Structured web programming

An introduction to Dart

Seth Ladd
Developer Advocate
@sethladd / +Seth Ladd

#dartlang

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

-- Our goal

#dartlang

Agenda

- Introduction
- Motivations
- Dart language
- Dart runtimes
- Dart tools
- Community and the future

The Dart project

Dart is Open Source

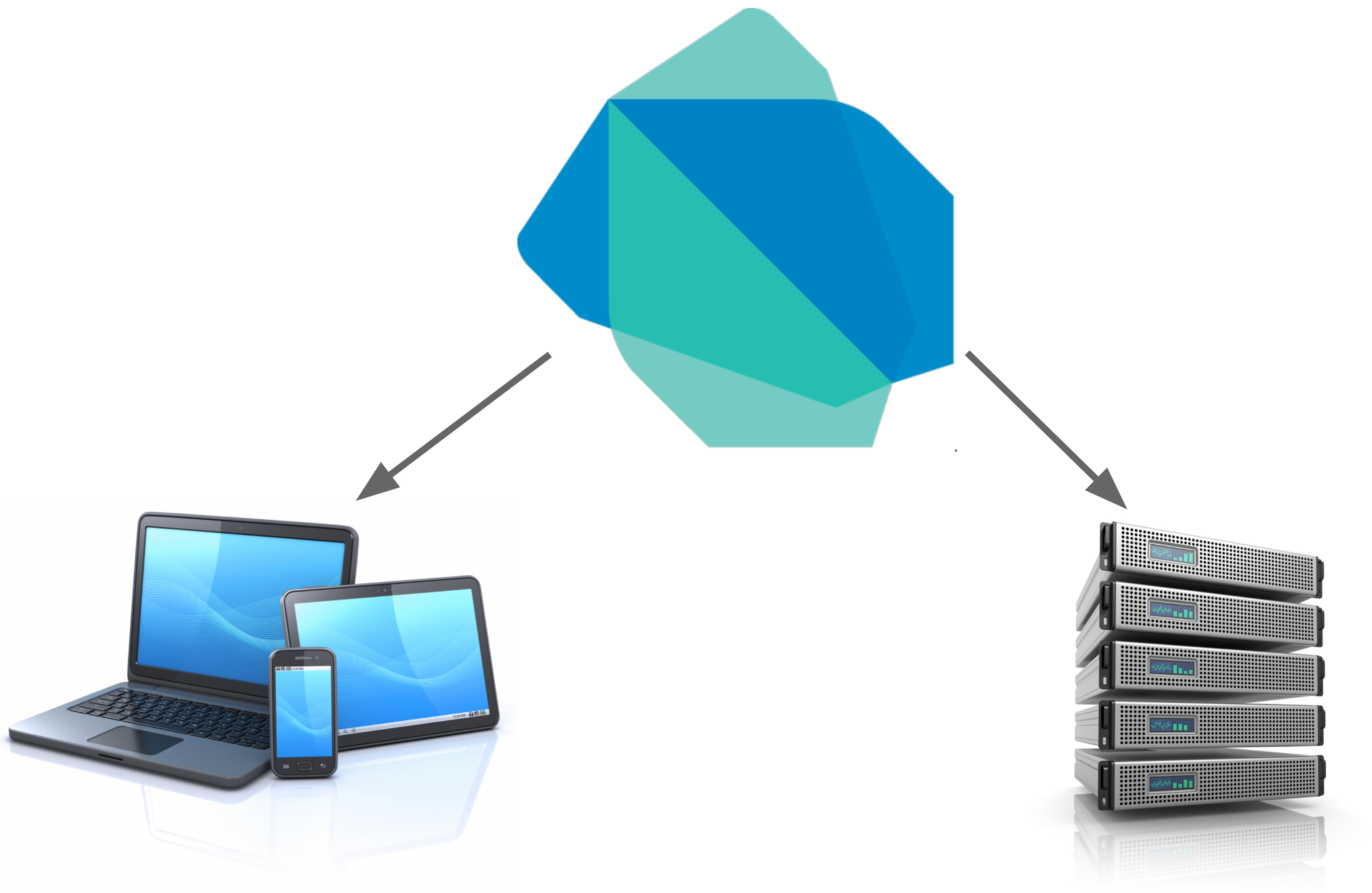
- BSD-style license
- dart.googlecode.com
  - GitHub mirror
- Contributing guide

open source initiative™

Dart comes with "batteries included"

