



Dan Rubel
Eric Clayberg



Overview

- Dart Project
- Dart Language
- Client-side Dart
- Server-side Dart
- Developing Dart Editor
- Questions



DART Project

The main header features a larger version of the DART logo on the left, which is a black rectangle containing the blue and cyan geometric shape and the word "DART" in grey. To the right of this logo, the word "Project" is written in a large, bold, black sans-serif font.

<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 ...

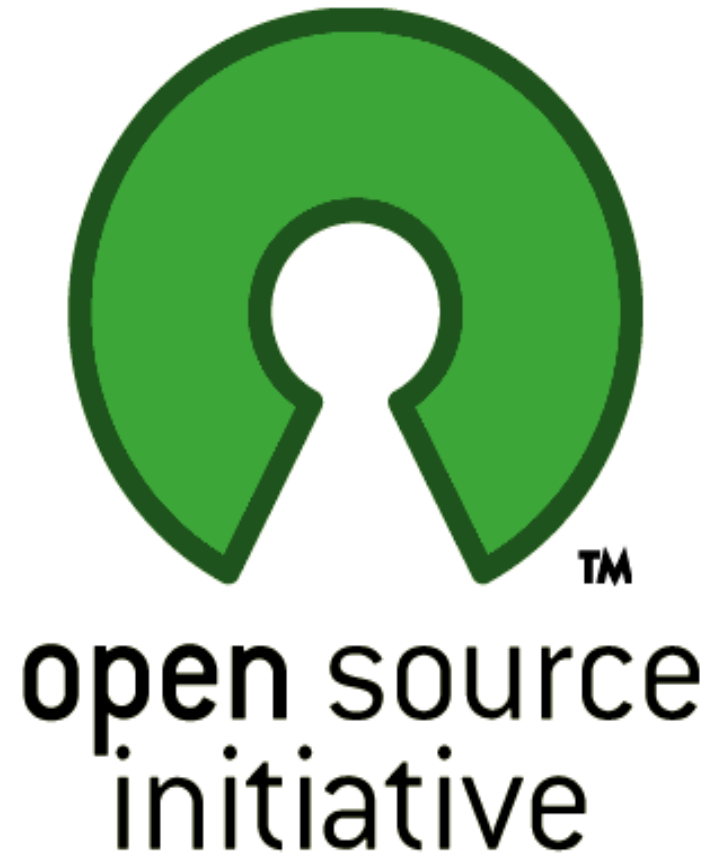
- Difficult to optimize performance
- Difficult to maintain large code bases
- Difficult to assert correctness



What is Dart?

Dart is open source

- BSD-style license
- dart.googlecode.com
- GitHub mirror
- Contributing guide

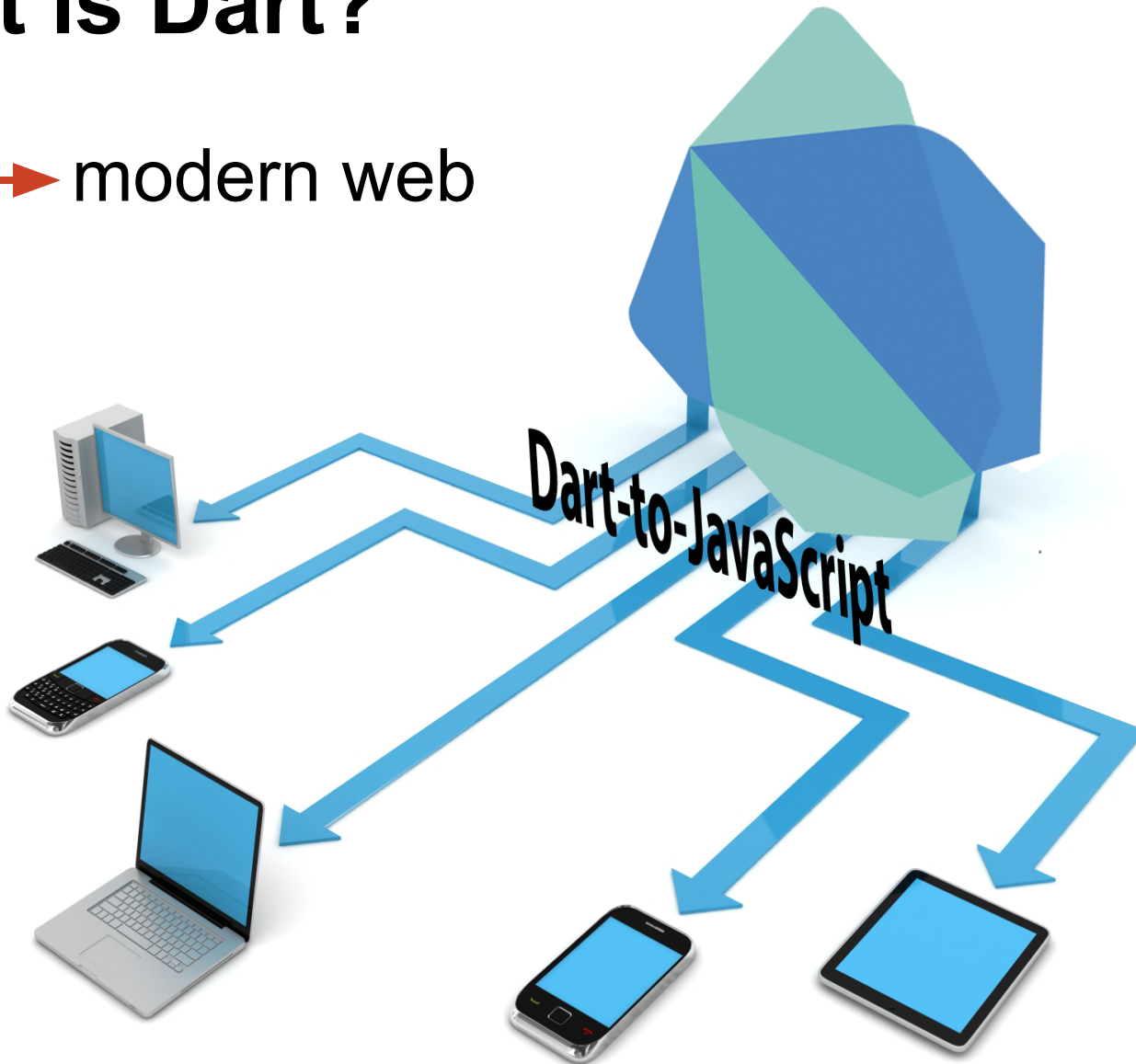


<http://www.dartlang.org>



What is Dart?

Dart → modern web





What is Dart?

- Rich client apps
- Server apps
- Offline-capable
- 60fps
- ES5+
- HTML5



<http://www.dartlang.org>

What is Dart?

Language

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

Editor

Virtual machine

Compiler to JavaScript

Browser integration

Package manager

Web Components integration

Community

"Batteries Included"





Language

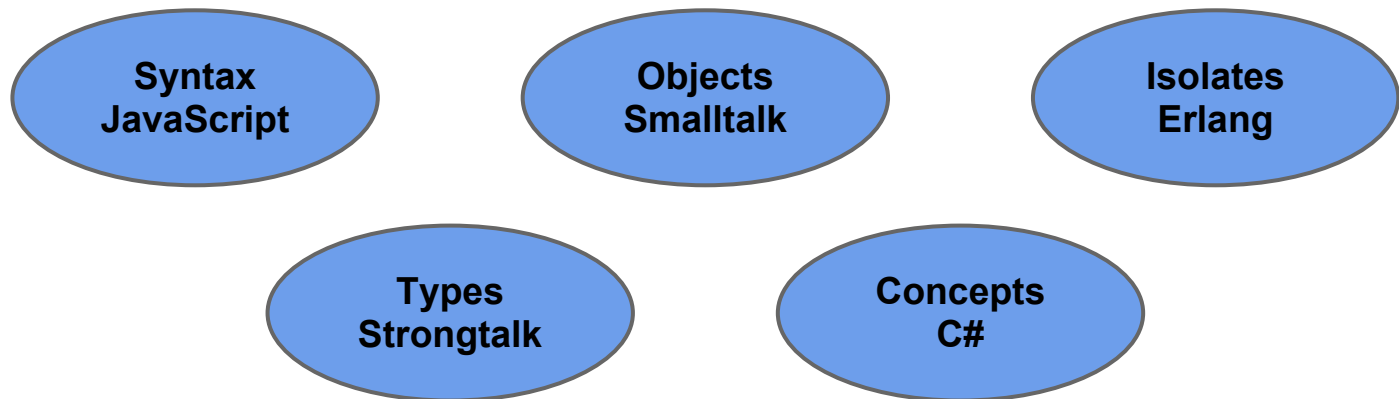
<http://www.dartlang.org>



Language Goals

- Unsurprising simple and familiar
- Class-based single inheritance
- Proper lexical scoping
- Single-threaded with isolates
- Optional static types

Dart =





Optional Static Types

Static types ...

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

You can mix typed and untyped code ...

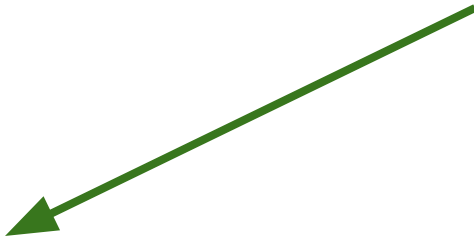
```
var p = new Point(2, 3);  
int x = 4;  
print (p + new Point(x, 5));
```

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

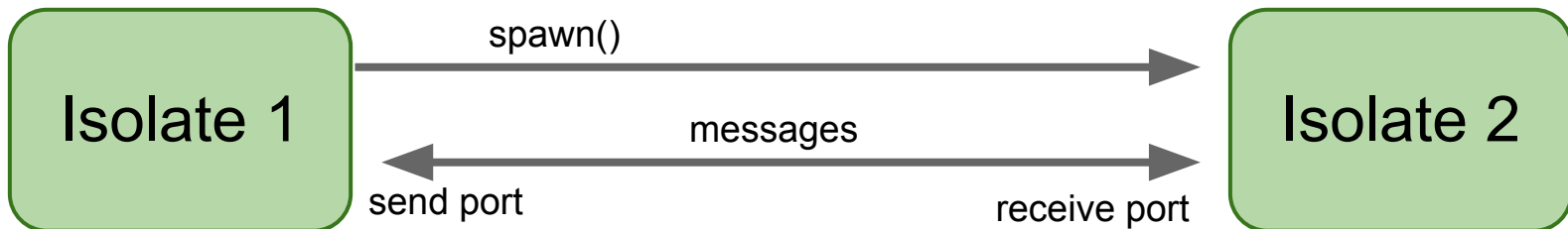
```
class PrettyPrinter extends ASTTool with Counter
{
  ...
}
```

```
class Counter {
  int count = 0;
  int get increment ==> ++count;
  int get decrement => --count;
}
```

... available early 2013

Isolates

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





Client-side DART



Hello World

```
import 'dart:html';

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

class Hello {
  void welcome(String name) {
    var message = "Hello $name";
    document.query('#status')
      ..text = message
      ..on.click.add(handler);
  }
}
```



Hello World

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

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

```
class Hello {  
  void welcome(String name) {  
    var message = "Hello $name";  
    document.query('#status')  
      ..text = message  
      ..on.click.add(handler);  
  }  
}
```

Libraries

A red arrow originates from the word "Libraries" and points to the `import 'dart:html';` line in the code above.



