



# Developing Dart Editor Eclipse Day at Googleplex 2013

Dan Rubel  
Eric Clayberg



# Overview

- Dart Project
- Dart Language
- Developing in Dart
- Developing Dart Editor
- Questions



<http://www.dartlang.org>

# Web Apps

- The web is everywhere
- Developing small apps is easy
- No installation required
- Supports incremental development
- Open platform

*... but we need to innovate*

- Productivity
- Scalability
- Speed

# Why?

What is wrong with JavaScript?

- Lack of structure
- Very difficult to find dependencies
- All numbers are floats
- Monkey patching
- Keep on truckin' mentality
- Libraries are files you concatenate

**... which makes it ...**

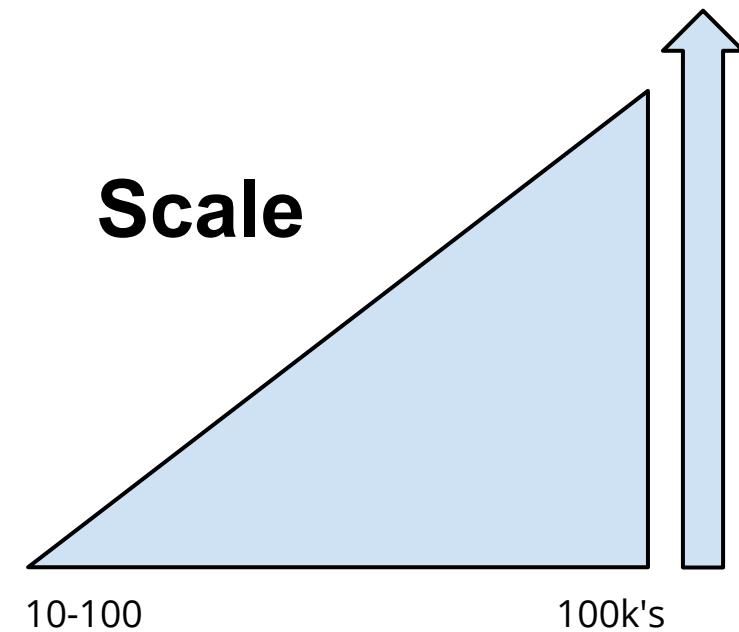
- Difficult to optimize performance
- Difficult to maintain large code bases
- Difficult to assert correctness

# Goal

“Dart helps developers from all platforms build complex, high performance client apps for the modern web.”

**VM Speed**

x2 → x?



# What is Dart?

## Dart is open source

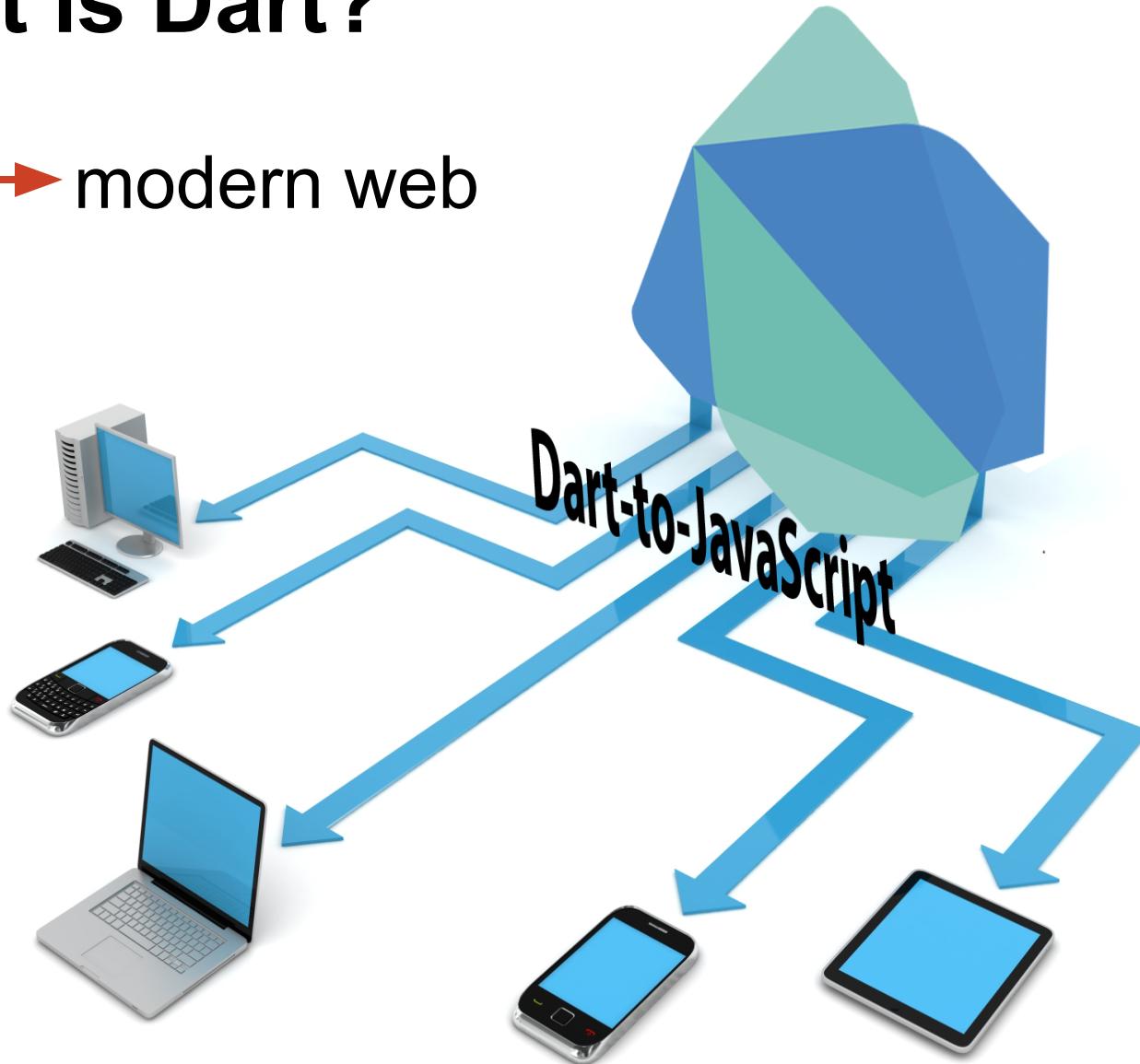
- BSD-style license
- [dart.googlecode.com](http://dart.googlecode.com)
- GitHub mirror
- Contributing guide
-  ECMA Standard (TC52)
-  Production ready (1.0)



open source  
initiative

# What is Dart?

Dart → modern web



# What is Dart?

- Rich client apps
- Server apps
- Offline-capable
- 60fps
- ECMAScript 5+
- HTML5



<http://www.dartlang.org>

# What is Dart?

Language

Libraries ( core, math, html, isolate, io, json,  
uri, utf, crypto, polymer,  
angular... )

Editor

Virtual machine

Compiler to JavaScript

Browser integration

Package manager

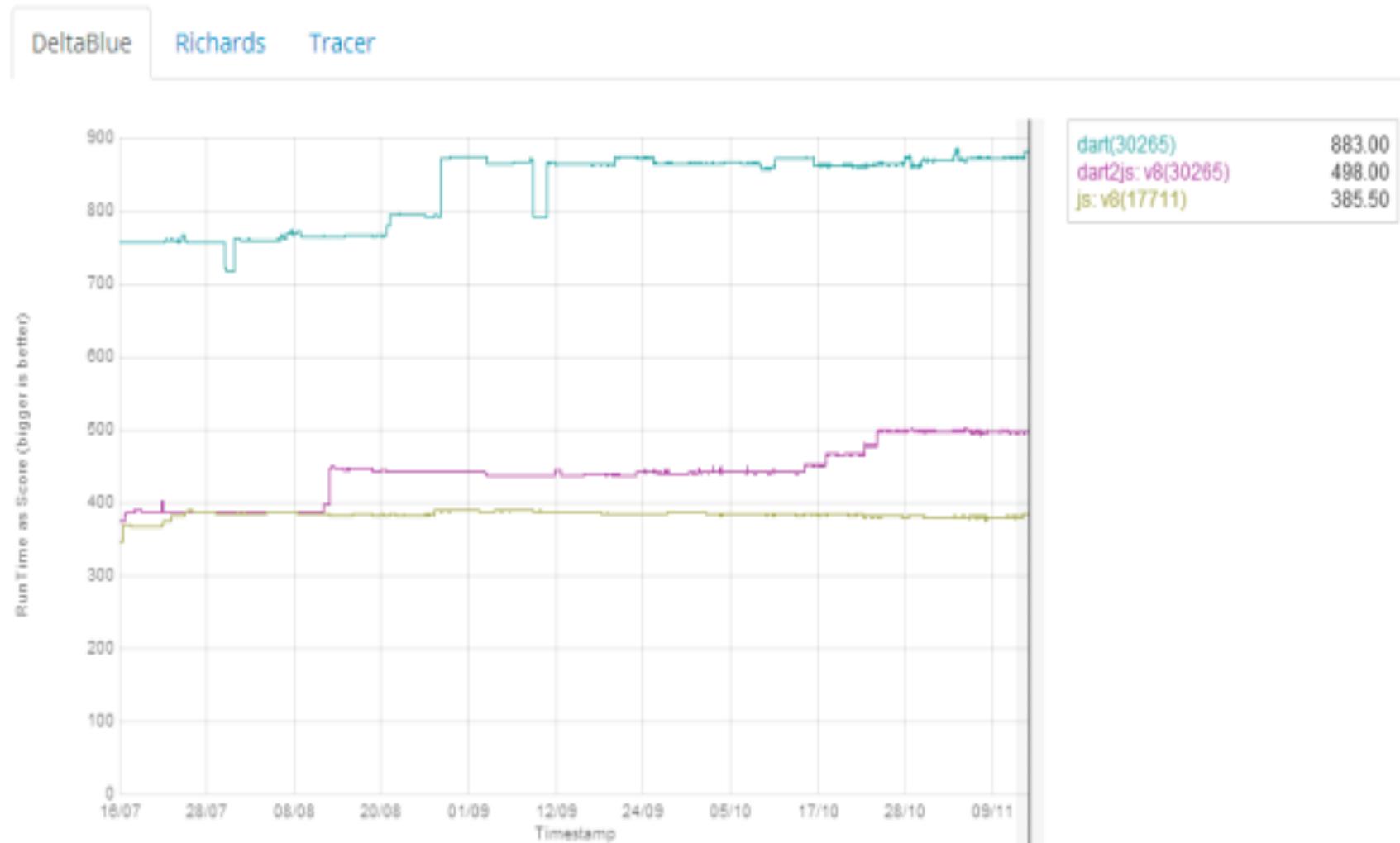
Web Components (Polymer & Angular)

Community

"Batteries Included"



# Performance - x2 now

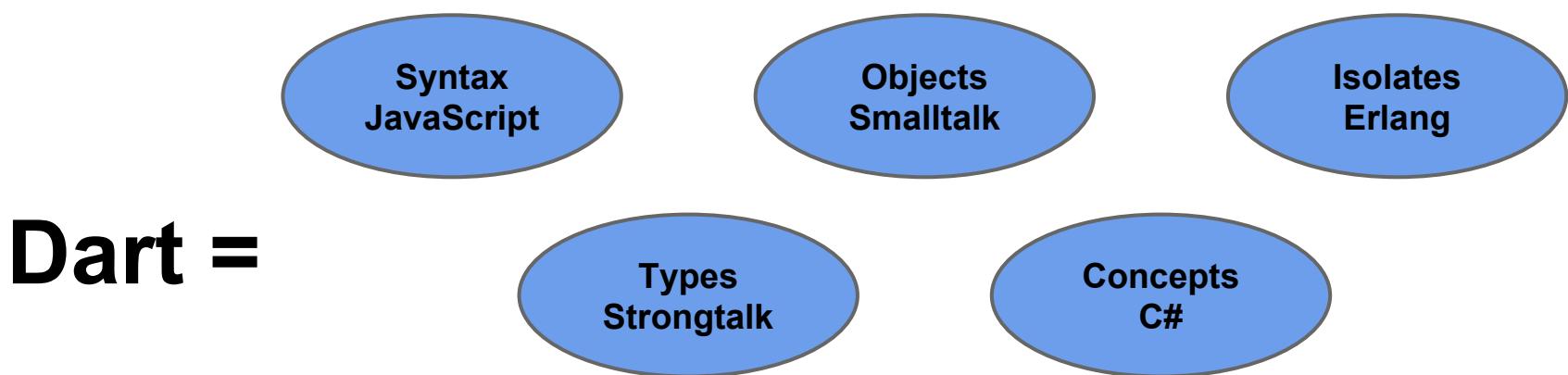




<http://www.dartlang.org>

# Language Goals

- Unsurprisingly simple and familiar
- Class-based single inheritance (w/ mixins)
- Proper lexical scoping
- Single-threaded with isolates
- Optional static types



# Optional Static Types

Static types ...