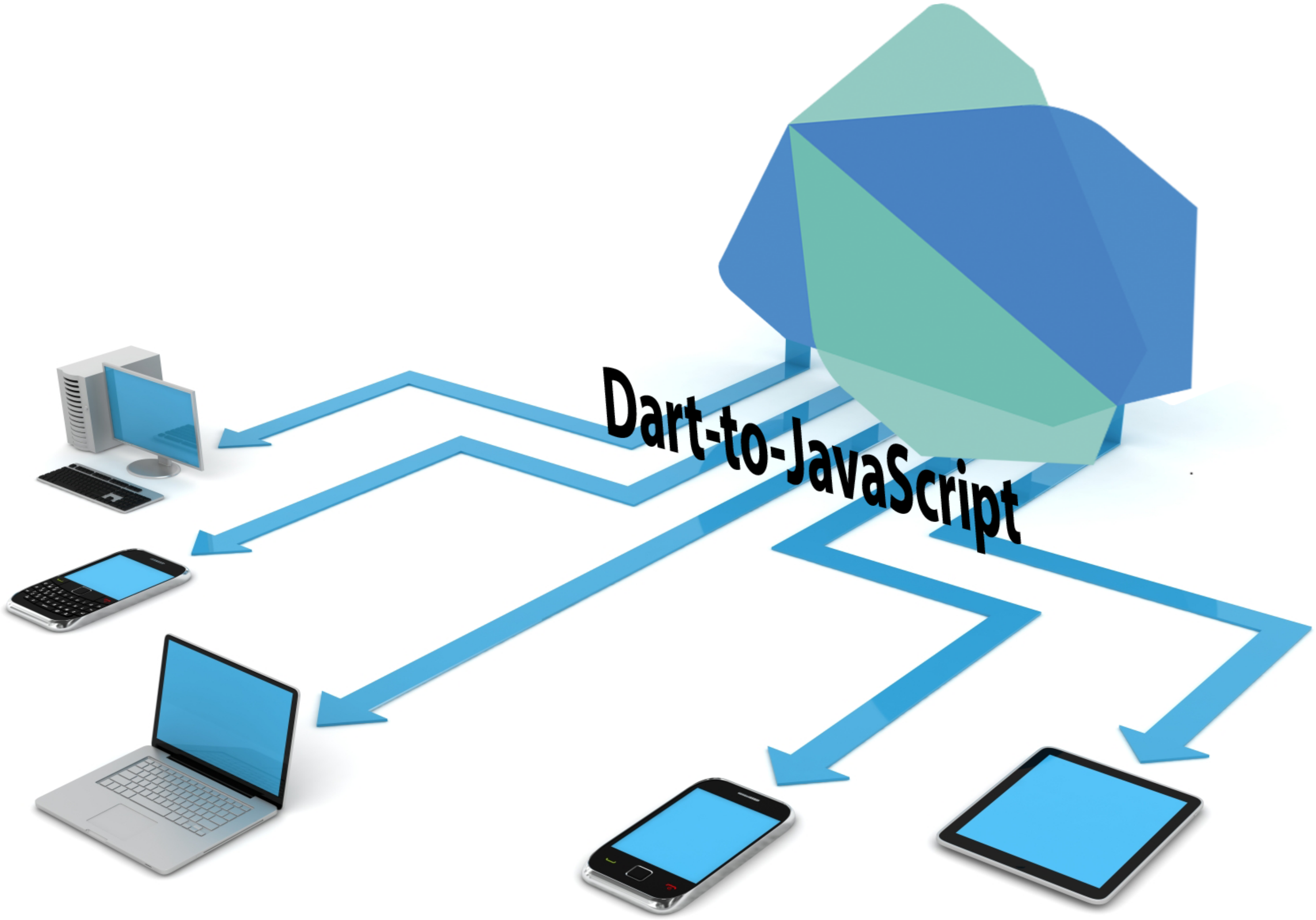
- Language
- Libraries
- Virtual machine
- Dart Editor
- Browser integration
- Compiler to JavaScript