Hello World

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

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

← Functions

```
class Hello {  
  void welcome(String name) {  
    var message = "Hello $name";  
    document.query('#status')  
      ..text = message  
      ..on.click.add(handler);  
  }  
}
```



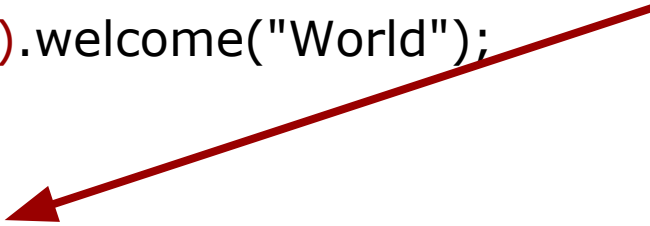
Hello World

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

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

```
class Hello {  
  void welcome(String name) {  
    var message = "Hello $name";  
    document.query('#status')  
      ..text = message  
      ..on.click.add(handler);  
  }  
}
```

Classes





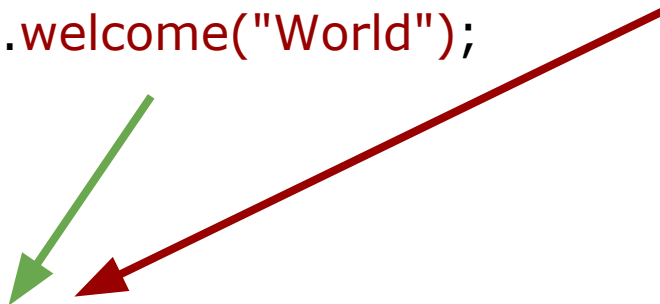
Hello World

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

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

```
class Hello {  
  void welcome(String name) {  
    var message = "Hello $name";  
    document.query('#status')  
      ..text = message  
      ..on.click.add(handler);  
  }  
}
```

Methods





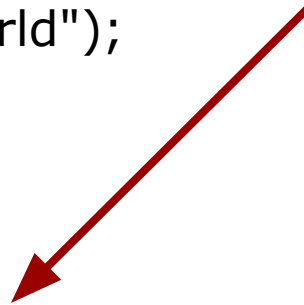
Hello World

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

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

```
class Hello {  
  void welcome(String name) {  
    var message = "Hello $name";  
    document.query('#status')  
      ..text = message  
      ..on.click.add(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.query('#status')  
      ..text = message  
      ..on.click.add(handler);  
  }  
}
```

Optional Parameters

A red arrow originates from the text "Optional Parameters" and points diagonally down and to the left, ending at the parameter list "[String name = \"Bob\"]" in the code block above.



Hello World

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

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

```
class Hello {  
  void welcome([String name = "Bob"]) {  
    var message = "Hello $name";  
    document.query('#status')  
      ..text = message  
      ..on.click.add(handler);  
  }  
}
```

Optional Types

A red arrow originates from the text "Optional Types" and points to the "var" keyword in the code block below.



Hello World

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

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

```
class Hello {  
  void welcome([String name = "Bob"]) {  
    String message = "Hello $name";  
    document.query('#status')  
      ..text = message  
      ..on.click.add(handler);  
  }  
}
```

Optional Types

A red arrow originates from the text "Optional Types" and points to the **String** type annotation in the code block above.



Hello World

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

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

```
class Hello {  
  void welcome([String name = "Bob"]) {  
    String message = "Hello $name";  
    document.query('#status')  
      ..text = message  
      ..on.click.add(handler);  
  }  
}
```

String Interpolation

A red arrow originates from the text "String Interpolation" and points to the "\$name" part of the string "Hello \$name" in the code block above.



Hello World

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

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

Cascades

```
class Hello {  
  void welcome([String name = "Bob"]) {  
    String message = "Hello $name";  
    document.query('#status')  
      ..text = message  
      ..on.click.add(handler);  
  }  
}
```



Hello World

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

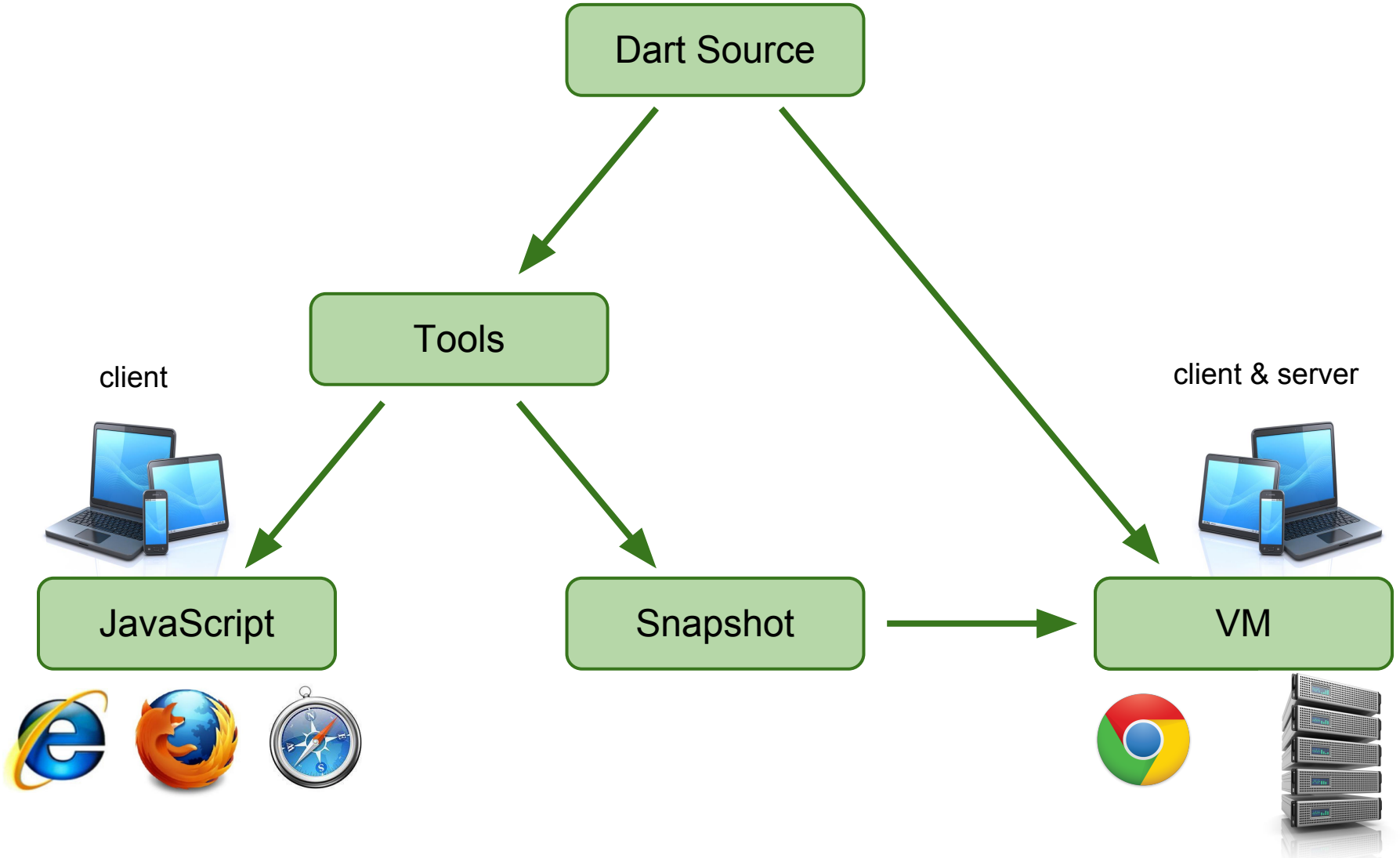
```
void main() {  
  new Hello().welcome("World");  
}
```

```
class Hello {  
  void welcome([String name = "Bob"]) {  
    String message = "Hello $name";  
    document.query('#status')  
      ..text = message  
      ..on.click.add(handler);  
  }  
}
```

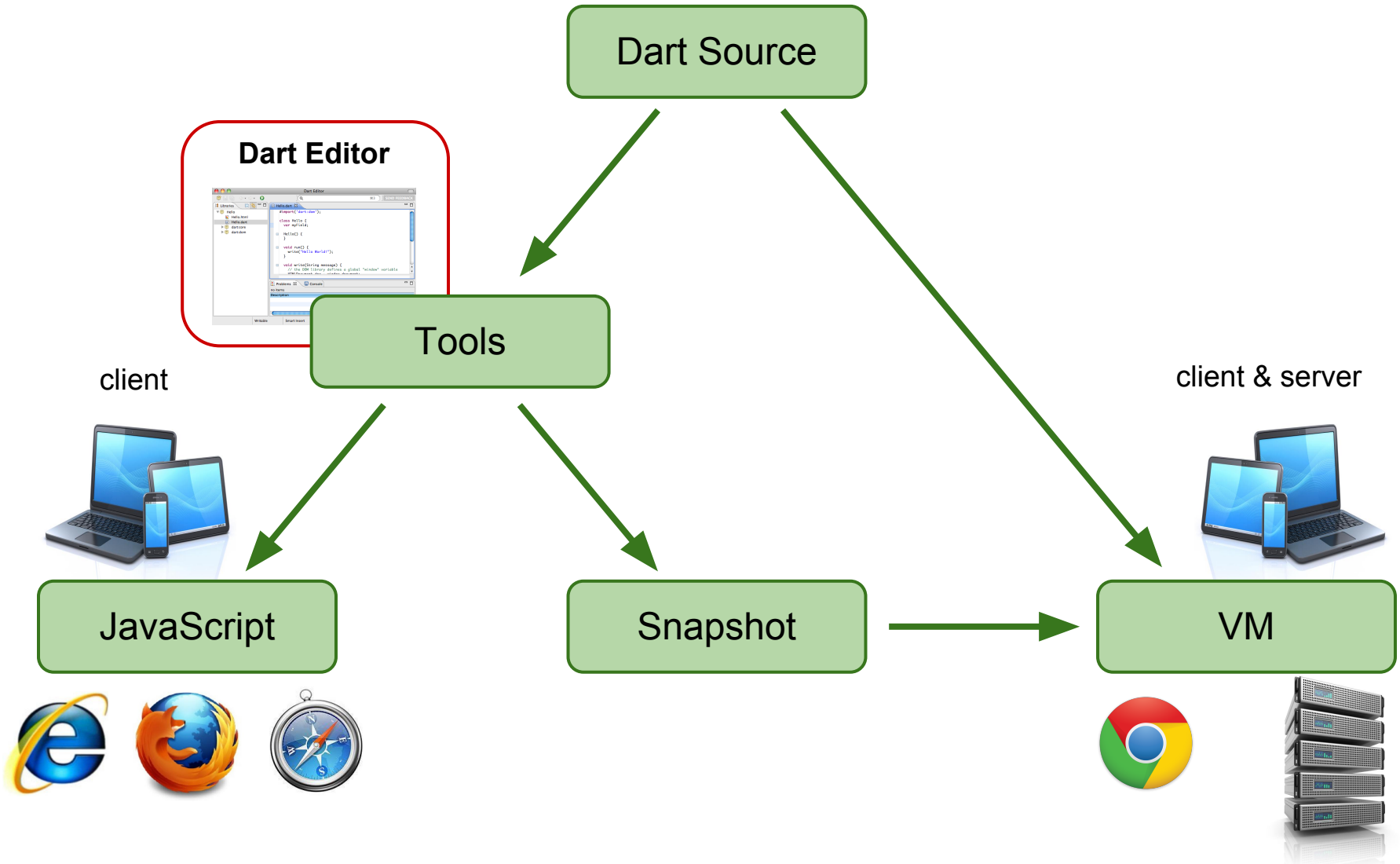
Simple DOM API

A red arrow originates from the text "Simple DOM API" and points diagonally down and to the left, ending at the red text in the code line `..on.click.add(handler);`.