- Communicate developer intent
- Used by tools to assert program correctness
- Not used at runtime unless specified

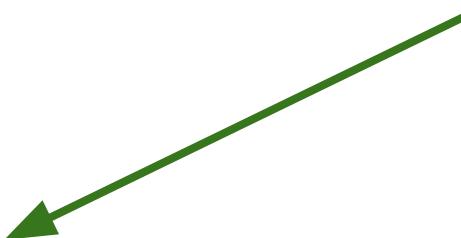
You can mix typed and untyped code ...

```
var p = new Point(2, 3);  
int x = 4;  
  
print (p + new Point(x, 5));
```

# Implicit Interfaces

```
class Printer {  
  void print(String message) {  
    // print the message  
  }  
}
```

Mock for testing



```
class MockPrinter implements Printer {  
  void print(String message) {  
    // validate method was called  
  }  
}
```



# Mixins



```
class PrettyPrinter extends ASTTool with Counter
```

```
{
```

```
...
```

```
}
```

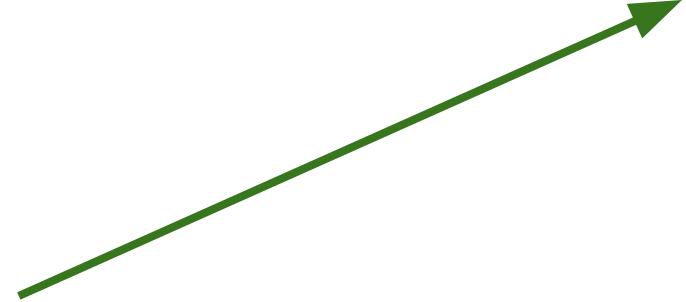
```
class Counter {
```

```
    int count = 0;
```

```
    int get increment => ++count;
```

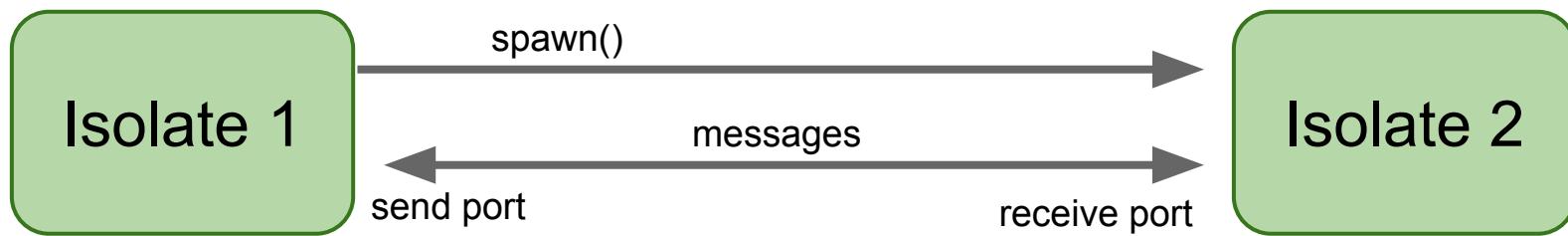
```
    int get decrement => --count;
```

```
}
```



# Isolates

- Inspired by Erlang
- Lightweight units of execution
  - Each isolate conceptually a process
  - Nothing shared
  - All communication via message passing
- Support concurrent execution



# Futures

Use Futures to perform asynchronous operations.

```
import 'dart:io';
void main() {
  new File('some.txt').readAsString().then((content) => print(content));
  print('main complete');
}
```



returns a Future



Executed when contents are read

Output:

```
main complete
contents of file
```



# Developing in DART

<http://www.dartlang.org>



# Hello World

```
import 'dart:html';

void main() {
  new Hello().welcome("World");
}

class Hello {
  void welcome(String name) {
    var message = "Hello $name";
    document.querySelector('#status')
      ..text = message
      ..onClick.listen(handler);
  }
}
```

# Hello World

```
import 'dart:html';
```

```
void main() {  
    new Hello().welcome("World");  
}
```

```
class Hello {  
    void welcome(String name) {  
        var message = "Hello $name";  
        document.querySelector('#status')  
            ..text = message  
            ..onClick.listen(handler);  
    }  
}
```

Libraries



# Hello World

```
import 'dart:html';

void main() { ← Functions
  new Hello().welcome("World");
}
```

```
class Hello {
  void welcome(String name) {
    var message = "Hello $name";
    document.querySelector('#status')
      ..text = message
      ..onClick.listen(handler);
  }
}
```

# Hello World

```
import 'dart:html';
```

```
void main() {  
    new Hello().welcome("World");  
}
```

```
class Hello {  
    void welcome(String name) {  
        var message = "Hello $name";  
        document.querySelector('#status')  
            ..text = message  
            ..onClick.listen(handler);  
    }  
}
```

Classes



# Hello World

```
import 'dart:html';

void main() {
  new Hello().welcome("World");
}

class Hello {
  void welcome(String name) {
    var message = "Hello $name";
    document.querySelector('#status')
      ..text = message
      ..onClick.listen(handler);
  }
}
```

## Methods



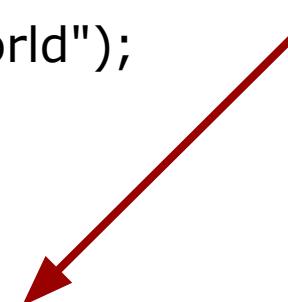
# Hello World

```
import 'dart:html';

void main() {
  new Hello().welcome("World");
}
```

```
class Hello {
  void welcome(String name) {
    var message = "Hello $name";
    document.querySelector('#status')
      ..text = message
      ..onClick.listen(handler);
  }
}
```

Optional Parameters



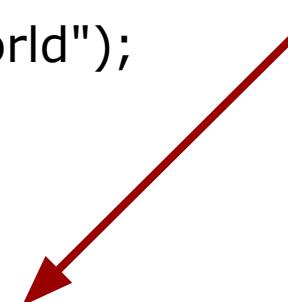
# Hello World

```
import 'dart:html';
```

```
void main() {  
  new Hello().welcome("World");  
}
```

```
class Hello {  
  void welcome([String name = "Bob"]) {  
    var message = "Hello $name";  
    document.querySelector('#status')  
      ..text = message  
      ..onClick.listen(handler);  
  }  
}
```

Optional Parameters



# Hello World

```
import 'dart:html';

void main() {
  new Hello().welcome("World");
}

class Hello {
  void welcome([String name = "Bob"]) {
    var message = "Hello $name";
    document.querySelector('#status')
      ..text = message
      ..onClick.listen(handler);
  }
}
```

## Optional Types



# Hello World

```
import 'dart:html';

void main() {
  new Hello().welcome("World");
}

class Hello {
  void welcome([String name = "Bob"]) {
    String message = "Hello $name";
    document.querySelector('#status')
      ..text = message
      ..onClick.listen(handler);
  }
}
```

## Optional Types



# Hello World

```
import 'dart:html';

void main() {
  new Hello().welcome("World");
}

class Hello {
  void welcome([String name = "Bob"]) {
    String message = "Hello $name";
    document.querySelector('#status')
      ..text = message
      ..onClick.listen(handler);
  }
}
```

String Interpolation



# Hello World

```
import 'dart:html';

void main() {
  new Hello().welcome("World");
}
```

```
class Hello {
  void welcome([String name = "Bob"]) {
    String message = "Hello $name";
    document.querySelector('#status')
      ..text = message
      ..onClick.listen(handler);
  }
}
```

## Cascades

# Hello World

```
import 'dart:html';

void main() {
  new Hello().welcome("World");
}
```

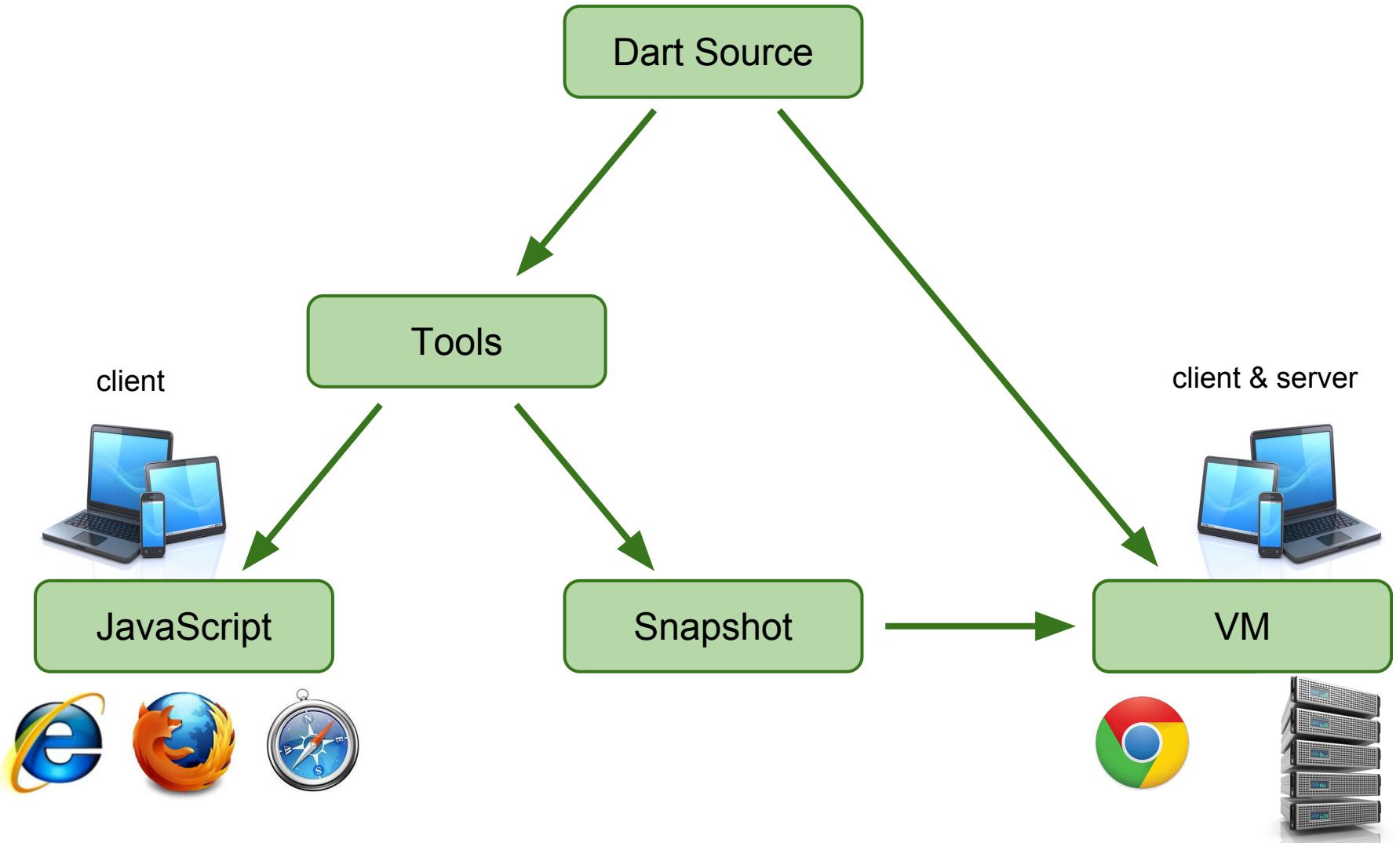
```
class Hello {
  void welcome([String name = "Bob"]) {
    String message = "Hello $name";
    document.querySelector('#status')
      ..text = message
      ..onClick.listen(handler);
  }
}
```

## Simple DOM API

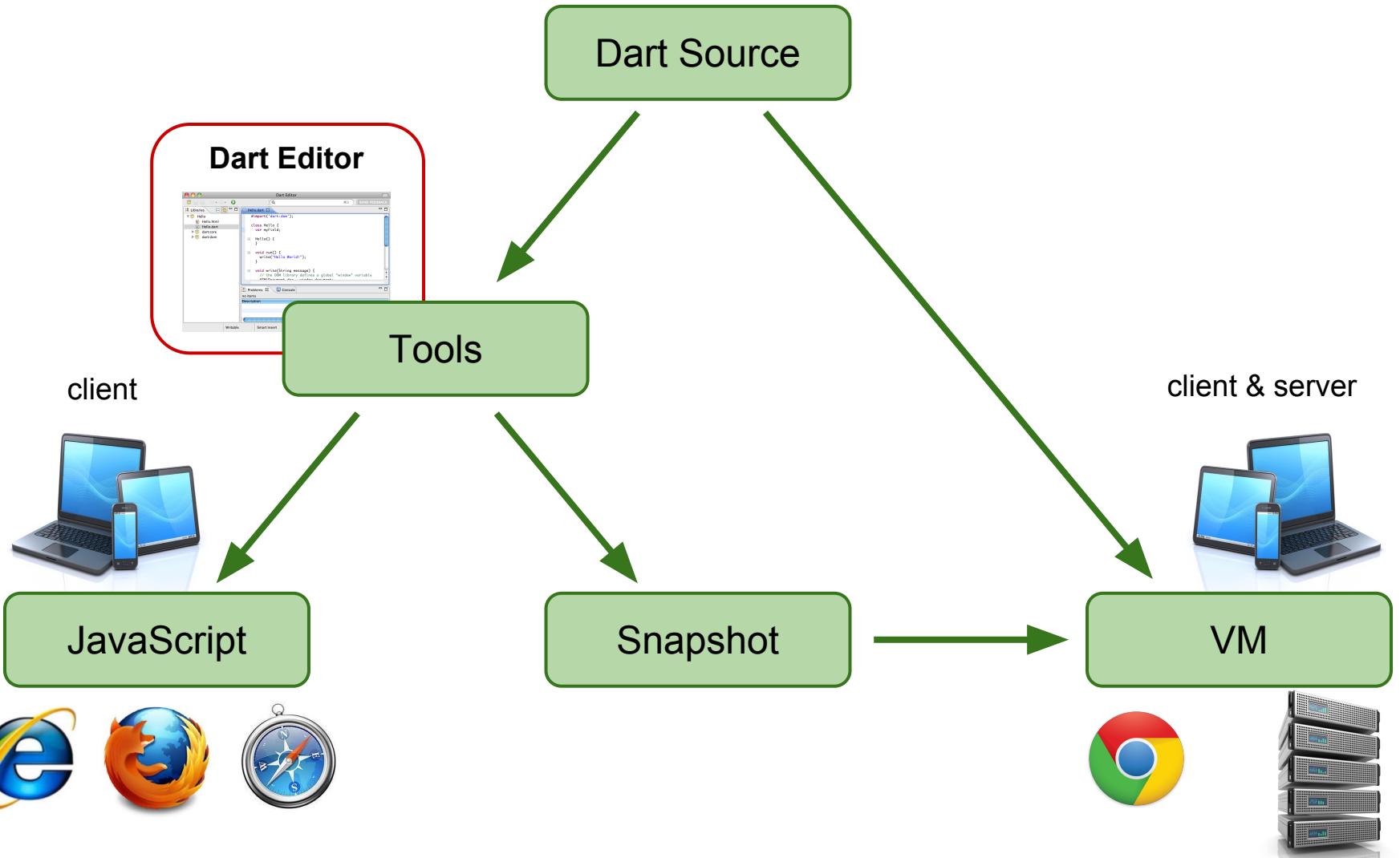
# Pub

- Package manager for Dart
- Helps with Dart code reuse and bundling
- Handles versioning and dependency management
- Available at <http://pub.dartlang.org/>