Dart runs on the client *and* server



#dartlang

Dart targets the entire modern web



Dart is for modern web apps

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

#dartlang

Dart is Technology Preview

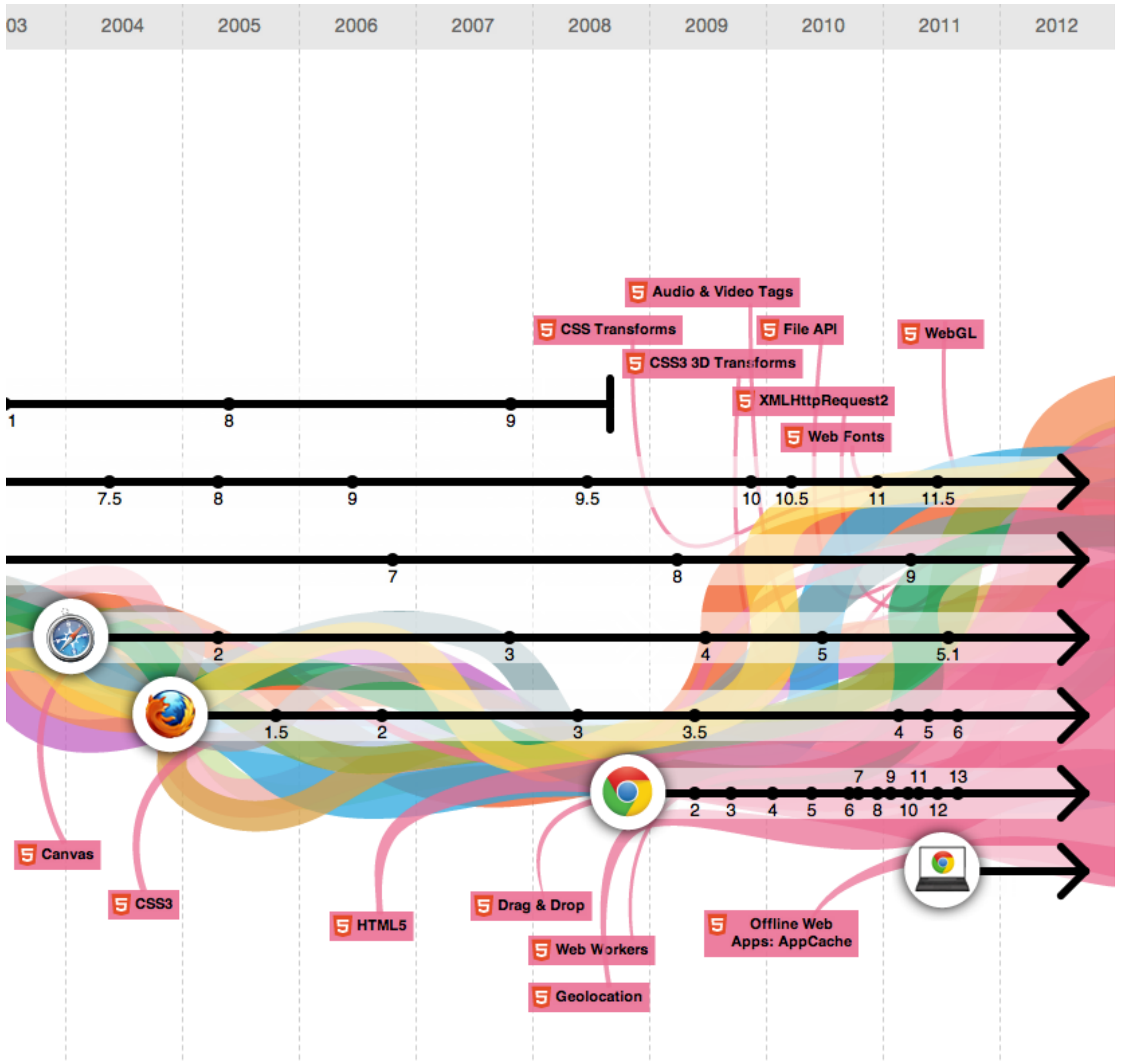
- Still building out the platform
- Some changes ahead
- Your feedback counts!