Tools



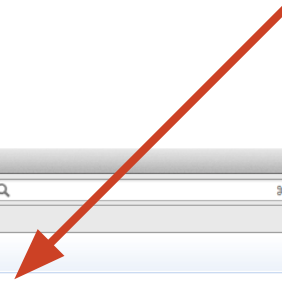
Tools





Dart Editor

Welcome Page

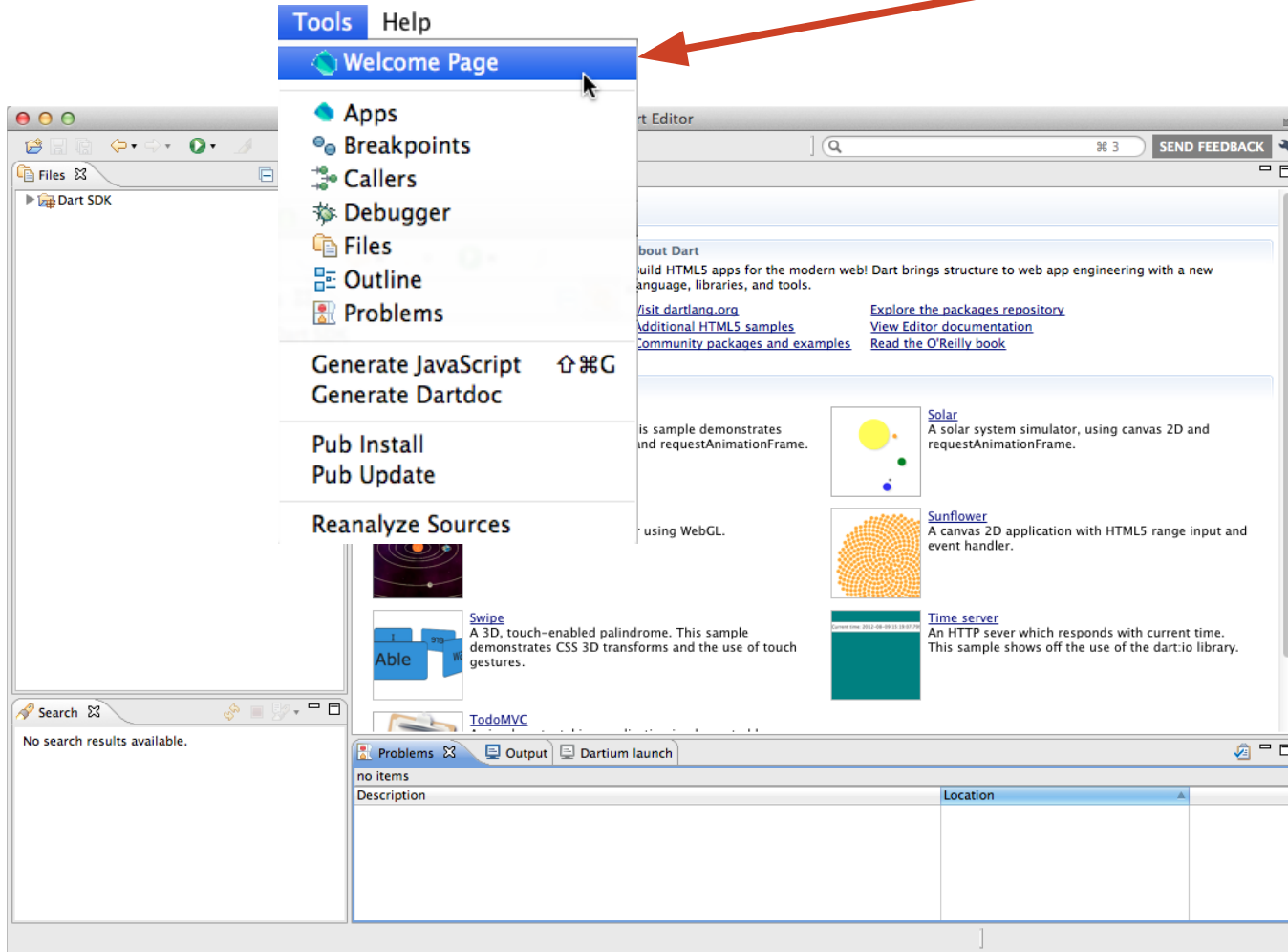


A screenshot of the Dart Editor application window. The window title is "Dart Editor". The interface is divided into several panes. On the left is a "Files" pane showing a "Dart SDK" folder. Below it is a "Search" pane with the text "No search results available." The main area is the "Welcome" page, which has a header "Welcome to Dart Editor!". Below the header are two columns of content. The left column is titled "Getting Started" and contains two buttons: "Create an application..." and "Open existing code...". The right column is titled "About Dart" and contains a paragraph of text followed by three links: "Visit dartlang.org", "Additional HTML5 samples", and "Community packages and examples". Below this is a "Sample Applications" section with six items, each with a small image and a description: "Clock", "Solar", "Solar 3D", "Sunflower", "Swipe", and "Time server". At the bottom of the window is a "Problems" pane with the text "no items" and a table with columns "Description" and "Location".