# Tools

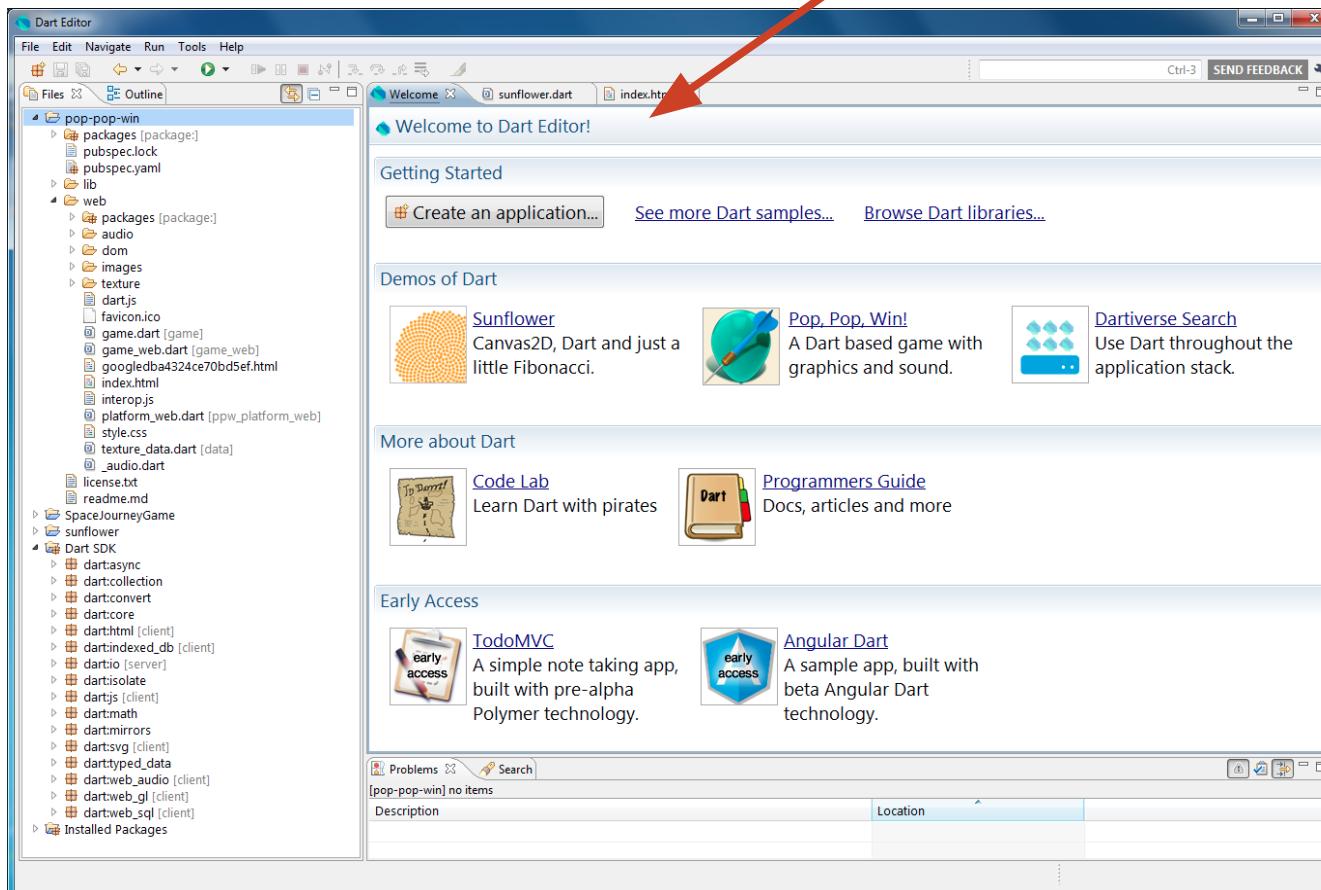


# Tools



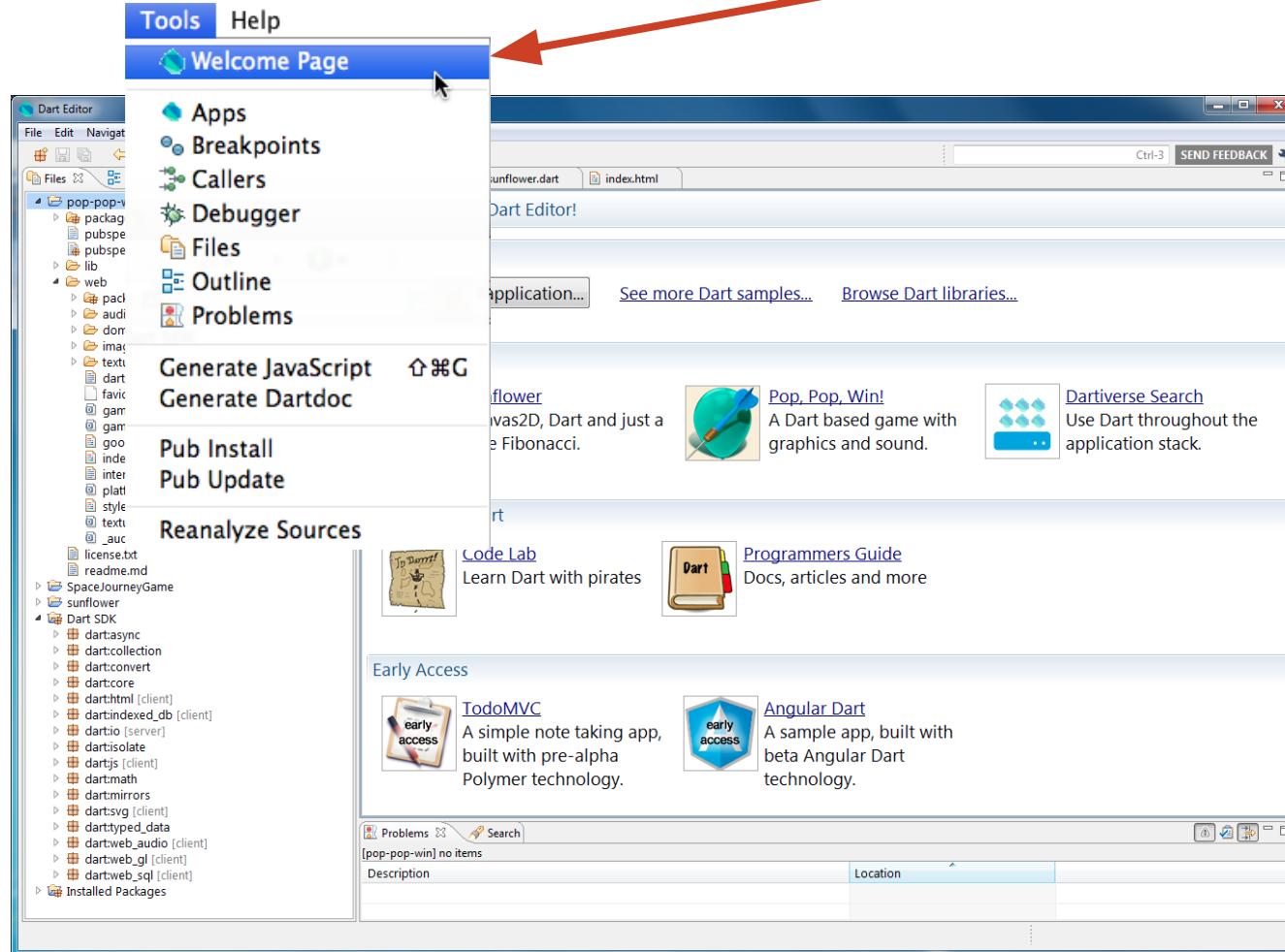
# Dart Editor

## Welcome Page



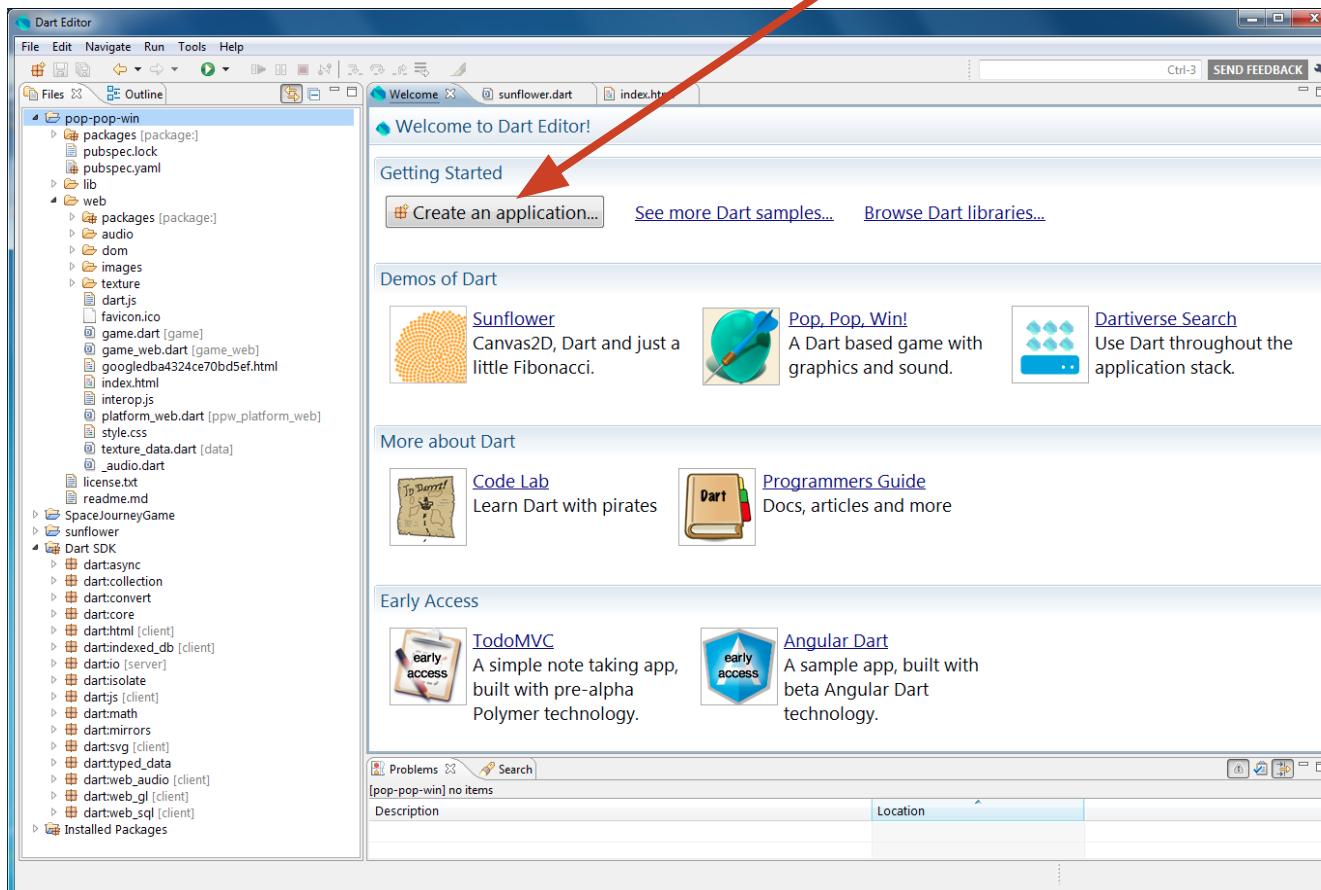
# Dart Editor

## Welcome Page



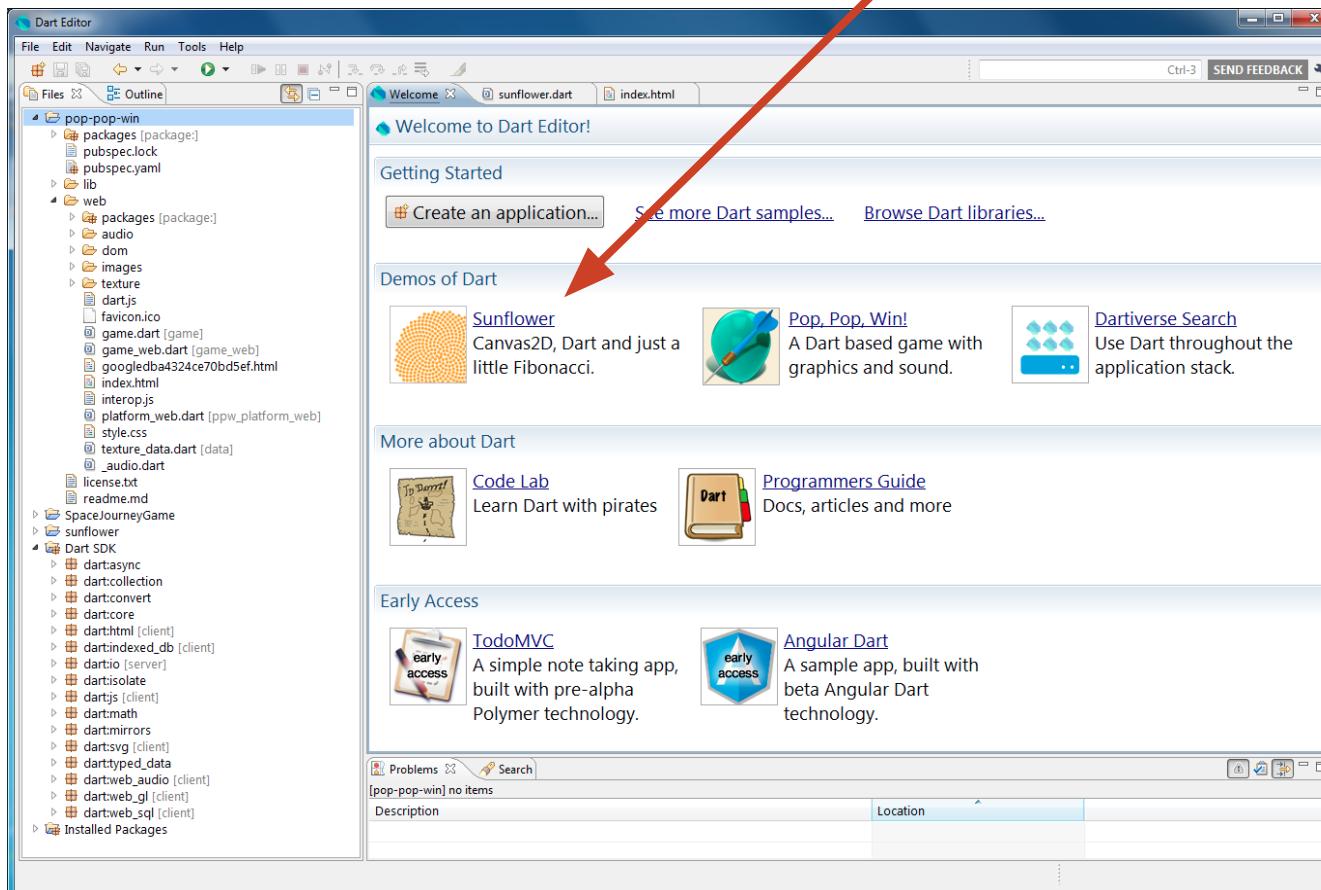
# Dart Editor

## Create application



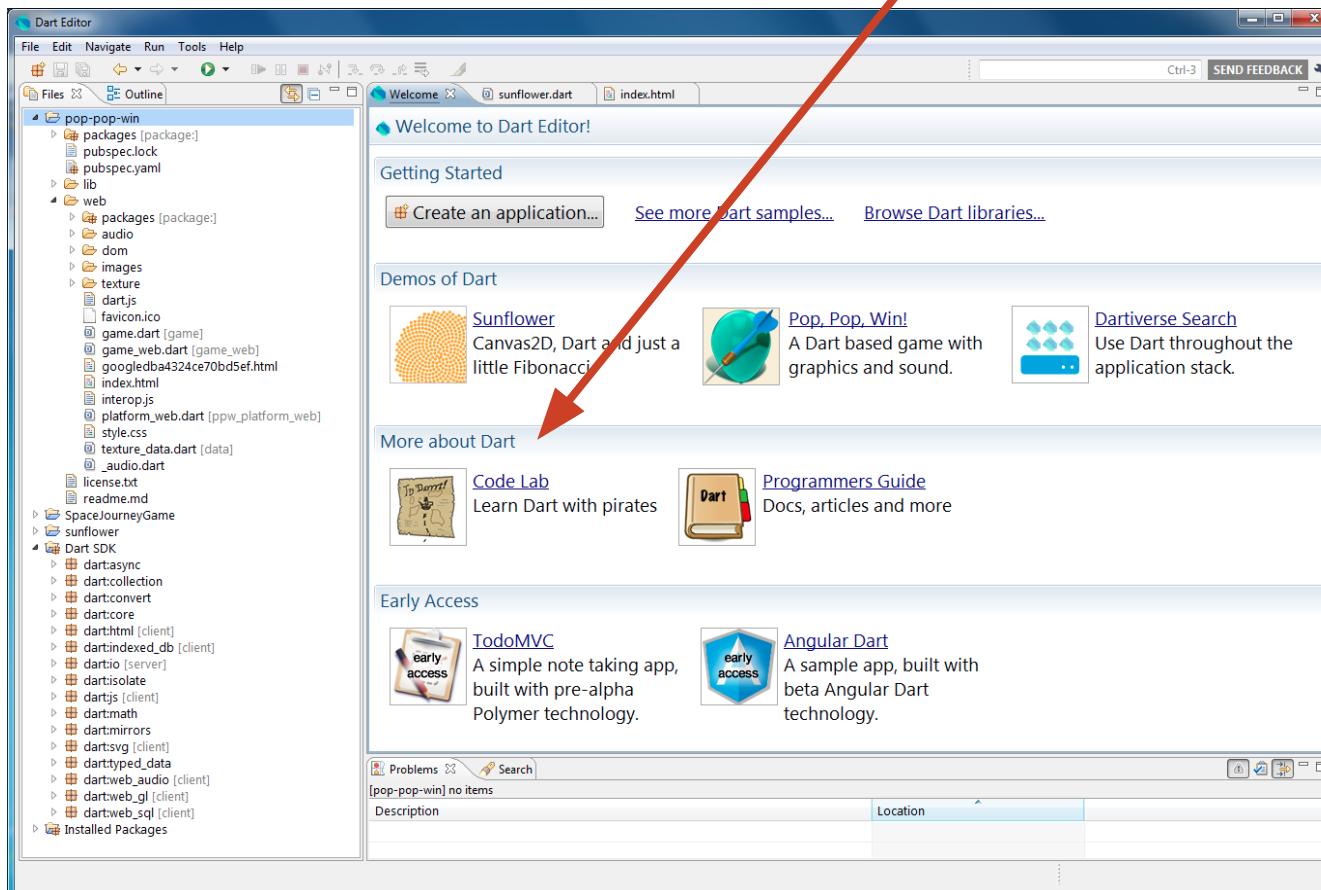
# Dart Editor

## Load Sample Code



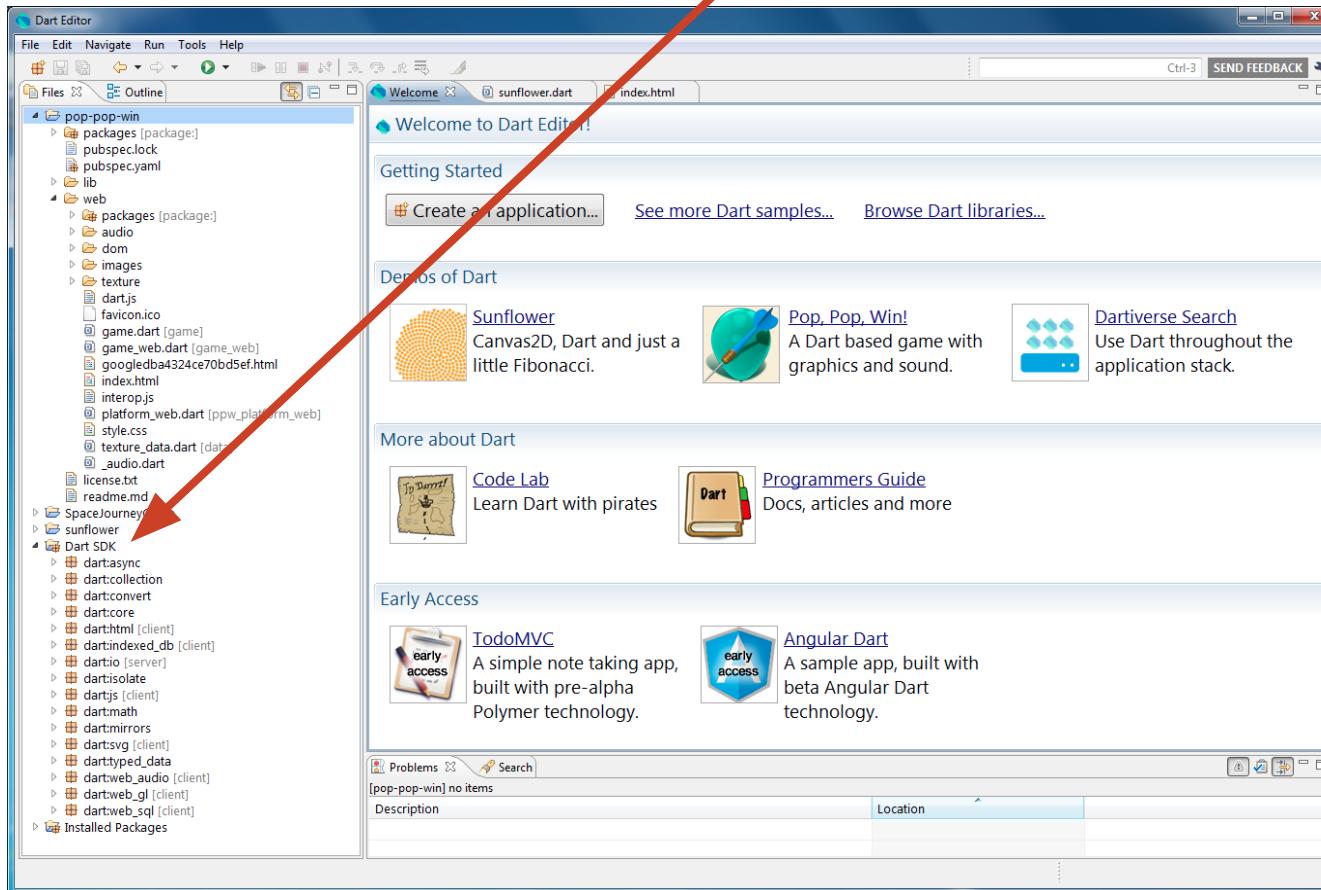
# Dart Editor

## More Information



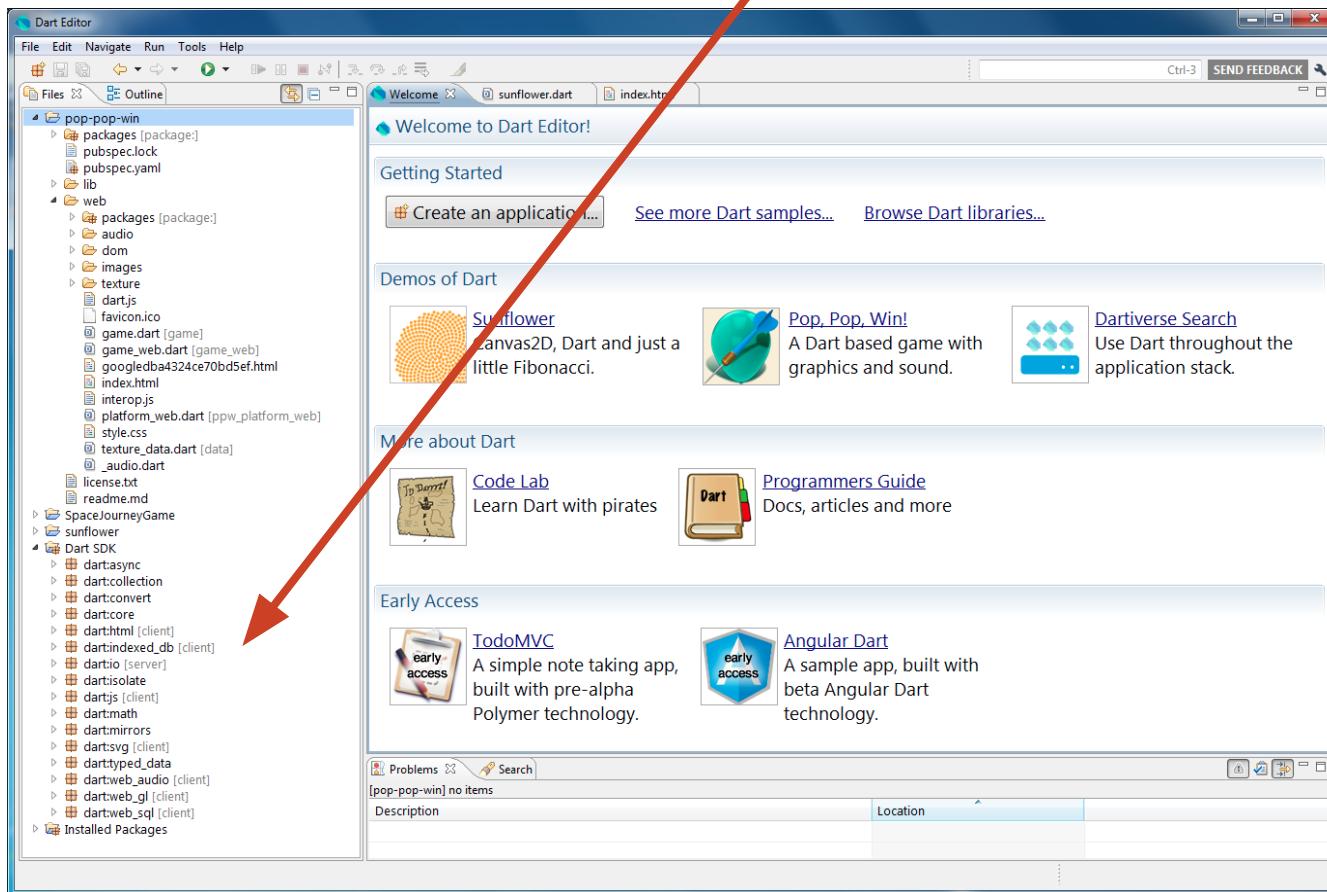
# Dart Editor

## Dart SDK



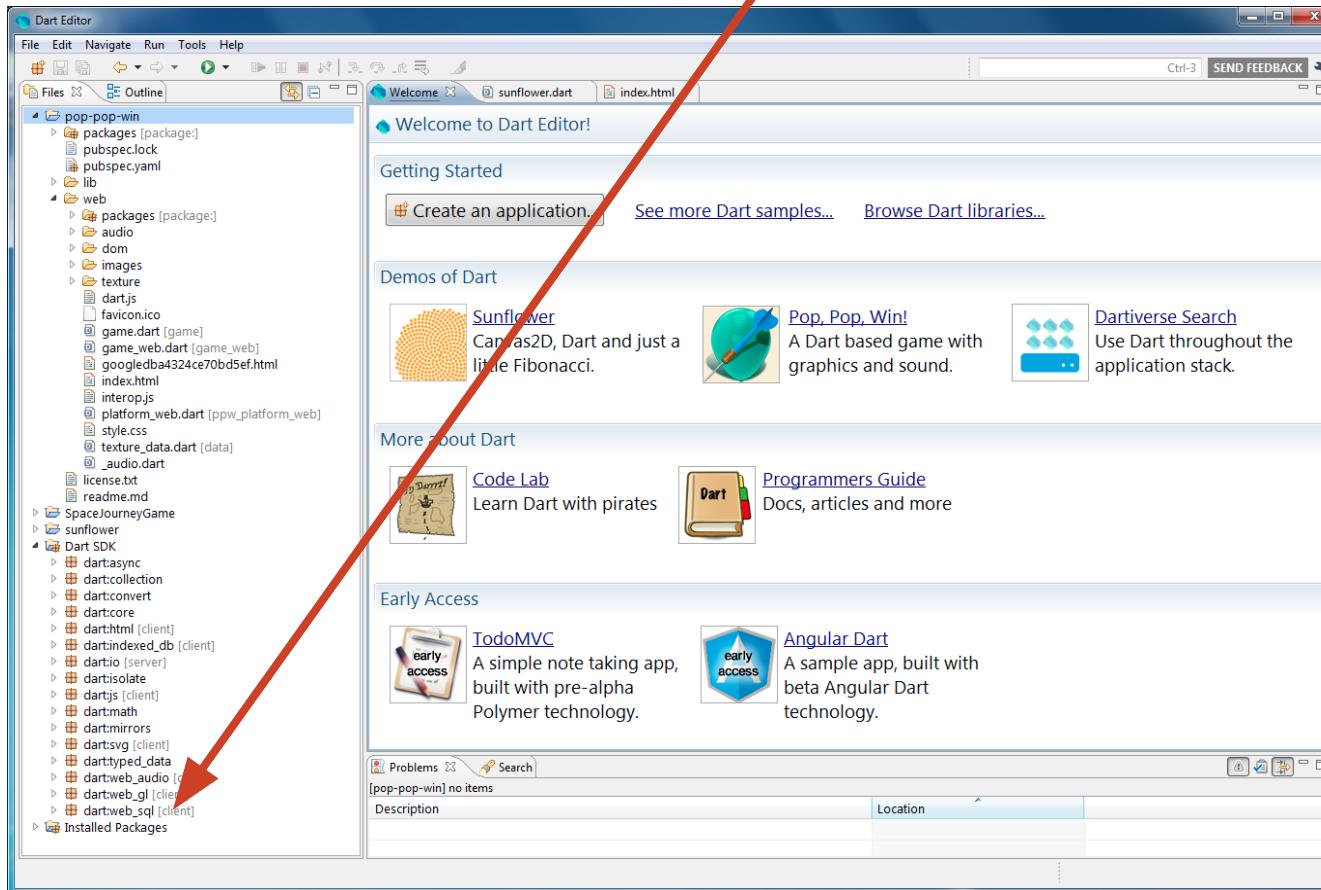
# Dart Editor

## Dart SDK Libraries



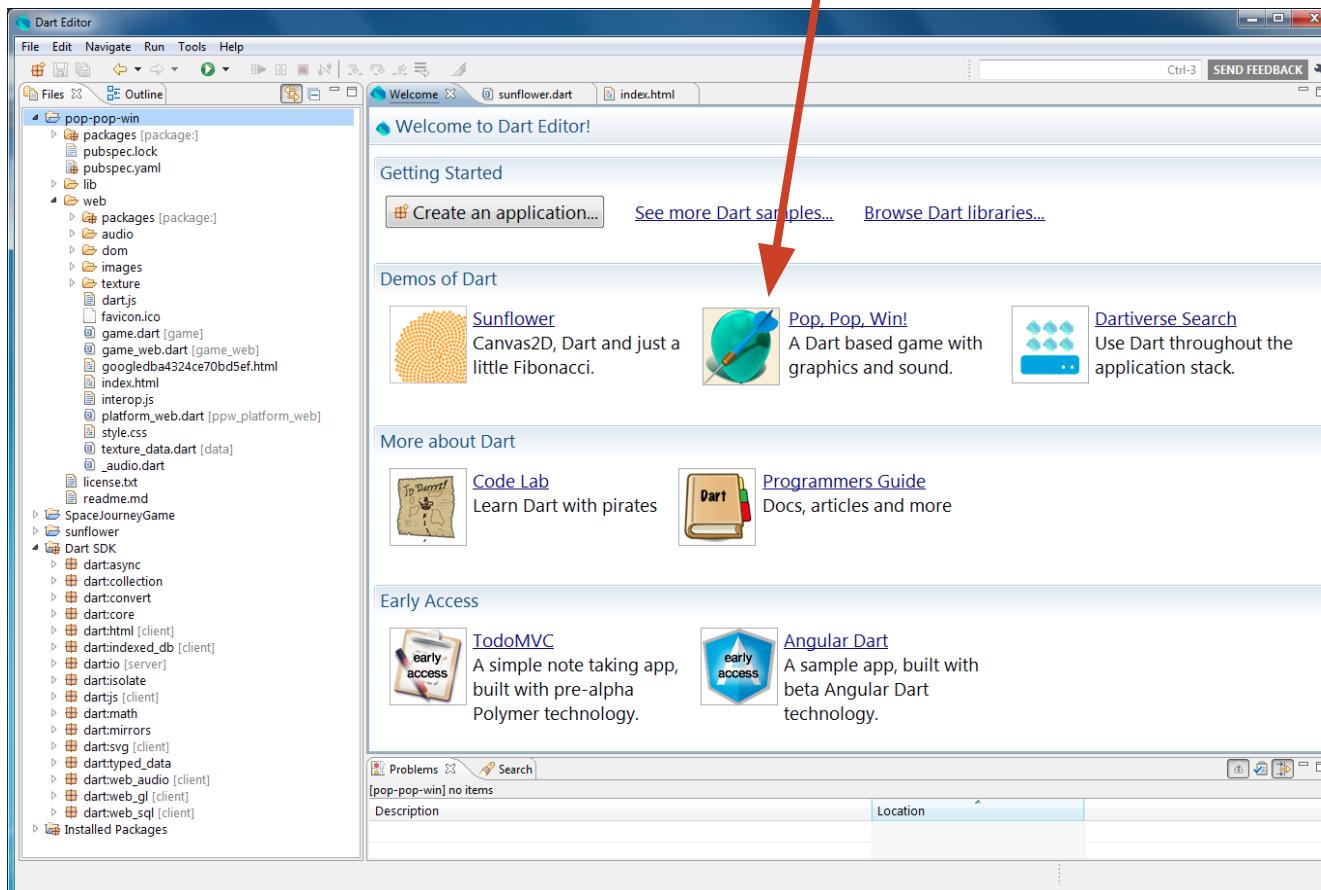
# Dart Editor

## Installed Pub Packages



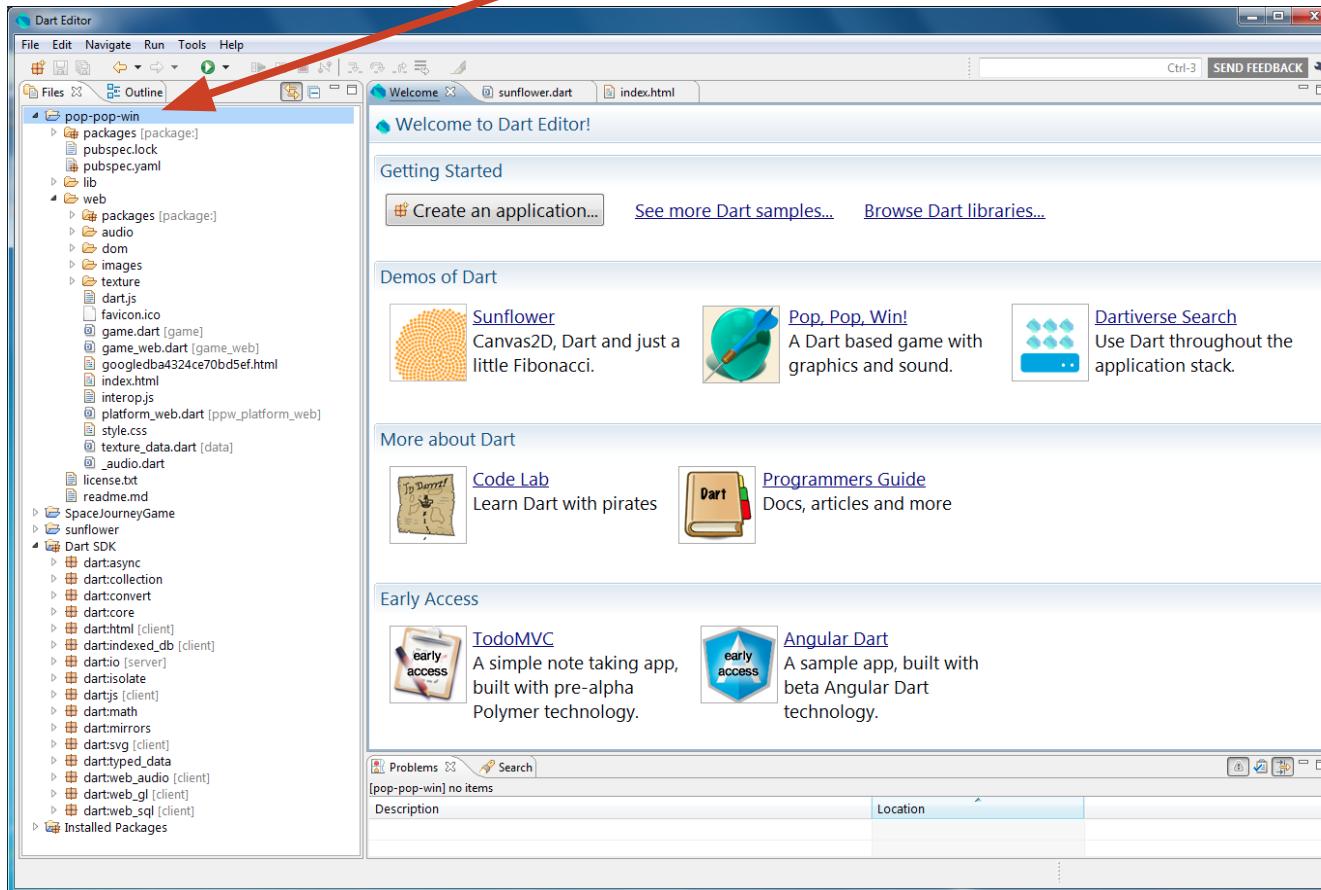
# Dart Editor

## Load Pop Pop Win Sample



# Dart Editor

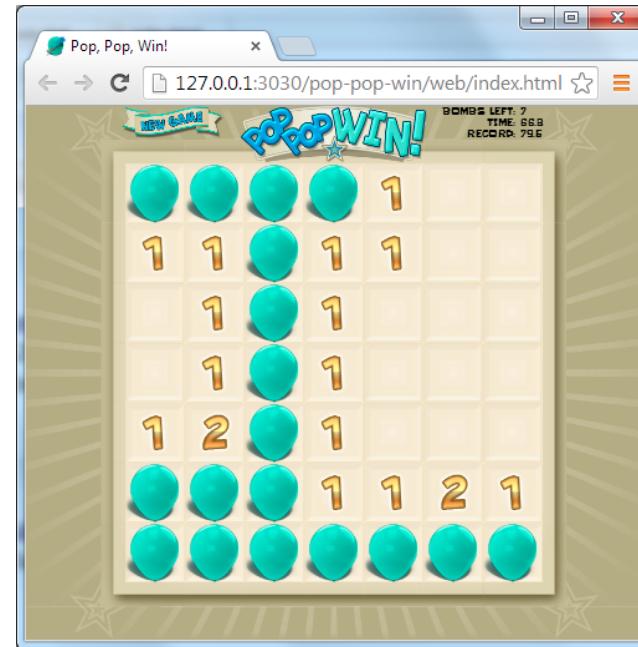
## Pop Pop Win Sample



# Sample

Pop Pop Win

- Dartium
- JavaScript
- Debugging



Source



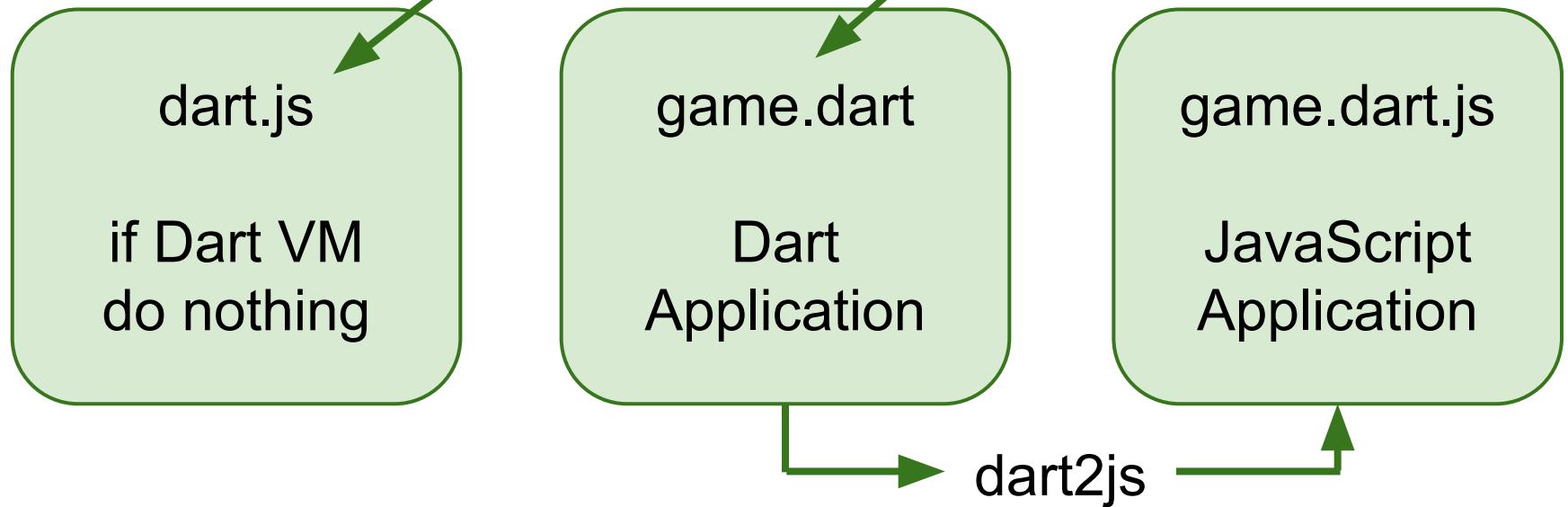
dart2js



# HTML + Dart



```
<html>
  <body>
    <script type="application/dart" src="game.dart"/>
    <script type=".../dart.js"/>
```

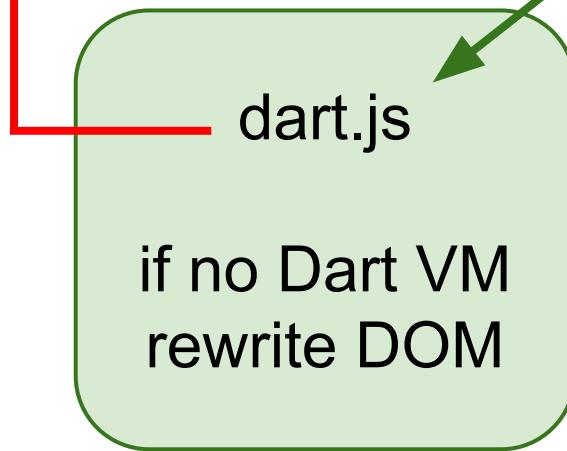


# HTML + Dart



Others

```
<html>
  <body>
    <script type="application/dart" src="game.dart.js"/>
    <script type=".../dart.js"/>
```