http://www.flickr.com/photos/38605191@N05/4328361364

Motivations

Web development is *good*

- Iterative development
- Platform independent
- Evolving fast
- Getting faster
- Modern browsers on the rise



#dartlang

But, it should be easier to:

- Understand program structure
- Build better tool support
- Start up faster
- Integrate code across frameworks
- Work with larger teams
- Meet user demands
- Sidestep 15 years of cruft

Innovation is Essential

- Dart fills a vacuum
- Dart provides an option
- Non-endemic developers should also build for the web

The Dart language

You can: Learn Dart quickly

- Class-based, single inheritance, OO language
- Interfaces
- Optional static typing
- Real lexical scoping
- Single threaded
- Familiar syntax

Variables, lists, iteration, final, string interpolation

Dart

```
main() {
  var fruits = ['apples', 'bananas', 'oranges'];
  for (final fruit in fruits) {
    print('I like to eat $fruit');
  }
}
```

Functions, function aliases, and closures

Dart

```
typedef num adder(num);  // alias

adder makeAdder(num n) {
  return (num i) => n + i;
}

main() {
  adder add2 = makeAdder(2);
  print(add2(3)); // 5
}
```

Hard to share code today

```
// MooTools
var Cat = new Class({
  initialize: function(name){
    this.name = name;
  }
});
```

```
// Prototype
var Person = Class.create({
  initialize: function(name) {
    this.name = name;
  }
});
```

```
// Dojo
declare("mynamespace.MyClass", null, {
  // Custom properties and methods here
});
```

```
// ExtJS
Ext.define('Ext.Window', {
  extend: 'Ext.Panel',
  requires: 'Ext.Tool'
});
```

You can: Structure and share code

Dart

```
class Point {
  num x, y;

  Point(this.x, this.y);

  num distanceTo(Point other) {
    var dx = x - other.x;
    var dy = y - other.y;
    return Math.sqrt(dx * dx + dy *
dy);
  }
}
```

Dart

```
interface Widget {
  Element parent;
  render();
}

// From framework A
class Panel implements Widget {
  ...
}

// From framework B
class Menu extends Panel {
  ...
}
```

#dartlang

You can: Reason about unfamiliar code more easily

JS

```
function recalculate(origin, offset, estimate) {
  ...
}
```

Dart

```
num recalculate(Point origin, num offset,
      [bool estimate=false]) {
  ...
}
```

#dartlang

Optional static type annotations

- Dart supports duck typing, but sometimes we want:
  - inline documentation
  - easier to understand code for developers
  - machines to give us early errors and warnings
- Helps you scale from small idea to large app

Demo time. To the Dart Editor!

You can: rely on 'this' not changing

Dart

```
#import('dart:html');

class Awesome {
  Awesome(Element button) {
    button.on.click.add((e) => cool()); // which cool?
                                        // lexically scoped cool!
  }
  cool() {
    window.alert("ice cold");
  }
}

main() {
  new Awesome(document.query("#button"));
}
```

You can: Expect sane for loops

Dart

```
main() {
  var callbacks = [];
  for (var i = 0; i < 2; i++) {
    callbacks.add(() => print(i));
  }

  callbacks.forEach((c) => c());
}

// 0
// 1
```

You can: Create readable method calls

Dart

```
flipFlags(bool on, bool up, bool hidden) {}

// wha??
flipFlags(true, false, true);

// turn on named args, default values
flipFlags([bool on, bool up, bool hidden=false]) {}

// ahh... better!
flipFlags(on: true, up: false, hidden: true);

// optional args!
flipFlags(on: true, up: false); // hidden == false
```

Hard to write deeply async code

JS

```
// Yikes! This will lock the page by running too
// many long processes in the main UI thread.
button.on.click.add((e) {
  costlyQuery();
  expensiveWork();
  lengthyComputation();
  print("done!");
});
```

Hard to write deeply async code

JS

```
// Nasty nested callbacks
button.on.click.add((e) {
  costlyQuery(() {
    expensiveWork(() {
      lengthyComputation(() {
        print("done!");
      });
    });
  });
});
```