Dart Editor

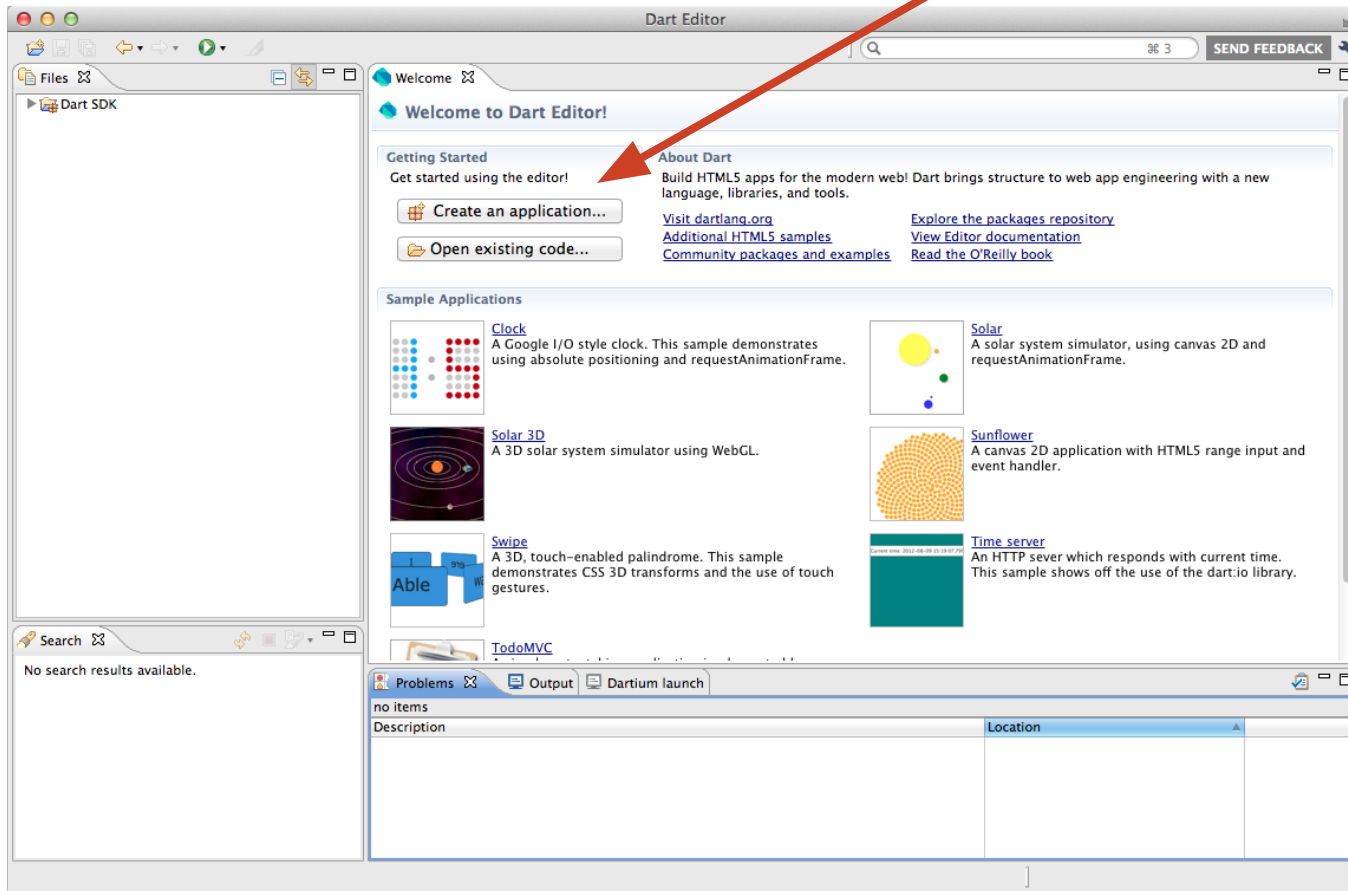
Welcome Page





Dart Editor

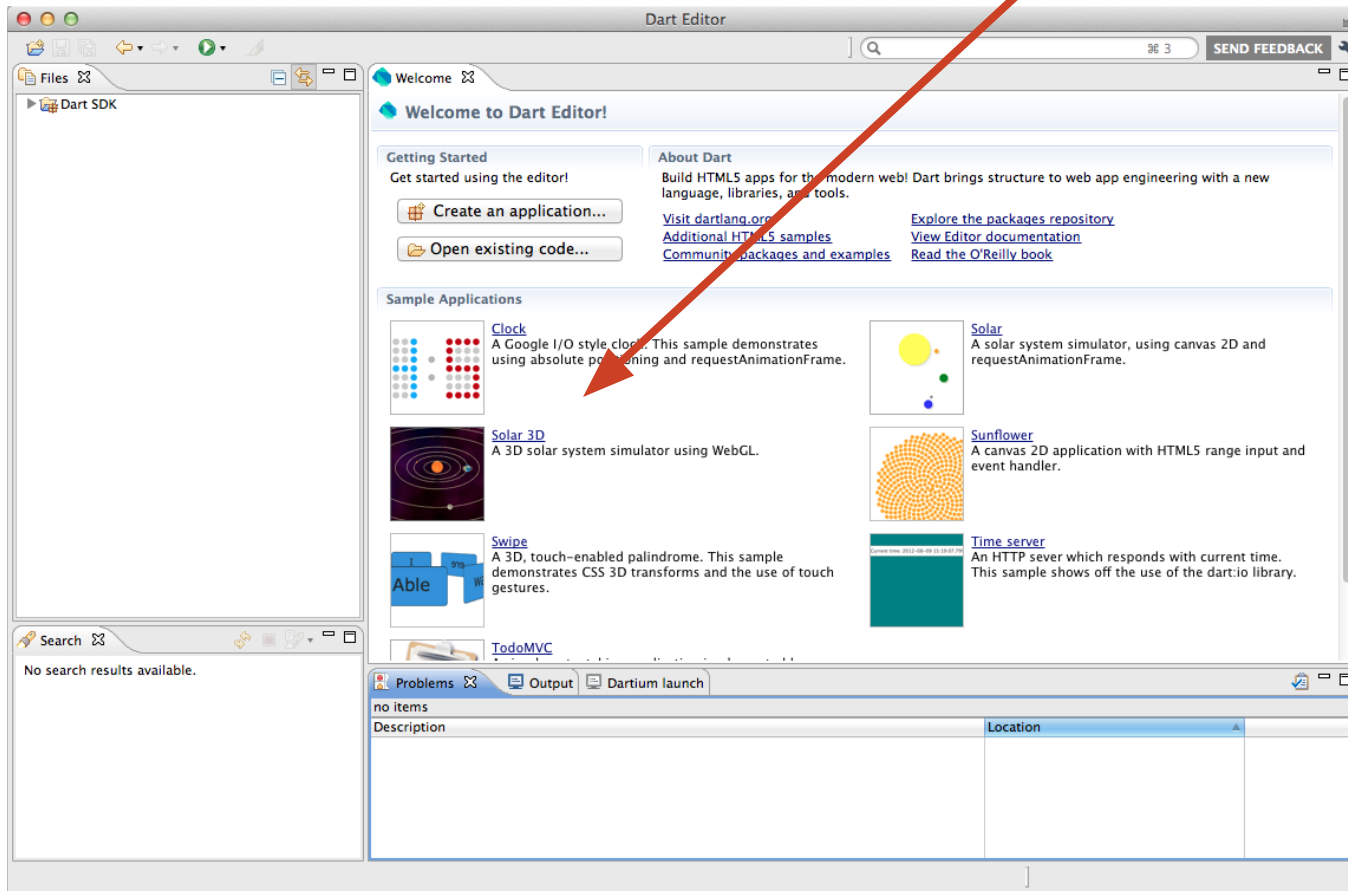
Create application





Dart Editor

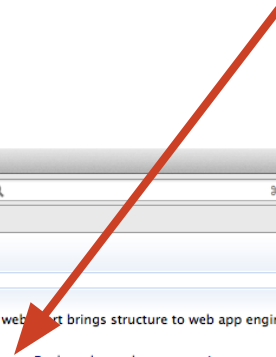
Load Sample Code





Dart Editor

More Information

A screenshot of the Dart Editor application. The window title is "Dart Editor". The interface includes a "Files" sidebar on the left showing the "Dart SDK" directory. The main area displays a "Welcome to Dart Editor!" message. Under "Getting Started", there are buttons for "Create an application..." and "Open existing code...". The "About Dart" section describes building HTML5 apps and provides links for "Visit dartlang.org", "Additional HTML5 samples", "Community packages and examples", "Explore the packages repository", "View Editor documentation", and "Read the O'Reilly book". The "Sample Applications" section lists several examples: "Clock", "Solar 3D", "Swine", "Able", "Solar", "Sunflower", and "Time server". At the bottom, there are tabs for "Problems", "Output", and "Dartium launch". The "Problems" tab is active and shows "no items".

Files

Dart SDK

Welcome

Welcome to Dart Editor!

Getting Started
Get started using the editor!

Create an application...

Open existing code...

About Dart
Build HTML5 apps for the modern web. Dart brings structure to web app engineering with a new language, libraries, and tools.

[Visit dartlang.org](#)
[Additional HTML5 samples](#)
[Community packages and examples](#)

[Explore the packages repository](#)
[View Editor documentation](#)
[Read the O'Reilly book](#)

Sample Applications

Clock
A Google I/O style clock. This sample demonstrates using absolute positioning and requestAnimationFrame.

Solar 3D
A 3D solar system simulator using WebGL.

Swine
A 3D, touch-enabled palindrome. This sample demonstrates CSS 3D transforms and the use of touch gestures.

Able

Solar
A solar system simulator, using canvas 2D and requestAnimationFrame.

Sunflower
A canvas 2D application with HTML5 range input and event handler.

Time server
An HTTP sever which responds with current time. This sample shows off the use of the dart:io library.

TodoMVC

Search

No search results available.