# dart2js

Converts Dart source to JavaScript

- Targets ES5+ (modern browsers)
- Tree shaking and dead code elimination
- Written in Dart

In progress:

- Smaller JavaScript output
- Performance improvements

# Pub - package manager

## Declaration

- `pubspec.yaml` defines the package

## Operations

- `pub get` installs dependencies
- `pub upgrade` upgrade to latest

See packages on <http://pub.dartlang.org>



# Pub - site

pub.dartlang.org    Getting Started    Docs    Packages    Search

## Welcome to pub.dartlang.org!

This is your friendly repository of packages of software for the [Dart programming language](#). Explore packages here and install them using **pub**, the package manager for Dart.

[Get started](#)

### Recently updated packages

Package	Version	Description	Updated
<a href="#">http_server</a>	0.9.1	Library of HTTP server classes.	Dec 10, 2013
<a href="#">web_ui</a>	0.4.27+2	(deprecated) new versions are in the Polymer.dart package	Dec 10, 2013
<a href="#">math_expressions</a>	0.0.8	A math library for parsing and evaluating expressions in real, interval and vector contexts. Also supports simplification and differentiation.	Dec 10, 2013
<a href="#">csslib</a>	0.9.1	A library for parsing CSS.	Dec 10, 2013
<a href="#">grinder</a>	0.4.4	Grinder - a task based, dependency aware build system.	Dec 10, 2013