You can: Handle callbacks with Futures

JS

```
// Using futures
 button.on.click.add((e) {
   // Each function returns a Future
   costlyQuery()
   .chain((value) => expensiveWork())
   .chain((value) => lengthyComputation())
   .then((value) => print("done!"));
 });
```

You can: Write concurrent apps

Dart

```
#import('dart:isolate');

echo() {
  port.receive((msg, SendPort replyTo) {
    replyTo.send("I received: $msg");
  });
}

main() {
  SendPort echoPort = spawnFunction(echo);
  echoPort.call("Hello from main").then((replyMsg) {
    print(replyMsg);    // I received: Hello from main
  });
}
```

Dart isolates

- Inspired by Erlang processes
- Isolated memory heaps
- No shared state
- Communicate with message passing
- Can run on separate thread or process
- Compile to Web workers for HTML5 apps

You can: Still write dynamic code

Dart

```
class Ninja {
  equip(weapon) { .. }
}

class Pirate {
  equip(weapon) { .. }
}
```

Dart

```
armForBattle(warrior) {
  // duck typing at work
  warrior.equip(new SuperSword());
}

main() {
  armForBattle(new Ninja());
  armForBattle(new Pirate());
}
```

#dartlang

You can: Get more dynamic-er

Dart

```
class JsonObject {
  Map properties;
  JsonObject(String json) {
    properties = JSON.parse(json);
  }
  noSuchMethod(String functionName, List args) {
    // translate functionName to property name, get from Map
  }
}

main() {
  var jsonObj = new JsonObject("{'hello':'world'}");
  print(jsonObj.hello); // world
}
```

Dart on the server

#dartlang

You can: Write server apps in Dart

Dart

```
// simple web server
runServer(String basePath) {
  HttpServer server = new HttpServer();
  server.defaultRequestHandler = new StaticFileHandler(basePath).onRequest;
  server.listen('127.0.0.1', 1337);
}

main() {
  File script = new File(new Options().script);
  script.directory().then((Directory d) {
    runServer(d.path);
  });
}
```

Dart VM for server-side apps

- Files, directories
- Sockets
- HTTP server and client
- Web sockets server and client
- Async or Future style
- Share code on client and server
- Demo!

Dart on the client

You can: Use a friendlier DOM lib

Dart

```
// to the Dart Editor!!

#import('dart:html');

void main() {
  ButtonElement button = new Element.tag('button');

  button.text = 'Click me';
  button.classes.add('important');

  button.on.click.add((e) => window.alert('Clicked!!'));

  document.body.elements.add(button);
}
```

Compile Dart to JavaScript

- dart2js compiler in SDK and Dart Editor
- Targets ES5 (modern browsers)
- Tree shaking and dead code elimination
- Written in Dart
- In progress:
    - smaller JavaScript output
    - performance improvements

The Dart Editor and Dartium

You can: run Dart apps directly in Chromium

- Dartium == Chromium + Dart VM
- Bundled in Dart Editor download
- Great for development and debugging
    - Fast edit/reload cycles
    - Dev Tools integration

You can use the Dart Editor to:

- Jump to definition
- Perform simple refactorings such as renaming methods and variables
- Debug code
- Run Dart in Dartium
- Compile Dart to JavaScript

Package management

You can install 3rd party packages

- pub is the Dart package manager
- Clones remote package repos
- Manages depedencies
- Coming soon
  - pub.dartlang.org for discovery and publishing

Using pub

1) Dart

```
#library('catapp');
#import('dart:html');
#import('package:catpic/catpic.
 dart');
#import('package:frame/frame.dart');
#import('package:widget/widget.
 dart');
```

2) pubspec

```
dependencies:
  catpic:
    git: git://github.com/munificent/catpic.git
  frame:
    git: git://github.com/munificent/frame.git
  widget:
    git: https://bitbucket.org/munificent/widget.
git
```