Problems

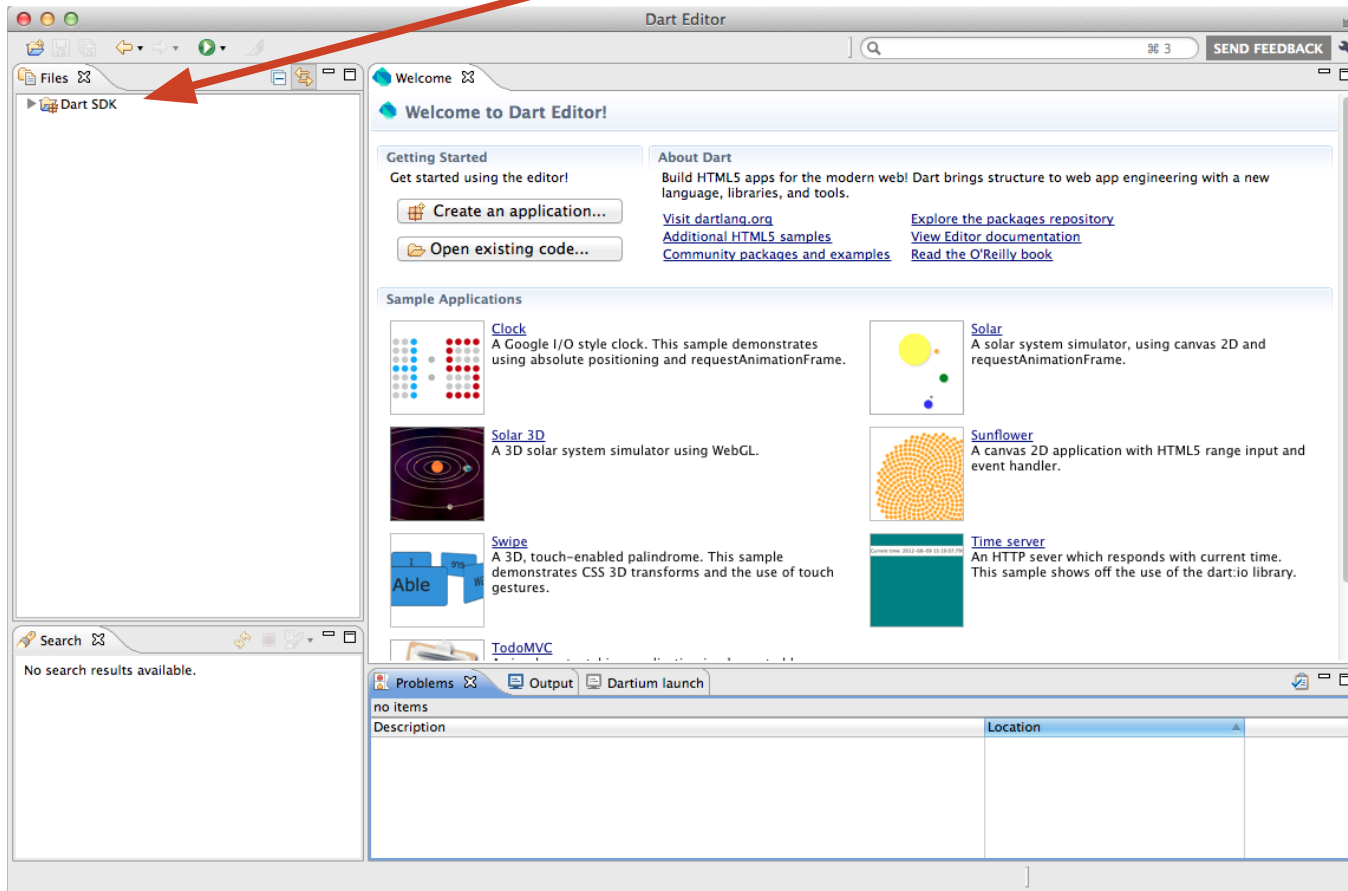
no items

Description	Location



Dart Editor

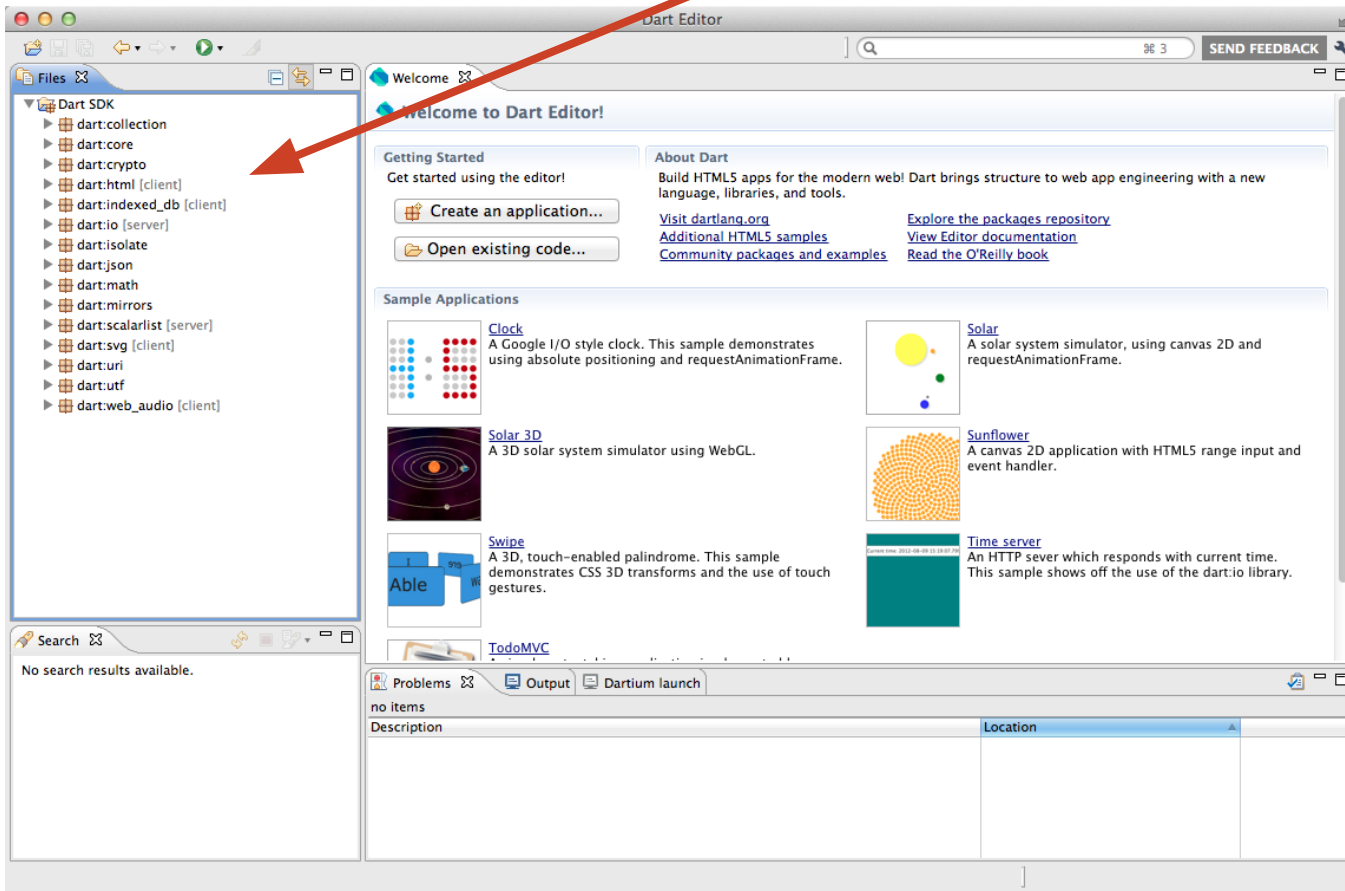
Dart SDK





Dart Editor

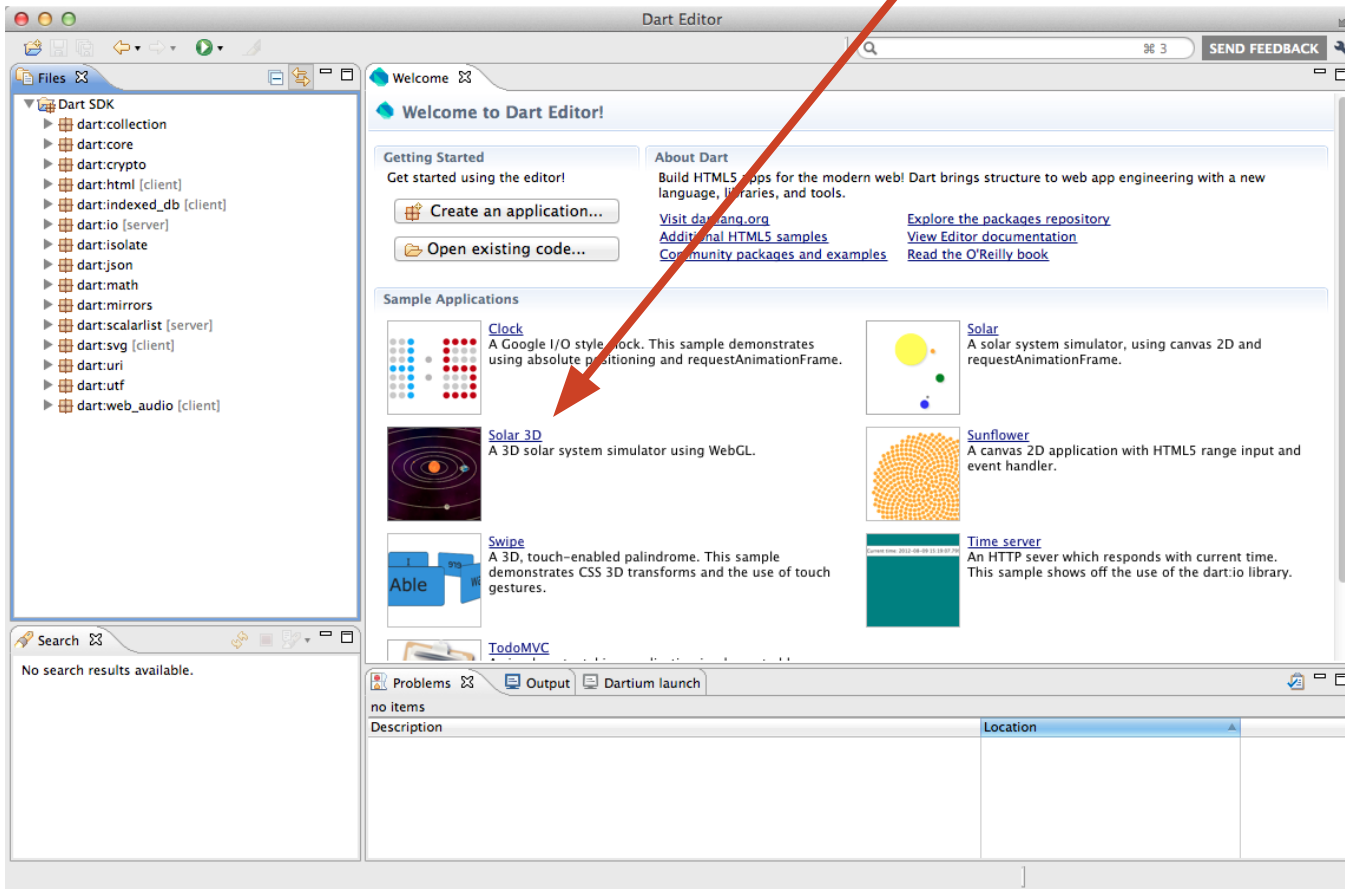
Dart SDK Libraries





Dart Editor

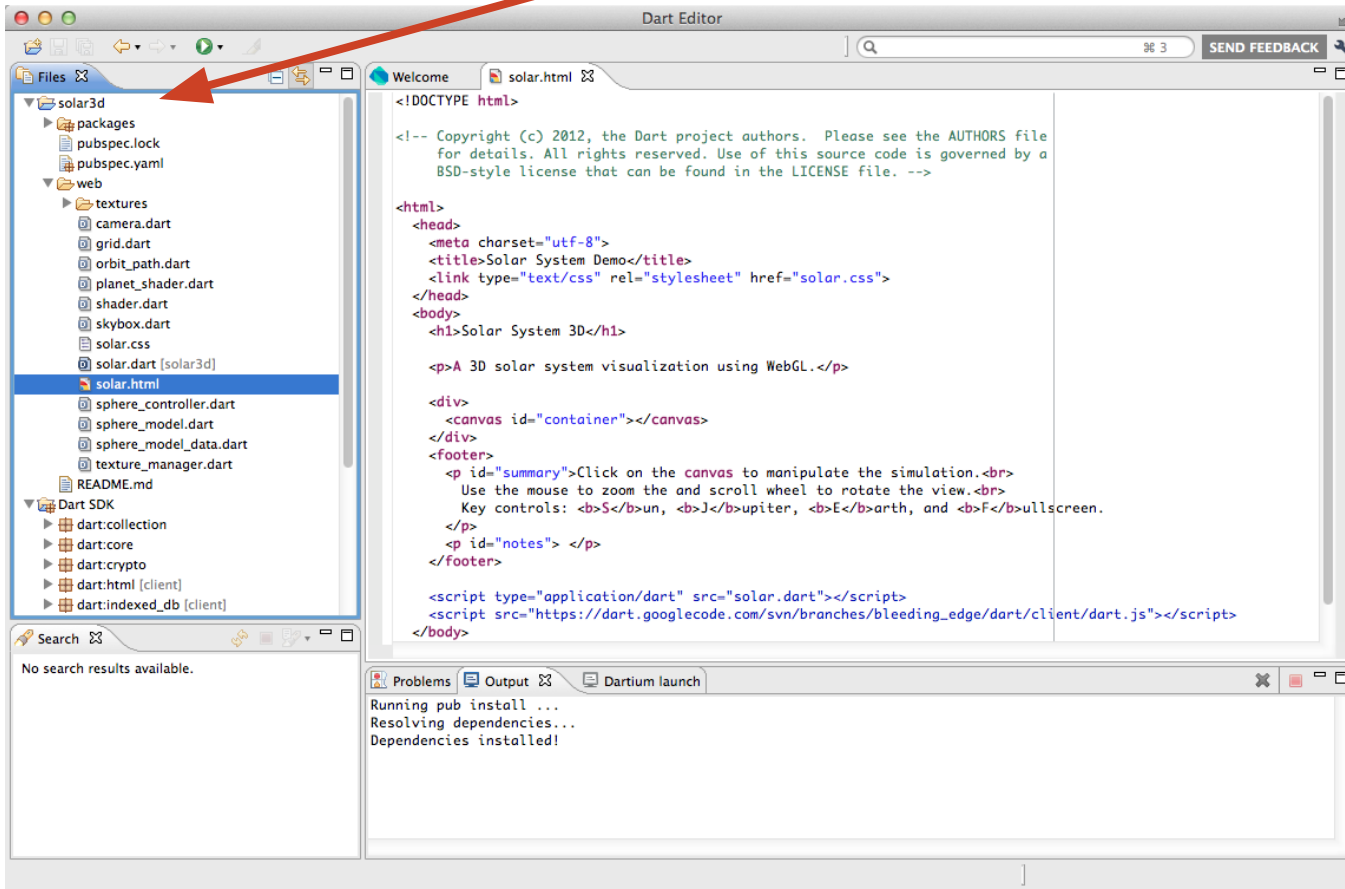
Load Solar 3D Sample





Dart Editor

Solar 3D Sample



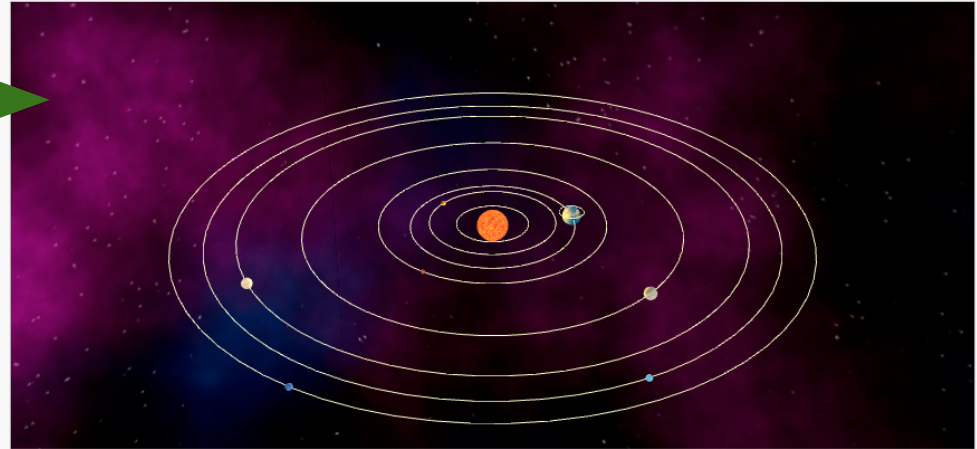
Demo

Solar 3D

- Dartium
- JavaScript
- Debugging

Solar System 3D

A 3D solar system visualization using WebGL.