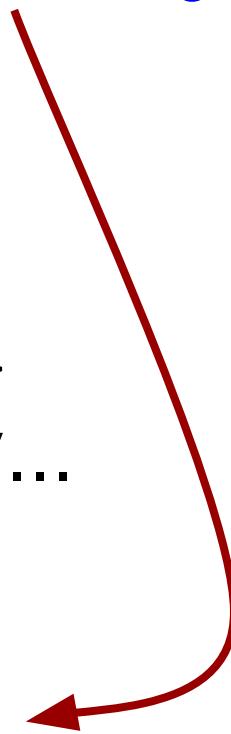
[More packages...](#)

**Fun facts:** 500+ packages, 1.5 Million LOC,  
top 20 packages have been downloaded 700K+ times

# Pub - pubspec.yaml

Semantic Versioning  
<http://semver.org/>

```
name: my_app
description: some application
version: 1.2.7
author: Bob <bob@smith.org>
homepage: http://www.smith.org/...
dependencies:
    one_package: any
    another_package: "1.2.1-dev4.0"
    my_package: ">=0.2.8+4 <0.2.9"
```





# Pub - layout

**my\_app/**

**pubspec.yaml**



Package declaration

**README.md**

**bin/**

**start\_my\_app**

**lib/**

**public\_stuff.dart**

**src/**

**internal\_stuff.dart**

**test/**

**my\_app\_test.dart**

**web/**

**index.html**

**main.dart**

**style.css**



# Pub - layout

```
my_app/
  pubspec.yaml
  README.md
  bin/ ← Server side code
    start_my_app
  lib/
    public_stuff.dart
    src/
      internal_stuff.dart
  test/
    my_app_test.dart
  web/ ← Client side code
    index.html
    main.dart
    style.css
```

# Pub - layout

```
my_app/  
  pubspec.yaml  
  README.md  
  bin/  
    start_my_app  
  lib/ ← Public code  
    public_stuff.dart  
  src/ ← Internal code  
    internal_stuff.dart  
test/  
  my_app_test.dart  
web/  
  index.html  
  main.dart  
  style.css
```

