象: 类
- 范型
- 函数、闭包
- 库: import
- 可选类型
- 词法作用域
- 异步与并发编程
- getter 和 setter
- 方法级联调用
- 操作符重载
- Markdown注释
- 支持 Mixin
- 基于Mirror的反射
- 不支持 eval
- .....

# 类型的好处

- 类型即文档，表明程序意图，提供概念框架
- 避免特定的变量命名模式或注释方式
- 适合范型
- 良好工具的支持
- 更早发现问题(类型检查):
  - 检查模式运行
  - 静态代码分析
- Dart 类型是可选的，而非强制的。

# 可选类型

```
class Point {  
    var x, y;  
    Point(this.x, this.y);  
    operator +(other) => new Point(x + other.  
x, y + other.y);  
    toString() => "($x,$y)";  
}
```

```
main() {  
    var p = new Point(2, 3);  
    print(p + new Point(4, 5));  
}
```



# 可选类型

```
class Point {  
    num x, y;  
    Point(this.x, this.y);  
    Point operator +(Point other) =>  
        new Point(x + other.x, y +  
other.y);  
    String toString() => "($x,$y)";  
}  
main() {  
    Point p = new Point(2, 3);  
    print(p + new Point(4, 5));  
}
```

# 方法级联

```
bg.style
  .position = 'absolute'
  .top = '0px'
  .left = '0px'
  .width = "${doc.offsetWidth}px"
  .height = "${doc.clientHeight}px"
  .backgroundColor = 'black'
  .opacity = '0.8'
  .zIndex = '1000';
```