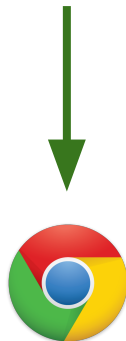


Click on the canvas to manipulate the simulation.
Use the mouse to zoom the and scroll wheel to rotate the view.
Key controls: Sun, Jupiter, Earth, and Fullscreen.

67 fps

Source

dart2js





HTML + Dart

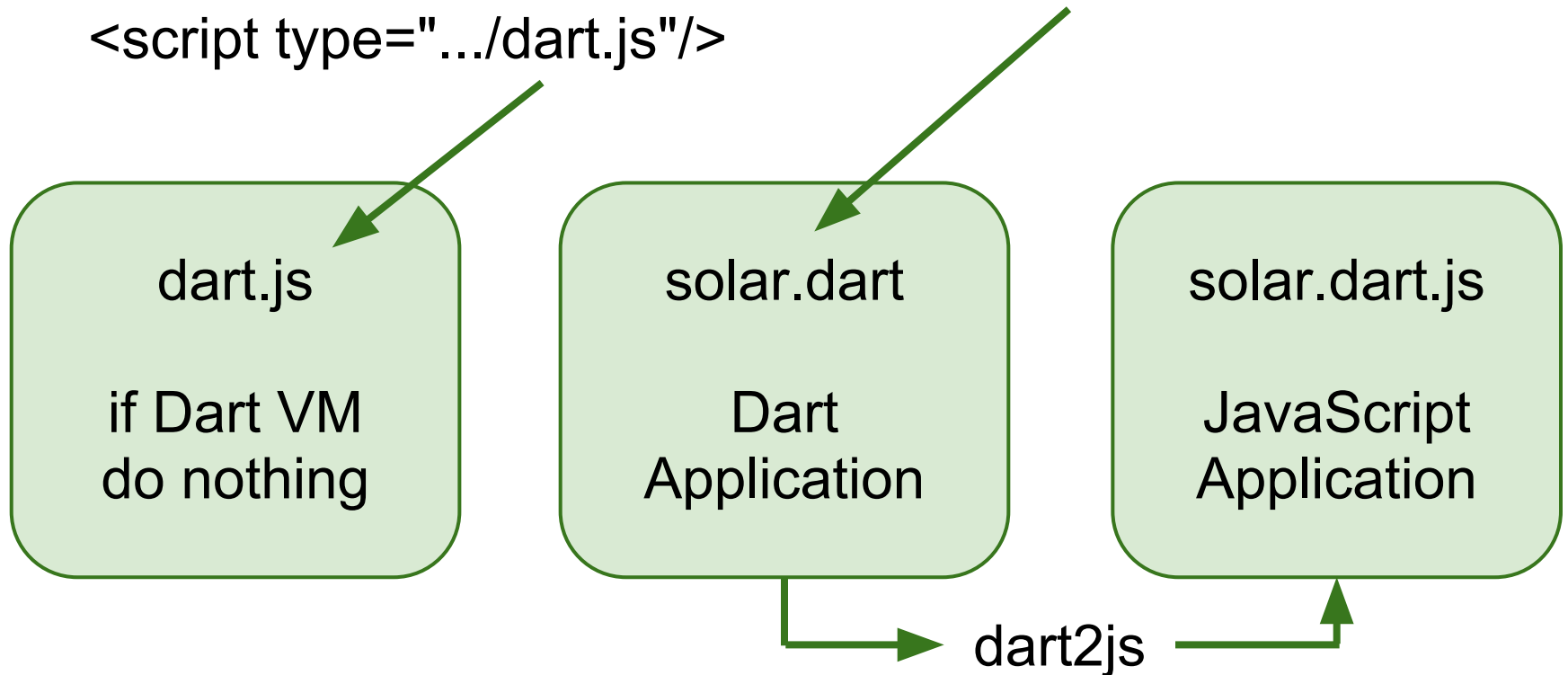


```
<html>
```

```
<body>
```

```
<script type="application/dart" src="solar.dart"/>
```

```
<script type=".../dart.js"/>
```





HTML + Dart



Others

```
<html>
```

```
<body>
```

```
<script type="application/dart" src="solar.dart.js"/>
```

```
<script type=".../dart.js"/>
```

dart.js

if no Dart VM
rewrite DOM

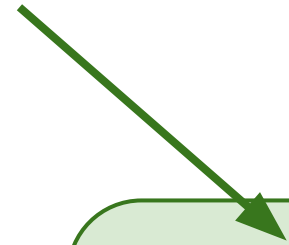
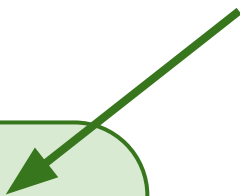
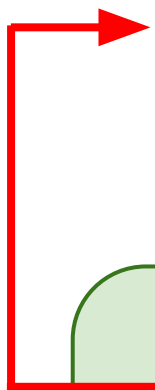
solar.dart

Dart
Application

solar.dart.js

JavaScript
Application

dart2js





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 install` installs dependencies
- `pub update` updates dependencies

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



Pub - pubspec.yaml

```
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"
  web_components: ">=0.2.8+4 <0.2.9"
```



Pub - layout

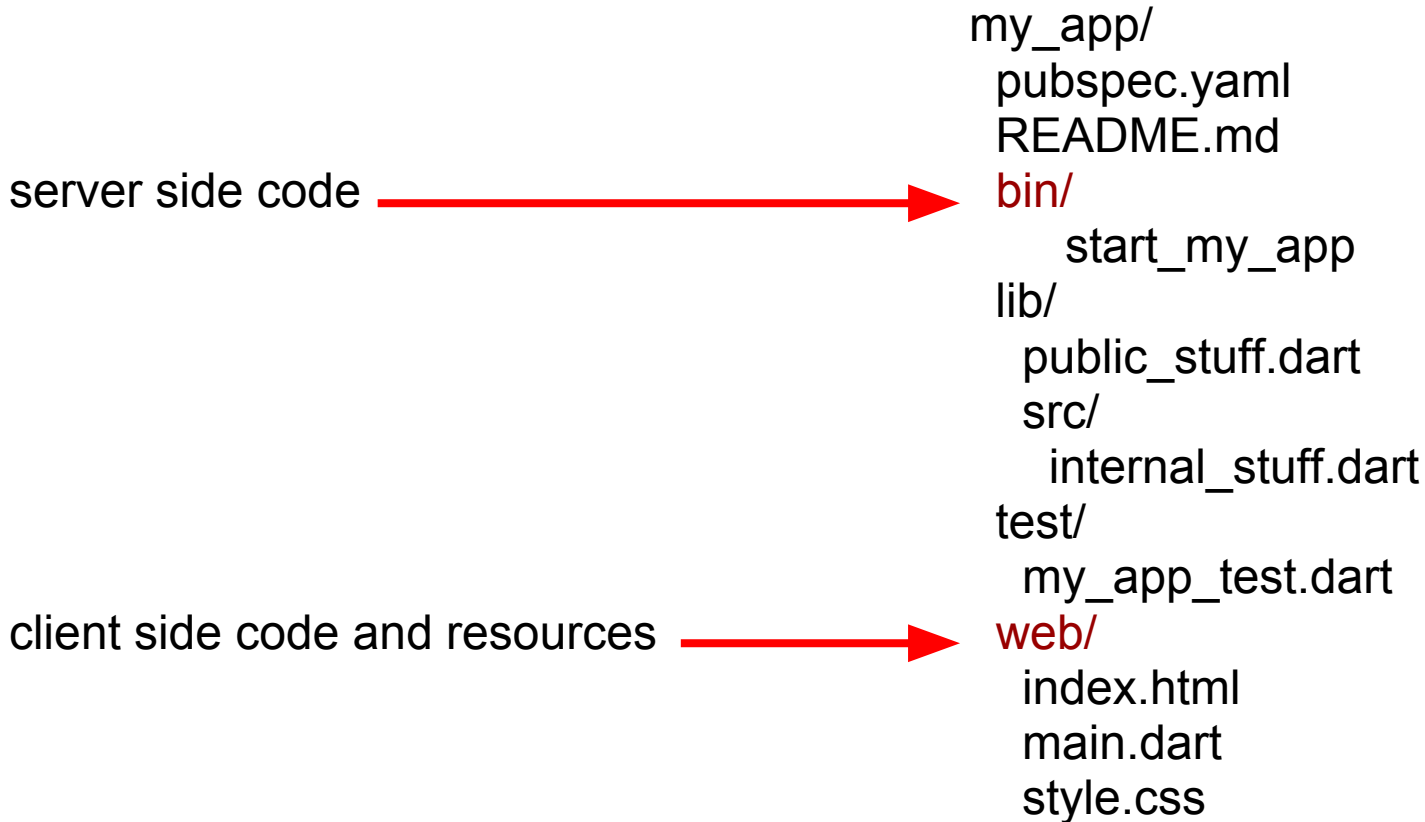
package declaration



```
my_app/  
pubspec.yaml  
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
```