The diagram illustrates the directory structure of a Dart project named 'my\_app'. It shows the following files and folders:

- my\_app/**: The root directory containing **pubspec.yaml** and **README.md**.
- bin/**: A directory containing **start\_my\_app**.
- lib/**: A directory containing **public\_stuff.dart**. A red arrow points from the left towards this folder, labeled "Public code".
- src/**: A directory containing **internal\_stuff.dart**. A red arrow points from the left towards this folder, labeled "Internal code".
- test/**: A directory containing **my\_app\_test.dart**.
- web/**: A directory containing **index.html**, **main.dart**, and **style.css**.



# Pub - layout

```
my_app/
  pubspec.yaml
  README.md
  bin/
    start_my_app
  lib/
    public_stuff.dart
    src/
      internal_stuff.dart
  test/ ← Tests
    my_app_test.dart
  web/
    index.html
    main.dart
    style.css
```



# Developing Dart Editor

<http://www.dartlang.org>

# Dart Editor - Users

- Web programmers of varying backgrounds
  - Many languages - HTML, JS, Python, Java
  - Wide range of programming experience
- Primarily **not** Eclipse users
- Dart Plugin available for Eclipse users



# Dart Editor - Goals

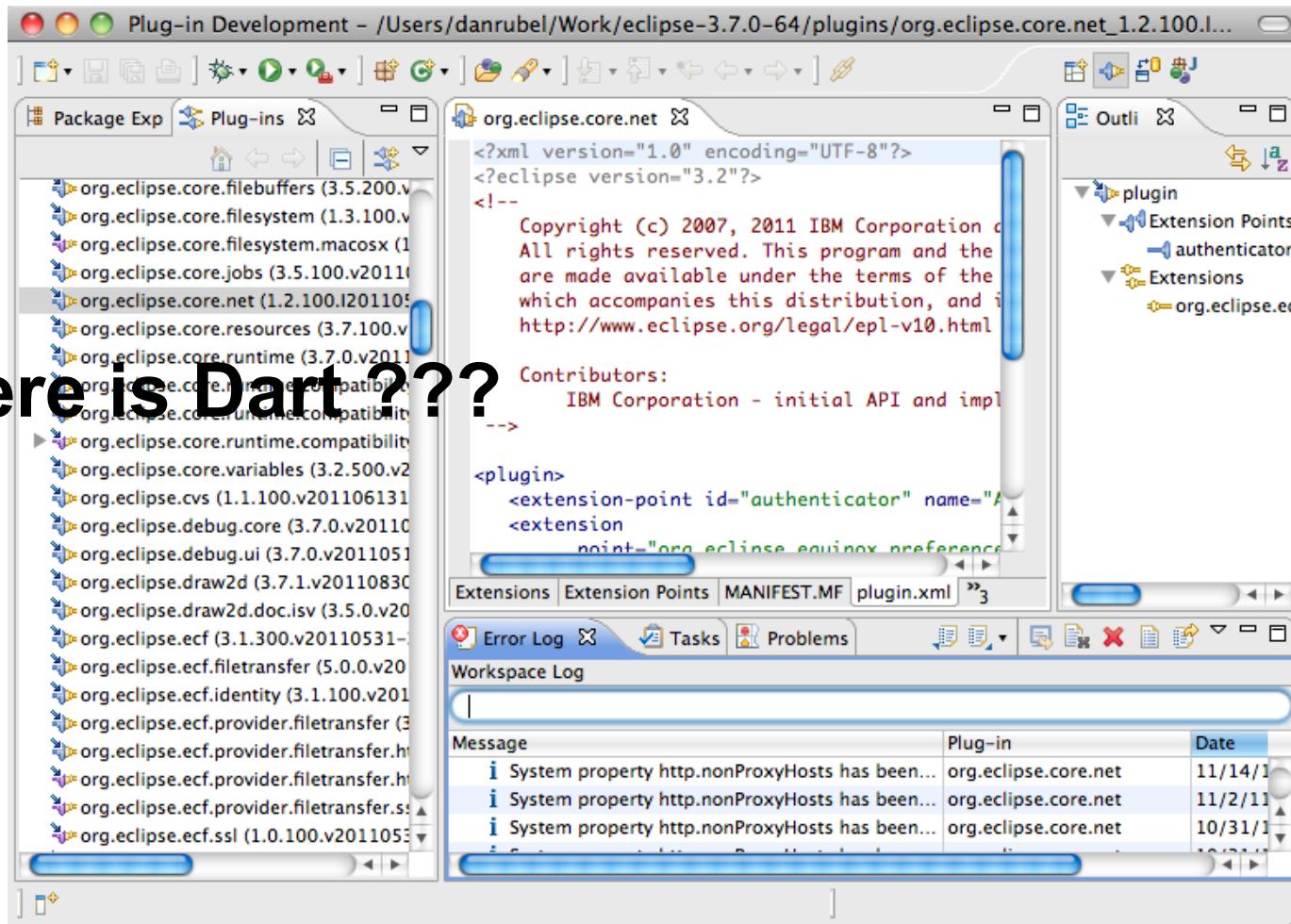
- Easy on-ramp to learn Dart
- Simplified UI

but also...