# Dart 小结

- 提高生产力
- 简洁、熟悉的语法
- 工具支持好
- 性能好
- 项目可伸缩
- Web 和 Server 编程
- 库正在逐渐丰富

# Dart 资源

官方网站: <http://dartlang.org>

邮件列表、Google Plus、Stackoverflow

开源项目: <https://code.google.com/p/dart/>

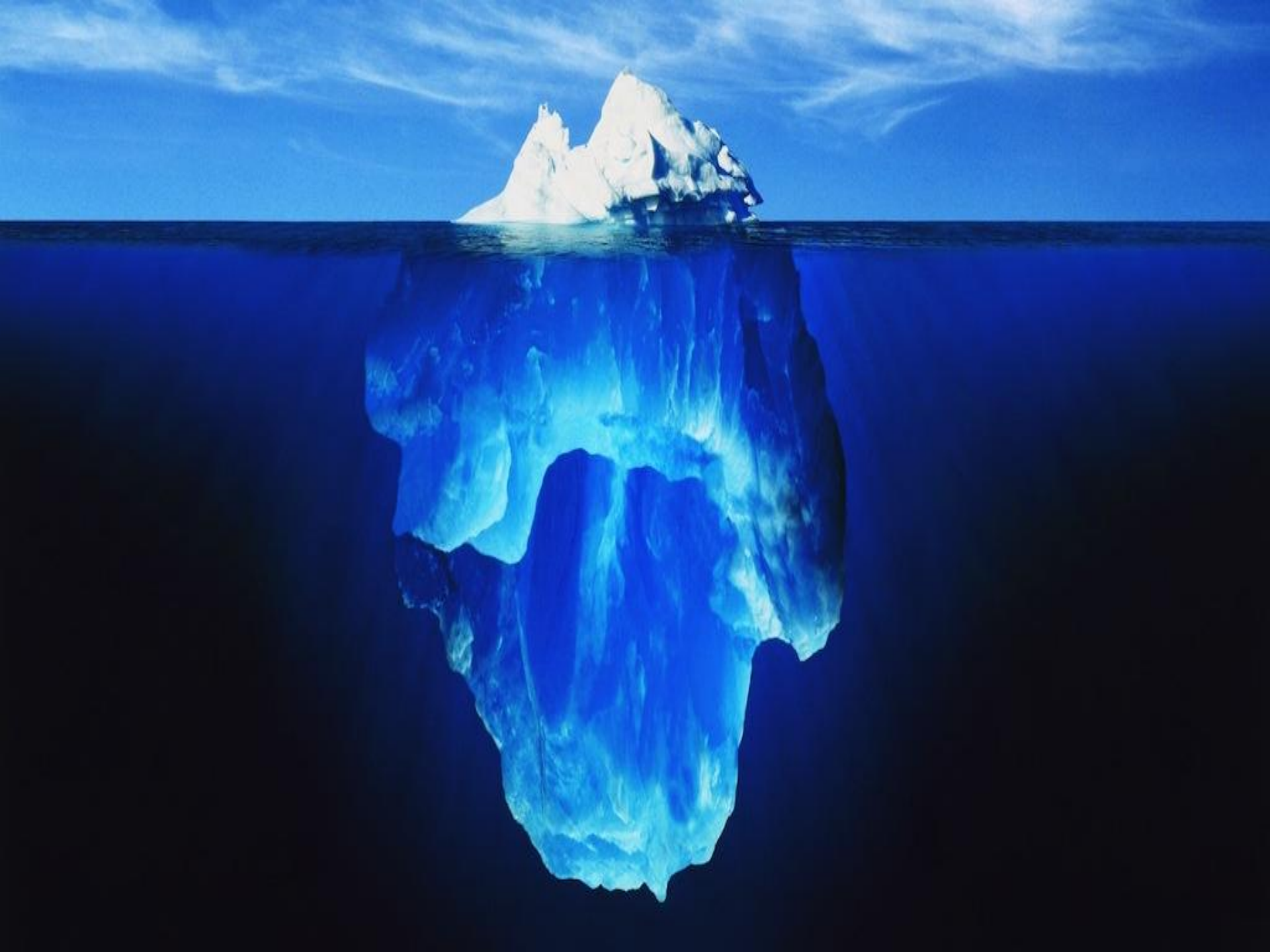
子项目: <https://github.com/dart-lang>

Pub: <http://pub.dartlang.org>

微博: @Dart 语言

# 编程语言小结

- JavaScript 继续改进, 仍是主流
- 语言的多元化
- 全新或兼容的新语言
- 更简洁 (CoffeeScript)
- 更平衡、实用 (Dart)



# 大纲

## 1. Web的演化

## 2. 语言篇

- a. 语言的发展

- b. Dart 介绍

## 3. 技术篇

- a. Web Components

- b. Polymer 库

# 最快速的开发方法

用别人做好的，再改改

前提：高质量、易调整

[北京GDG DevFest网站](#)

# 复 用

但是，很多因素影响软件的复用





# Web Components

- Templates
- Custom Elements
- Shadow DOM
- Imports

# Templates

```
<template id="mytemplate">  
  <img src="">  
  <div class="comment"></div>  
</template>
```

# Templates

- 脚手架
- 惰性的 DOM 块, 稍后需要时再激活
- 解析, 但不渲染
- `<script>` 不执行、样式和图片不加载、多媒体不播放
- 被 document 隐藏, 不能直接操作内部
  - eg: `document.querySelector('#mytemplate .comment') == null`

# Template 使用示例

```
<template>
  <span>Instance: <b>0</b></span>
  <script>alert('kthxbai!')</script>
</template>
<button onclick="useIt()">Stamp</button>
<script>
  function useIt() {
    var content = document.querySelector('template').
content; // 1. Get guts.

    var b = content.querySelector('b');
    b.textContent = parseInt(b.textContent) + 1; // 2.
Modify template's DOM at runtime.

    document.body.appendChild(content.cloneNode(true)); //
3. Clone to stamp it out.
  }
</script>
```

# Custom Elements

- 定义自己的标签(名字中要包含 - )
- 纽带, 封装其它元素和属性、行为、JS代码
- 创建新的 HTML 元素, 扩展 HTML 的语义
- 扩展现存的 DOM 对象
- 组件复用的单元
- 让标签更有意义