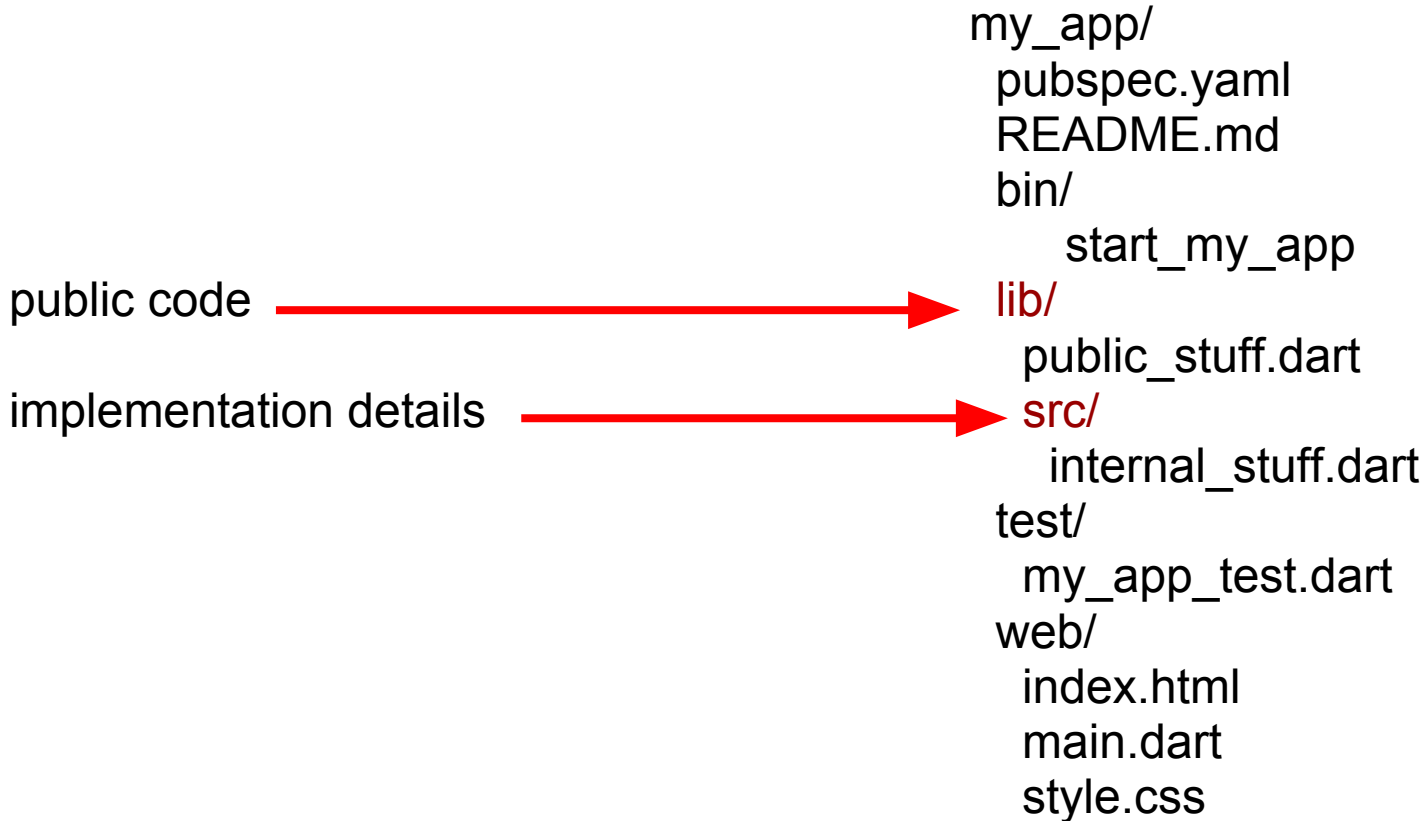


Pub - layout





Pub - layout





Pub - layout

```
my_app/  
pubspec.yaml  
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
```

tests





Server-side DART



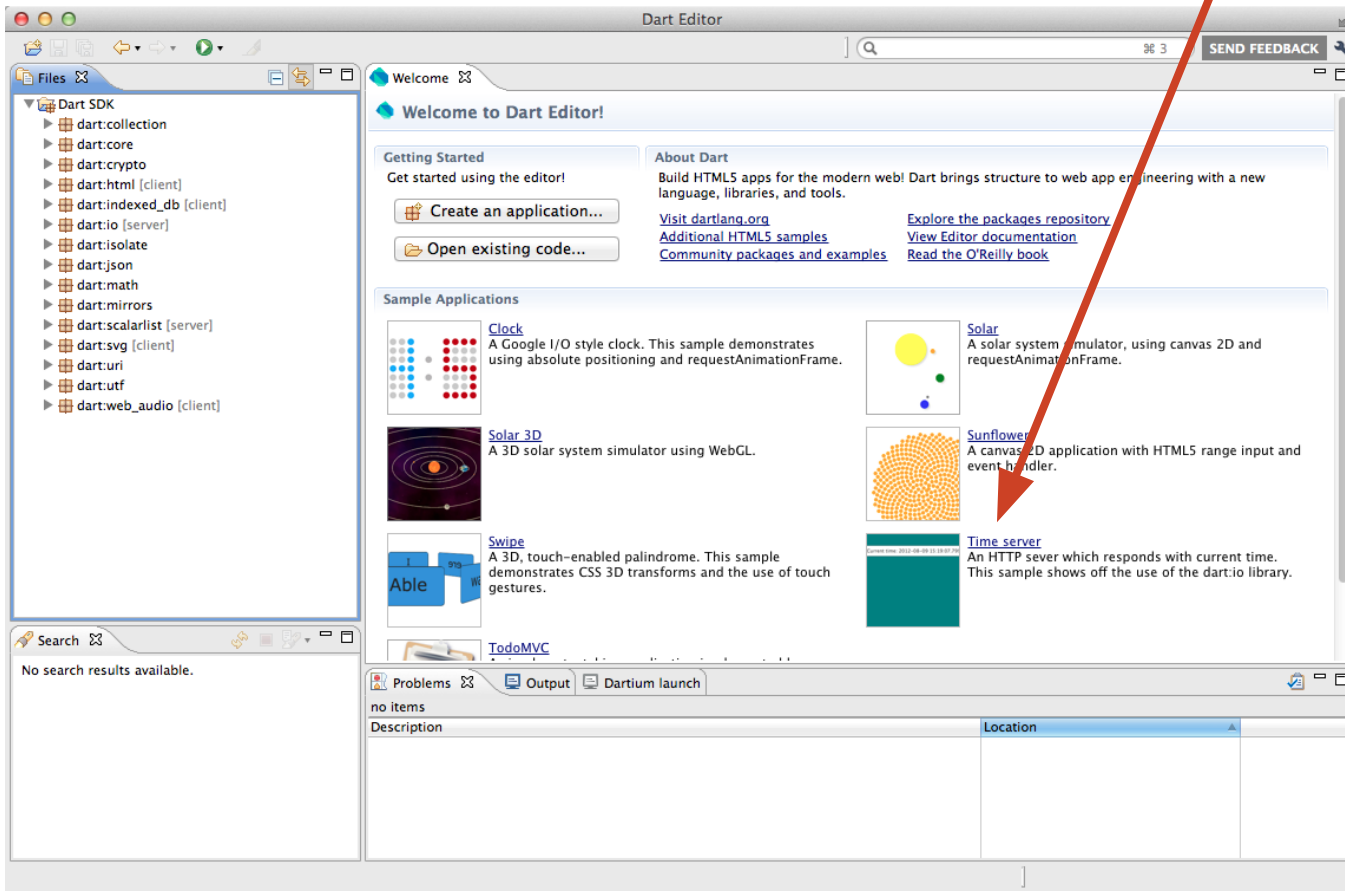
Dart on the server

- File system
- Sockets
- HTTP server and client
- Web sockets server and client
- Async or Future style
- Share code on client and server



Dart on the server

Time Server Sample





Dart on the server

Time Server Sample

The screenshot shows the Dart Editor interface with the following components:

- Files Panel:** Shows a project structure with folders like 'solar3d' and 'time'. The file 'time_server.dart [time_server]' is selected.
- Code Editor:** Contains the source code for 'time_server.dart'.

```
// Copyright (c) 2012, the Dart project authors. Please see the AUTHORS file
// for details. All rights reserved. Use of this source code is governed by a
// BSD-style license that can be found in the LICENSE file.

library time_server;

import "dart:io";
import "dart:utf";

const HOST = "127.0.0.1";
const PORT = 8080;

const LOG_REQUESTS = true;

void main() {
  HttpServer server = new HttpServer();

  server.addRequestHandler((HttpRequest request) => true, requestReceivedHandler);

  server.listen(HOST, PORT);

  print("Serving the current time on http://${HOST}:${PORT}.");
}

void requestReceivedHandler(HttpRequest request, HttpResponse response) {
  if (LOG_REQUESTS) {
    print("Request: ${request.method} ${request.uri}");
  }

  String htmlResponse = createHtmlResponse();
  List<int> encodedHtmlResponse = encodeUtf8(htmlResponse);
```
- Problems Panel:** Shows the execution output:

```
time_server.dart [Dart command-line launch] /time/time_server.dart
dart --enable-checked-mode time_server.dart

Serving the current time on http://127.0.0.1:8080.
Request: GET /
Request: GET /favicon.ico
```

A red arrow points from the 'Time Server Sample' title to the 'time_server.dart' file in the Files panel.



Dart on the server

Create server

```
void main() {  
  HttpServer server = new HttpServer();  
  server  
    ..addRequestHandler((HttpRequest request) => true, handler)  
    ..listen(HOST, PORT);  
  print("Serving the current time on http://{HOST}:{PORT}.");  
}  
void handler(HttpRequest request, HttpResponse response) {  
  // process request ... send response  
}
```



Dart on the server

Add multiple handlers

```
void main() {  
  HttpServer server = new HttpServer();  
  server  
    ..addRequestHandler((HttpRequest request) => true, handler)  
    ..listen(HOST, PORT);  
  print("Serving the current time on http://{HOST}:{PORT}.");  
}  
void handler(HttpRequest request, HttpResponse response) {  
  // process request ... send response  
}
```




Dart on the server

Matcher



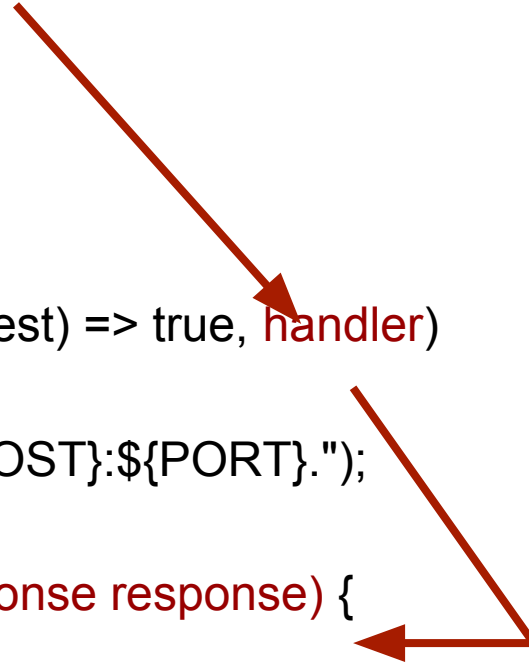
```
void main() {  
  HttpServer server = new HttpServer();  
  server  
    ..addRequestHandler((HttpRequest request) => true, handler)  
    ..listen(HOST, PORT);  
  print("Serving the current time on http://{HOST}:{PORT}.");  
}  
void handler(HttpRequest request, HttpResponse response) {  
  // process request ... send response  
}
```



Dart on the server

```
void main() {  
  HttpServer server = new HttpServer();  
  server  
    ..addRequestHandler((HttpRequest request) => true, handler)  
    ..listen(HOST, PORT);  
  print("Serving the current time on http://{HOST}:{PORT}.");  
}  
void handler(HttpRequest request, HttpResponse response) {  
  // process request ... send response  
}
```

Handler





Dart on the server

Start handling requests

```
void main() {  
  HttpServer server = new HttpServer();  
  server  
    ..addRequestHandler((HttpRequest request) => true, handler)  
    ..listen(HOST, PORT);  
  print("Serving the current time on http://{HOST}:{PORT}.");  
}  
void handler(HttpRequest request, HttpResponse response) {  
  // process request ... send response  
}
```