- Power tools
  - refactoring
  - quick fixes/assists
  - code completion
  - semantic search
  - debugging/inspecting
  - ...and more

# Dart Editor - Before

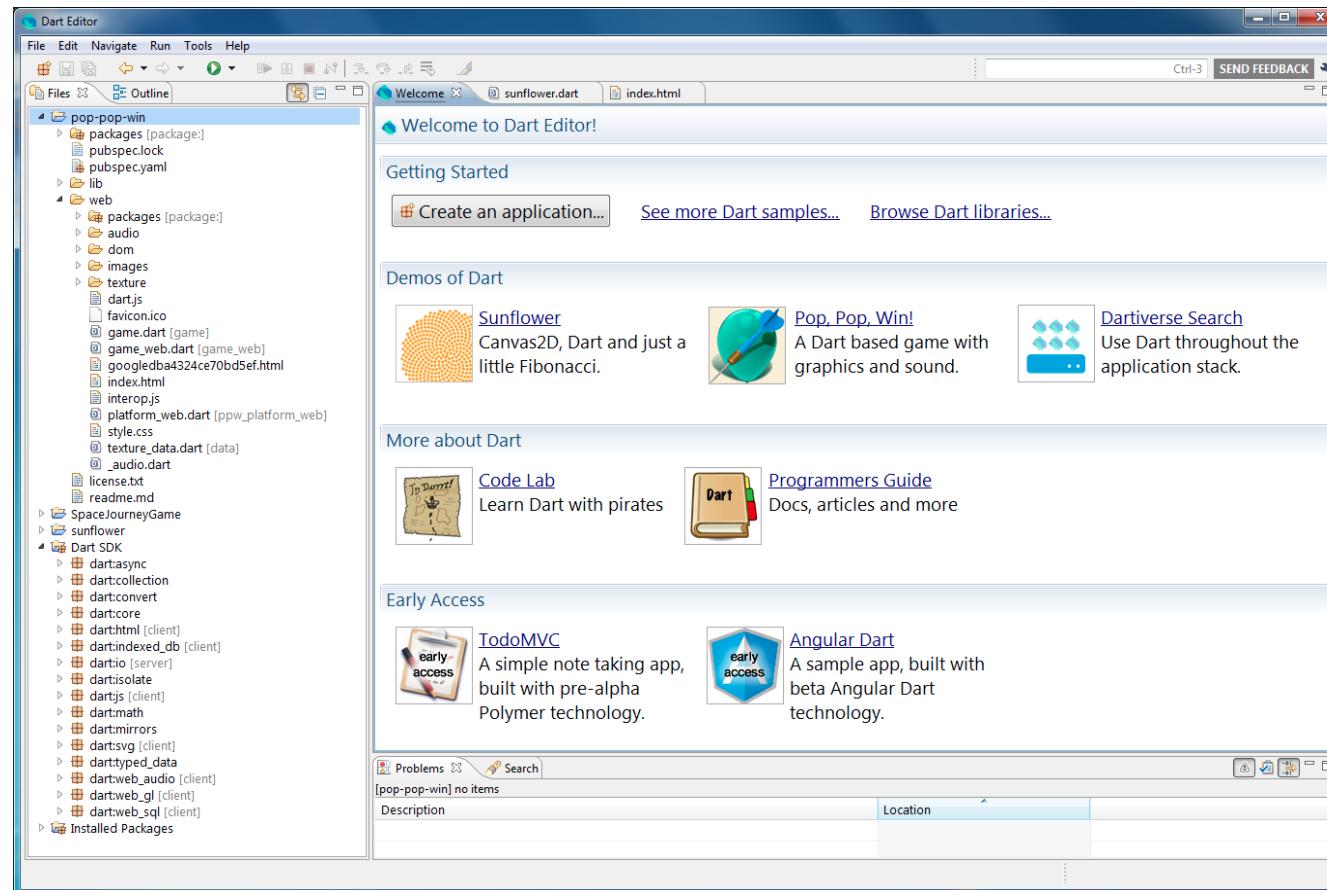
Where is Dart ???



# Dart Editor - Strategy

- Narrow the scope
  - Focus on doing a few things well
  - No Eclipse Team support
- Minimalist UI
  - Make it easy to understand
  - Reduce decision making

# Dart Editor - Now



Simple and Clean UI

# How?

- Single perspective
- Remove unnecessary plugins
- Redefine entire menu bar
- Use "activities" to suppress UI elements
- Key binding schema

# Start-up Performance

- Remove unused plugins
  - Modify plugins to remove dependencies
- Defer work until after UI appears
  - Early startup extension point
  - `Display.asyncExec(...)`
- Optimize load order
  - Record class load order
  - Reorder classes in plugin jar files

# Application Performance

- Profile and optimize the code
  - Identify hotspots with VM profiler
  - Rewrite or eliminate slow code
- Defer work to background tasks

# Performance-critical Areas

- Navigation
- Code completion
- Errors, warnings, and hints

Solutions:

- Background indexing
- Keep analysis results in memory
- Incremental analysis - ongoing work



# Metrics

First RCP build

70+ MB

200+ plugins

**25s** startup

Current build

54 MB

110 plugins

**9s** startup

# Standalone vs. Plugin issues

Same code base but not exactly the same

1) Move **.project** files to new location

- modified the FileSystemResourceManager

2) Wanted slightly different **code completion behavior**

- modified CompletionProposalPopup

3) **Back button** default if refactoring has errors

- modified RefactoringStatusDialog

# Standalone vs. Plugin issues

## Standalone

- Unique experience
- Specialized HTML editing
- Omnibox search

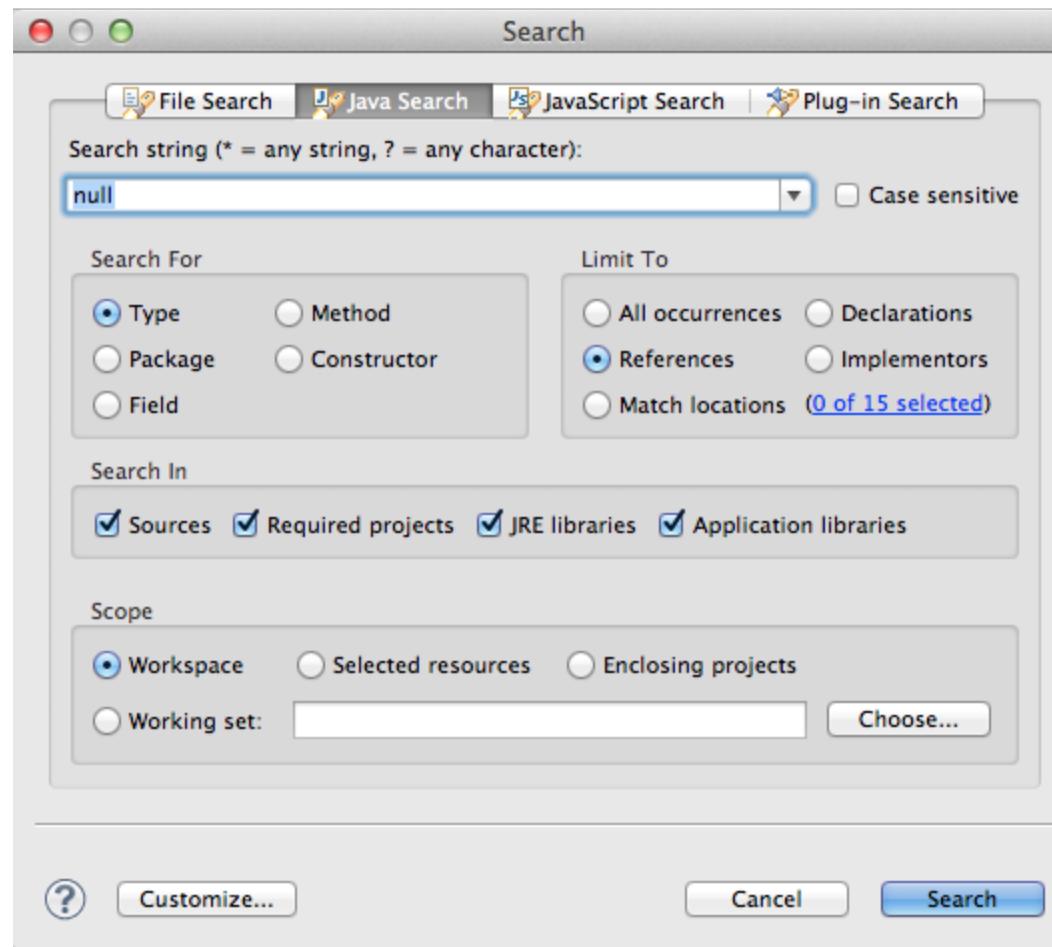
## Plugins

- Integrated feel
- Eclipse HTML editing
- Eclipse search

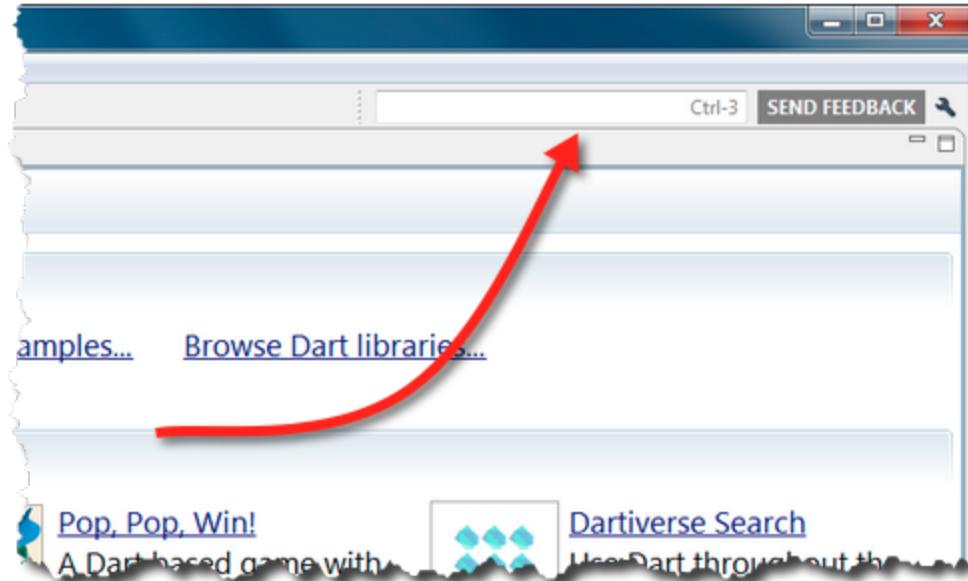
# Decision to use WTP

- Dart programs ⇒ .dart, .html and .css files
  - HTML & CSS editing needed
- Initially rolled our own light-weight version
  - Too many missing features and other problems
- Reluctantly decided to include (partial) WTP
  - Worried about size and unnecessary features
  - Stripped it down to remove unneeded pieces

# Eclipse Search



# Omnibox experience



Wanted a simple way to find almost anything

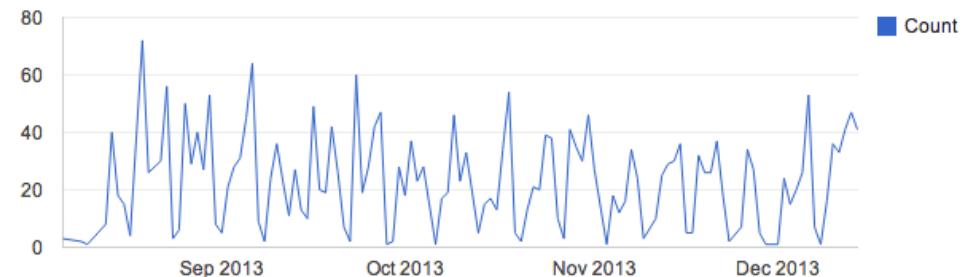
# Instrumentation

- Performance and error data
  - Timing for each task
  - Recurring exceptions
  - Limited to internal Google-only due to privacy
- UI Watchdog
  - Periodically check that the UI thread is responsive
  - Report periods when UI is blocked and by what

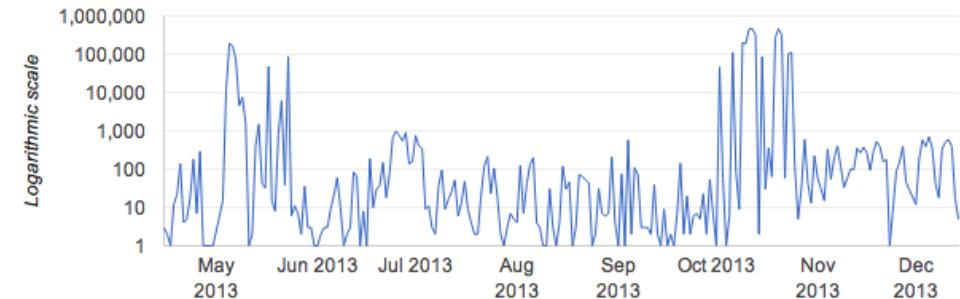
# Instrumentation Results

- Exceptions
- Memory usage
- Version uptake
- Usage patterns
- Overall “health”