# 大量底层标签可读性差

+Eric Search Images Maps Play YouTube News Gmail Drive Calendar More ▾

Google

Eric Bidelman + Share

Gmail ▾

1-18 of 18

Primary Social Promotions + Eric ▾ New Hangout

Resources Network Sources Timeline Profiles Audits Console

```
<div class="ata-asE" style="display: none;"></div>
▼ <div tabindex="0" style="position: relative; min-height: 100%;">
  ▶ <div class="vI8oZc cS">...</div>
  ▼ <div class="nH" style="width: 1218px;">
    ▼ <div class="nH" style="position: relative;">
      ▶ <div class="nH w-asV aiw">...</div>
      ▼ <div class="nH">
        ▼ <div class="no">
          ▶ <div class="nH oy8Mbf nn aeN" style="width: 176px; height: 342px;">...</div>
          ▼ <div class="nH nn" style="width: 865px;">
            ▼ <div class="nH">
              ▼ <div class="nH">
                ▼ <div class="ar4 z">
                  ▶ <div id=":ro" class="aeH">...</div>
                  ▼ <div class="A0">
                    ▼ <div id=":rp" class="Tm aeJ" style="height: 336px;">
                      ▼ <div id=":rr" class="aeF" style="min-height: 180px;">
                        ▼ <div class="nH">
                          ▼ <div class="BlthKe nH oy8Mbf aE3" role="main" style>
                            <div style></div>
                            <div class="afn"></div>
                            ▼ <div class="aKh aPb">
```

► Computed Style ☐ Show inherited

▼ Styles

element.style {  
}

Matched CSS Rules

.Cp { ?ui=2&shva=1:0  
position: relative;  
}

div { user agent stylesheet  
display: block;  
}

Inherited from body.aAU.hasGoo...

body, td, input, ?ui=2&shva=1:0  
textarea, select {  
font-family: arial,sans-serif;  
}

body, td, ?ui=2&shva=1:27  
input, textarea, select {

... div div div div div div div #:rp #:rr div div div div div div div div div.Cp ✖ 180 ⚠ 63

# 自定义元素可读性好

```
<g-tabs selected="0">  
  <span>One</span>  
  <span>Two</span>  
  <span>Three</span>  
  <span>Four</span>  
</g-tabs>
```

# 定义 custom element

```
<element name="x-foo" constructor="XFoo">
<section>I'm an x-foo!</section>
<script>
  var section = this.querySelector('section');
  this.register({
    prototype: {
      createdCallback: function() {
        this.textContent = section.textContent;
// this == <x-foo>
      },
      foo: function() { alert('foo() called'); }
    }
  });
</script>
</element>
```



# 使用 custom element

使用起来和标准元素一样(在注册之后)。声明使用或通过JS创建。

*// in HTML*

```
<x-foo></x-foo>
```

*// in JavaScript*

```
var elem = document.createElement('x-foo');  
elem.addEventListener('click', function(e) {  
    e.target.foo();  
});
```

*// 或者定义了构造函数*

```
document.body.appendChild(new XFoo());
```

# Shadow DOM

- 粘合剂
- DOM 和 样式的封装, 划出边界范围
- 隔离, 防止冲突
- 浏览器内部已有的封装技术
  - 示例 `<input type="date">`
  - chrome 设置中开启 show shadow dom

# HTML Imports

## 打包、分发、共享、复用

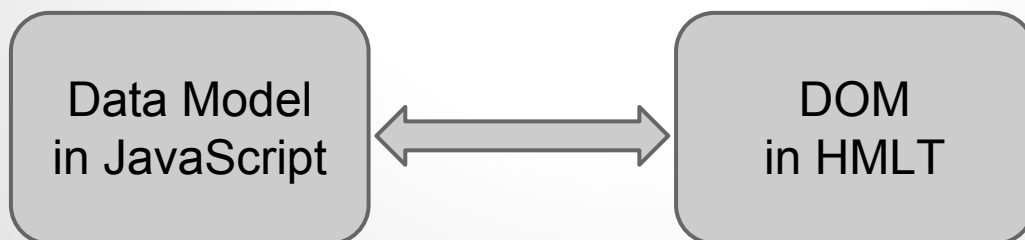
```
<!DOCTYPE html>
<html>
  <head>
    <link rel="import" href="x-foo.html">
  </head>
  <body>
    <x-foo></x-foo> <!-- Element definition
is in x-foo.html -->
  </body>
</html>
```

# 数据绑定

Web中最常见的任务：

1. 根据数据模型的变化修改DOM
2. 根据用户输入的变化修改数据模型

数据绑定就是自动化实现这种单向绑定或双向绑定



# 组件是新技术？

当然不是！思想和各种实现早就有，Web中也有，但 Web 缺少对组件技术的基础支撑。——我

- 组件技术适合于现代的软件环境
  - 软件更新快、规模大、强调协作等
  - 组件的概念、软件IC
- 组件的可独立发布性要求“接口与实现分离”
- 组件infrastructure,组件体系
  - COM、CORBA、J2EE/EJB
- 中间件(middleware)市场