Dart on the server

Process requests

```
void main() {  
  HttpServer server = new HttpServer();  
  server  
    ..addRequestHandler((HttpRequest request) => true, handler)  
    ..listen(HOST, PORT);  
  print("Serving the current time on http://{HOST}:{PORT}.");  
}  
void handler(HttpRequest request, HttpResponse response) {  
  // process request ... send response  
}
```



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 Editor - Goals

- Easy on-ramp to learn Dart
- Simplified UI

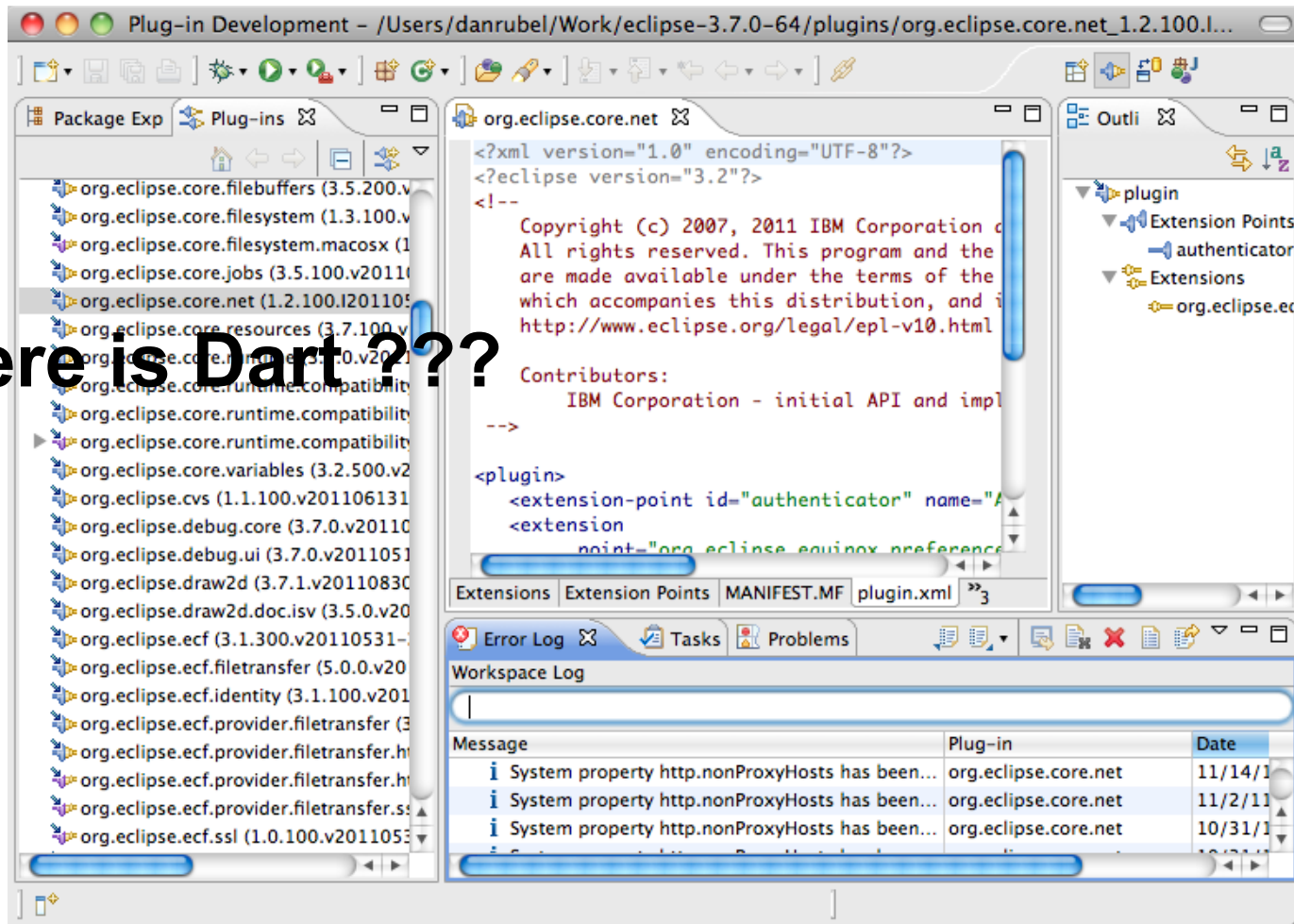
but also...

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



Dart Editor - Before

Where is Dart???



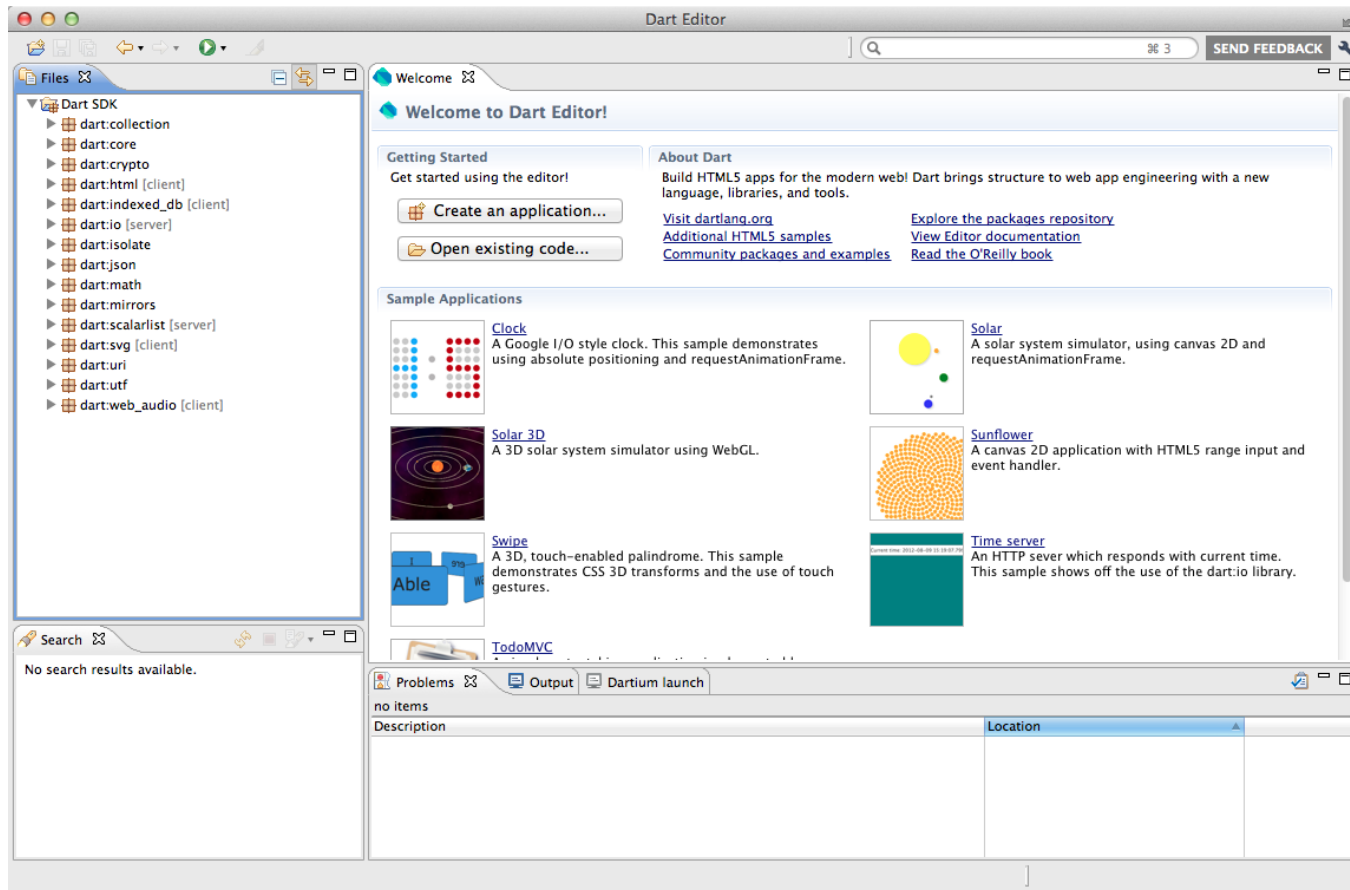


Dart Editor - Strategy

- **Narrow the scope**
 - Focus on doing a few things well
- **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

- Background indexing
- Code completion - how to make it fast
- Compilation time via incremental compilation



Metrics

First RCP build

65 MB

170 plugins

20s startup

Current build

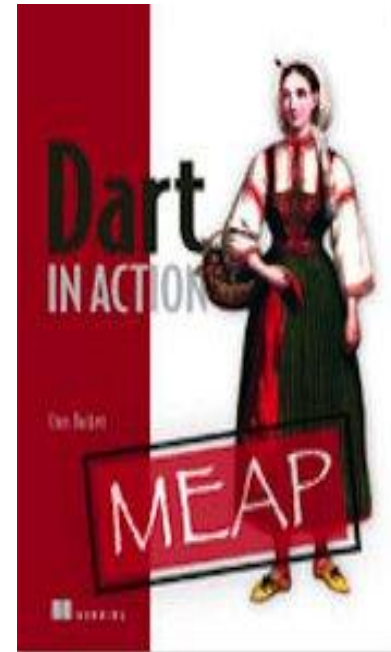
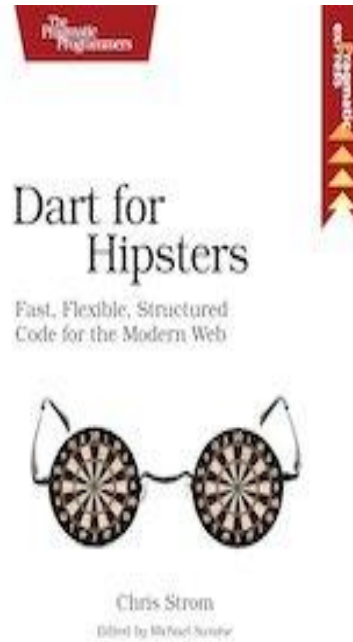
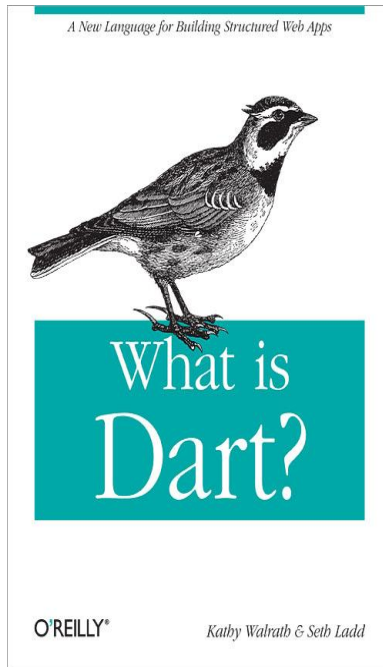
39 MB

75 plugins

5s startup



Books





Questions?

More information....

[**https://dartlang.org**](https://dartlang.org)

Introduction, language spec, articles

Download Dart Editor

[**https://code.google.com/p/dart/**](https://code.google.com/p/dart/)

Source code to editor, compiler, and virtual machine

See the wiki for instructions



Community

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