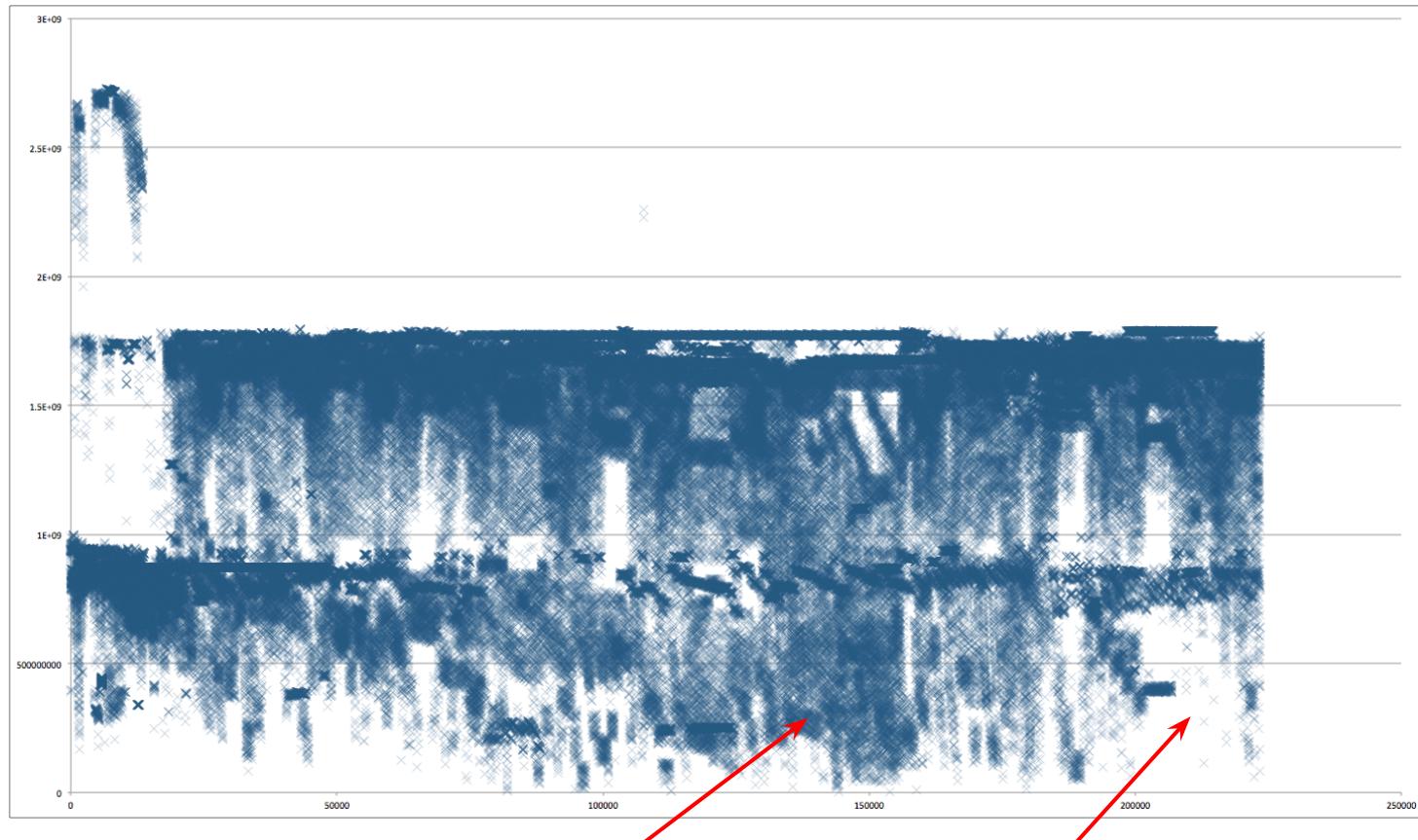
## ▼ Health Reports usage



## ▼ Exceptions/100



# Instrumentation - Memory Usage

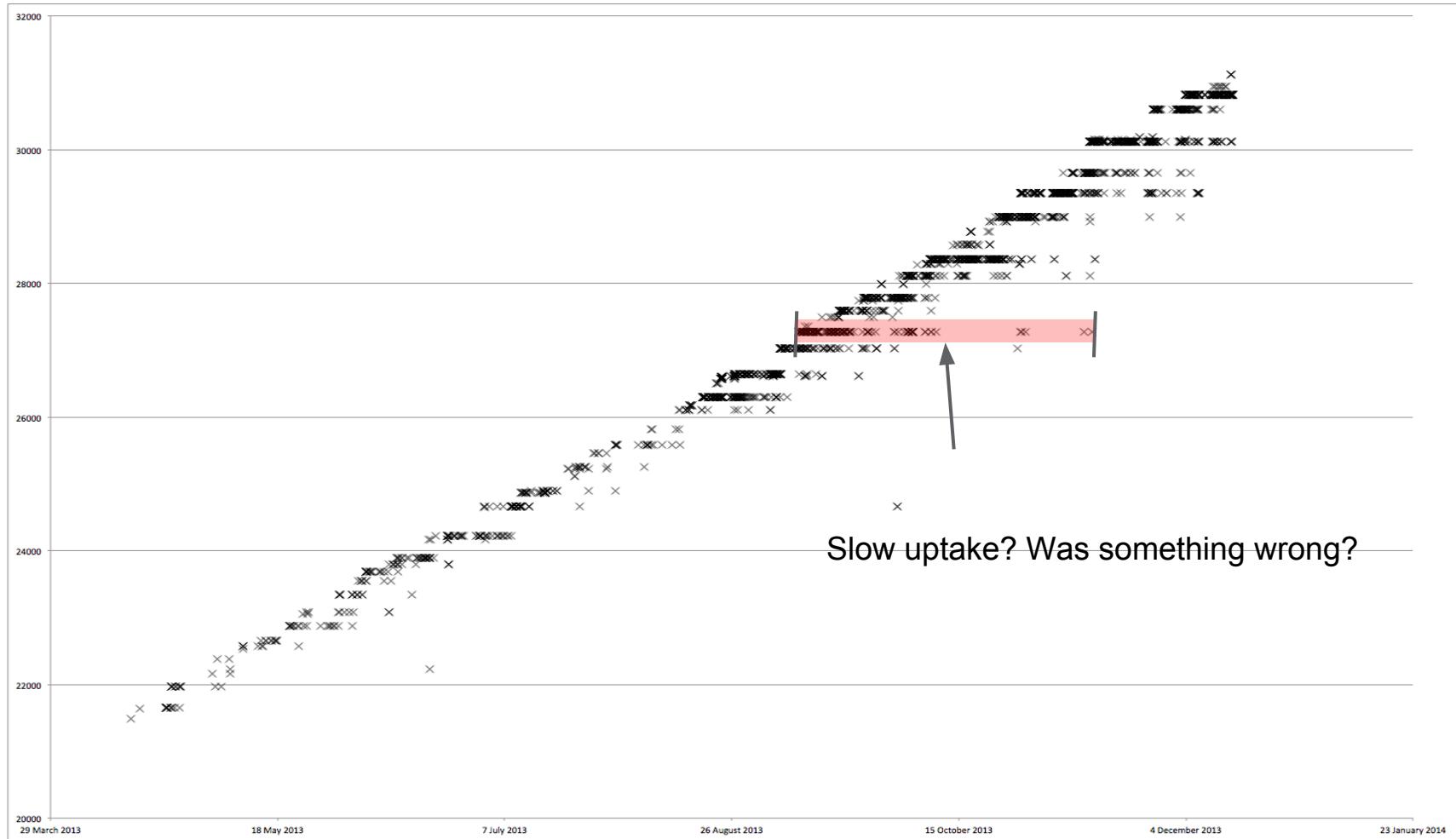


Memory problems

Sustained improvement

<http://www.dartlang.org>

# Instrumentation - Version Tracking



# Subsystems

## Eclipse Specific

- file navigation and code editing
  - debugging experience
  - UI for code completion, quick fixes, refactoring, ...
  - pub integration
  - background analysis
- 

## Pure Java (thus portable to Dart)

- analysis engine
  - code completion
  - quick fixes
  - refactorings
  - suggestions
  - indexer
- java2dart translator
- 

## Pure Dart

- formatter, cmd line analysis

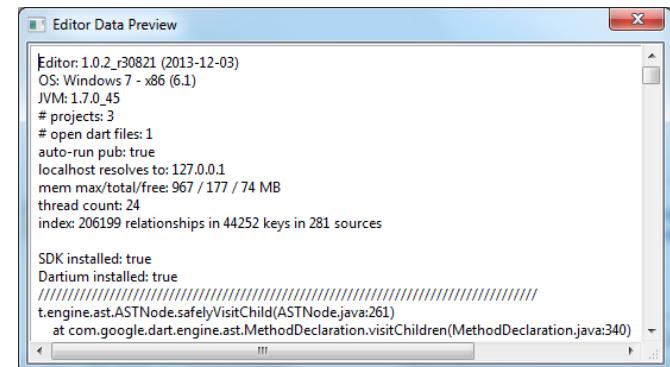
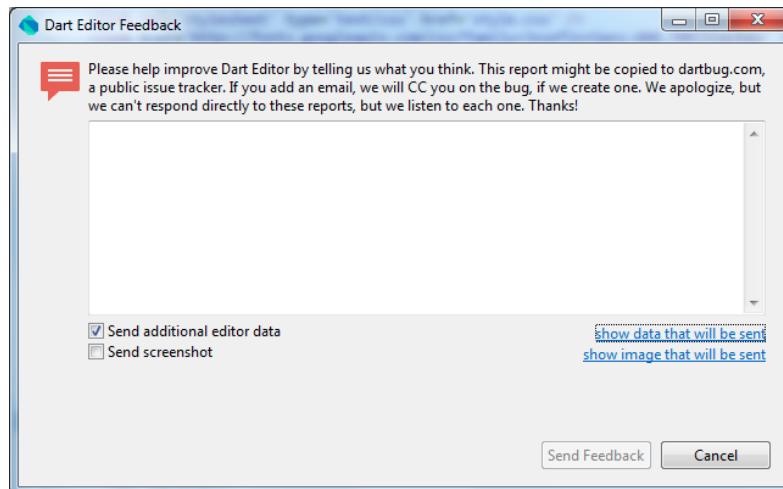
# Testing

- Unit tests
- Integration tests
- Functional tests
- Continuous test execution



# Feedback & Privacy

- Needed to collect high quality feedback
  - Description, OS info, memory info, screenshot
- Always opt-in and anonymous



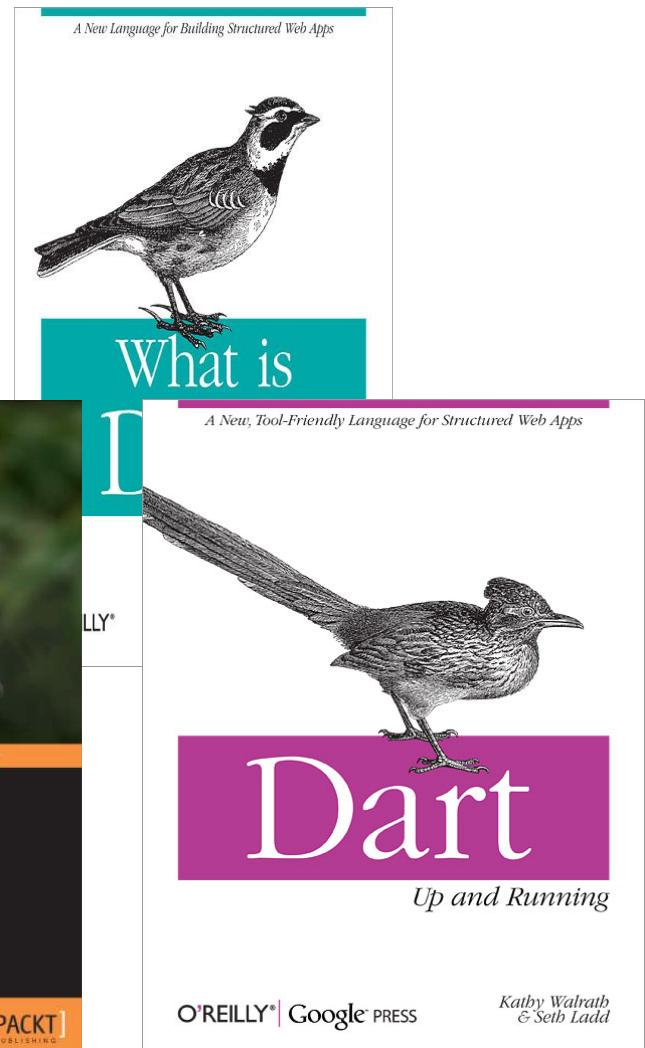
# Why Eclipse 3.x and not 4.x?

- Eclipse 3.x much faster when we started
- Not really an issue now, but costs to switch
  - Inertia
  - Time to move code
  - Time to retest everything
  - Time to update automated UI tests



# Books

<https://www.dartlang.org/books/>





# Questions?

More information....

**<https://dartlang.org>**

Introduction, language spec, articles

Download Dart Editor

**<https://code.google.com/p/dart/>**

Source code to editor, compiler, and virtual machine

See the wiki for instructions



# Community

- G+: [google.com/+dartlang](http://google.com/+dartlang)
- Mailing list: [misc@dartlang.org](mailto:misc@dartlang.org)
- Stack Overflow: Tag **dart**
- Twitter: [@dart\\_lang](https://twitter.com/dart_lang)
- Hashtag: [#dartlang](#)
- Blogs: <http://dartosphere.org>
- IRC: [#dart](#)  
<http://www.dartlang.org>