——潘爱民, 2002.9

# Web Components 当前实现状态

	HTML Imports	Custom Elements	Template Element	Shadow DOM
Chrome	in flags(30)	in flags(27)	已支持	已支持
Firefox	将会支持	开发中	已支持	开发中
Safari	未知	将会支持	将会支持	将会支持
IE	未知	未知	将会支持	将会支持

2013/08 的情况

# Web Components QA

1. 还没实现的技术，以后再说吧。

现在就可以用！**Polymer or Polyfill**


2. 应用前景：

a) 直接用。

b) 可能更多的时候是作为基础技术，被其它 Web 框架或库所使用。但需要你理解这些基础概念。

# Chromium Dashboard

<http://www.chromestatus.com/features>



## Chromium Dashboard

what we're up to

Web Platform Features (5)

Custom

No active development

Proposed

In development

31 canary/dev

**30 beta**

29 stable

28

27

26

### Custom Elements

Web Components

Method for registering (creating) custom elements in script.

#### Implementation Status

**Behind a flag** 27   **Prefixed** No   [Launch bug](#)   **Owner(s)** dominicc@chromium.org

#### Consensus & Standardization

Firefox IE Safari Web developers

Working draft or equivalent   Browser share %



# Web Components 资源

<http://ebidel.github.io/webcomponents/>

# 大纲

## 1. Web的演化

## 2. 语言篇

- a. 语言的发展

- b. Dart 介绍

## 3. 技术篇

- a. Web Components

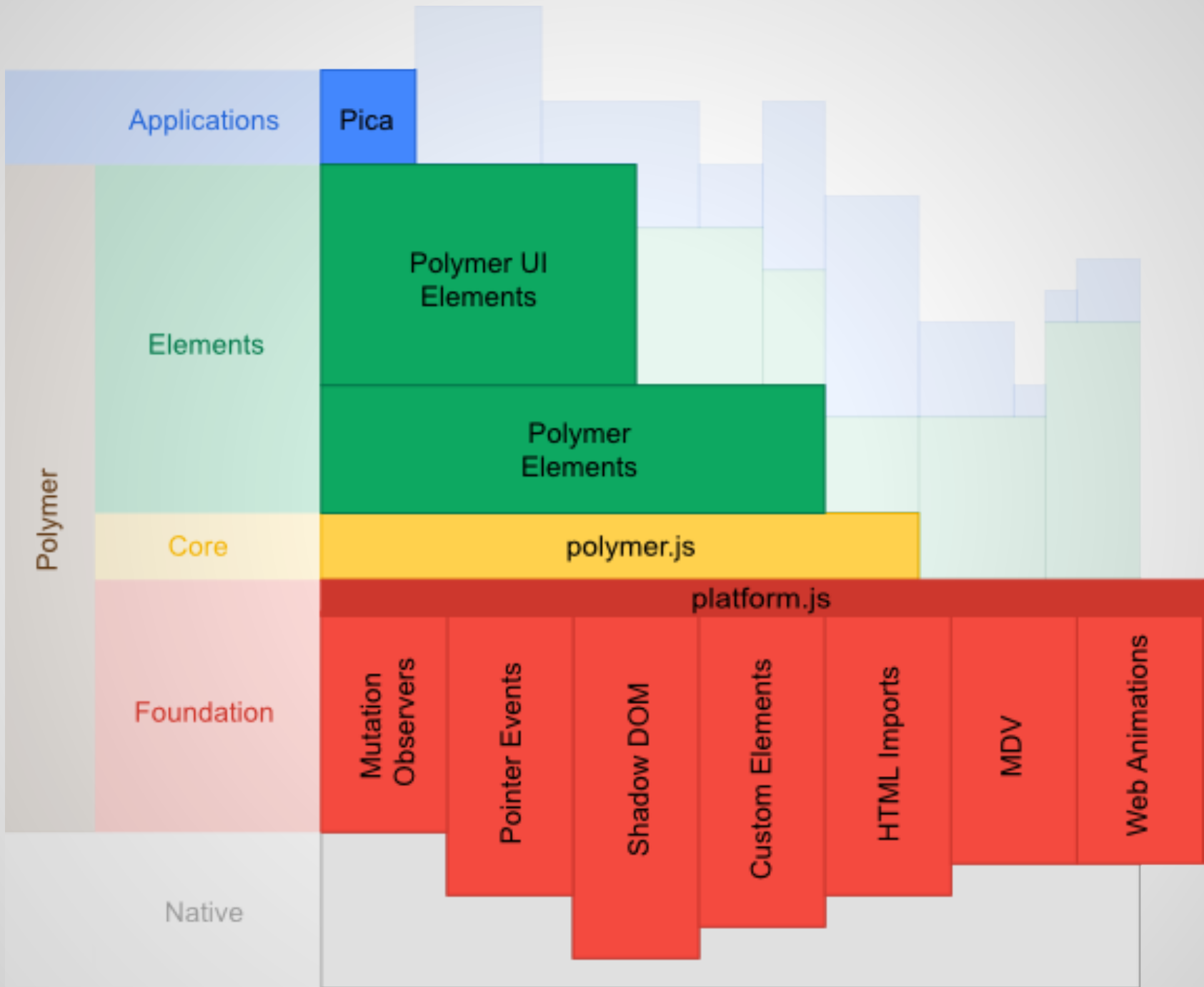
- b. Polymer 库

# Polymer 库

Polymer 是一个全新的基于 Web Components 基础之上的 Web 库, 为 Web 平台上浏览器的演进提供支持。

- 包含一组 polyfill 的库
- 以 Web Components 为核心
- 使用更方便的 API
- 丰富的UI组件(开发中)

<http://www.polymer-project.org/>



	Polyfill							
	Chrome Android	Chrome	Canary	Firefox	IE 10+	Safari 6+	Mobile Safari	
MutationObserver					[1]			
HTML Imports								
Custom Elements					[1]			
Shadow DOM								
MDV								
Pointer Events					[2]			
Pointer Gestures								
Web Animations								
Platform								
Toolkit								
	Native implementation							
	Chrome Android	Chrome	Canary	Firefox	IE 10+	Safari 6+	Mobile Safari	
MutationObserver								
HTML Imports								
Custom Elements								
Shadow DOM			25+ [3]					
MDV								
Pointer Events					[4]			
Pointer Gestures								
Web Animations								
Platform								
Toolkit								

# 什么是 Polyfill (腻子)

HTML5的很多功能并不是所有浏览器都支持。为了解决跨浏览器兼容性问题, Remy Sharp首次提到了polyfilling的概念。把浏览器想象成有裂缝的墙面, 而用腻子可以把这些裂缝填平, 最后得到的是光滑的浏览器“墙面”。

定义: 一段代码或插件, 可以让开发人员使用应有的技术, 就像浏览器原生提供该功能一样。换句话说, 它能帮你抹平API之墙。

详见: <http://www.ituring.com.cn/article/766> by 李松峰

# 两种 Web 技术风格

## 1. JavaScript 为主，标签为辅

大量使用 JavaScript 界面元素大都由 JavaScript 生成，如 ExtJS/Sencha。

优点：高度可控

## 2. 标签为主，JavaScript 为辅

和通用编程语言不同，Tag 是一种声明式语言，如 HTML、XML、Ant 等。

优点：简单清晰，更适合表达意图。

# Polymer 的理念

- 拥抱 HTML/DOM
  - custom element 是核心
  - 所有东西都是 custom element
- 随着 Web 平台的演进而演进, polyfill 也会随着平台的实现而逐渐废弃
- 最小化乏味的样板代码
- 对 Web 标准进行反馈循环



# Polymer Sandbox 演示

<http://www.polymer-project.org/tools/sandbox/>

1. Tabs & Menu bind
2. Publish new element
3. Input & Youtube bind
4. Sandbox App 无限递归

# Polymer 元素的使用

```
<polymer-element name="my-input"  
  constructor="MyInput" noscript>  
  <template>  
    ...  
  </template>  
</polymer-element>
```

使用时:

```
<script src="polymer.min.js"></script>  
<my-input></my-input>
```

Q & A

*Thank you!*

非前端工程师  
韩国恺

[hanguokai@gmail.com](mailto:hanguokai@gmail.com)

@Dart语言 / @hanguokai