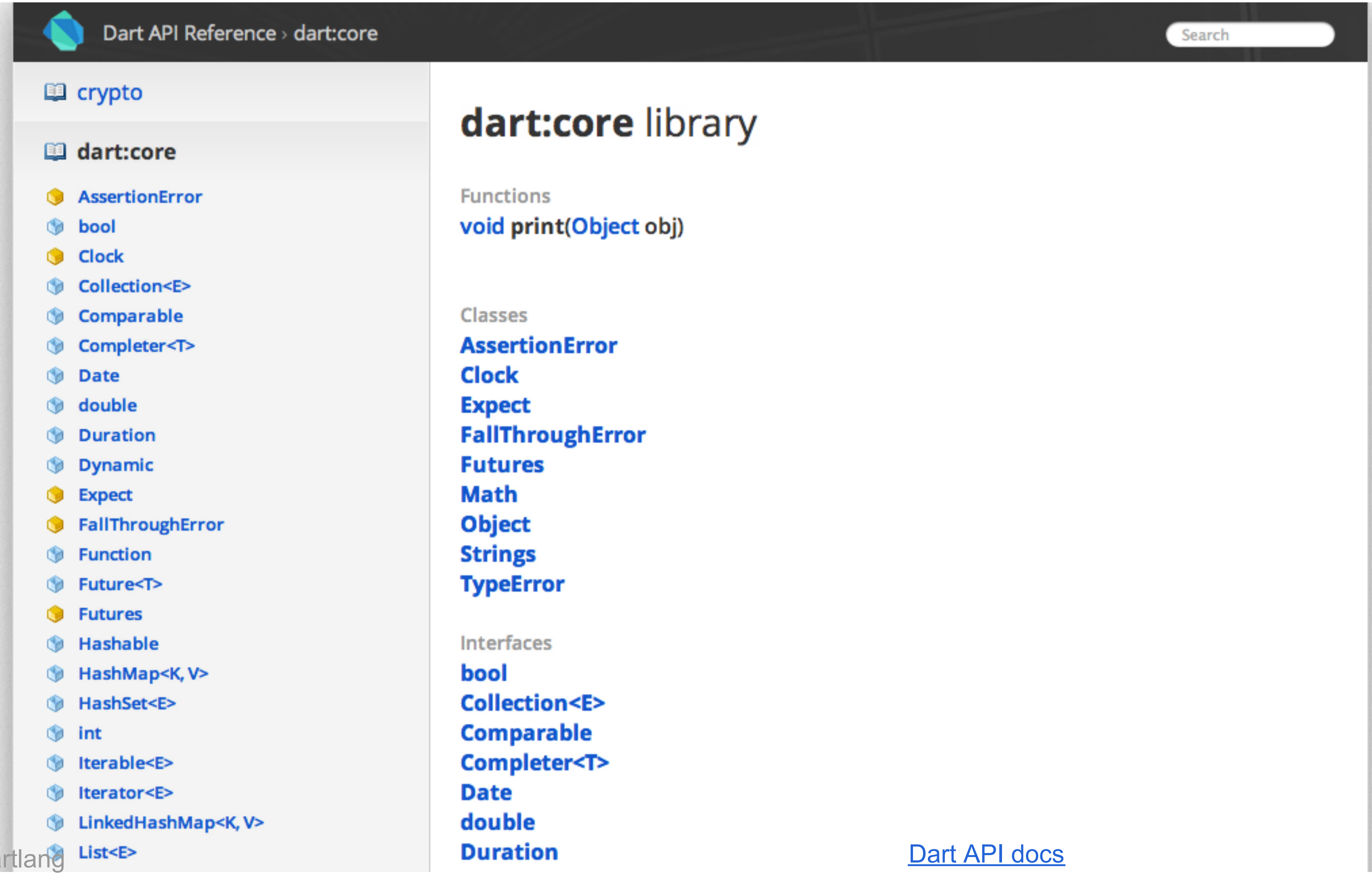
3) Command line

```
> pub install
  .. cloning libs ..
  Dependencies installed!
```

4) Deploy kittens! (demo)

Learn more

#dartlang



Dart API Reference › dart:core

Search

crypto

dart:core

- AssertionError
- bool
- Clock
- Collection<E>
- Comparable
- Completer<T>
- Date
- double
- Duration
- Dynamic
- Expect
- FallThroughError
- Function
- Future<T>
- Futures
- Hashable
- HashMap<K, V>
- HashSet<E>
- int
- Iterable<E>
- Iterator<E>
- LinkedHashMap<K, V>
- List<E>

dart:core library

Functions

void print(Object obj)

Classes

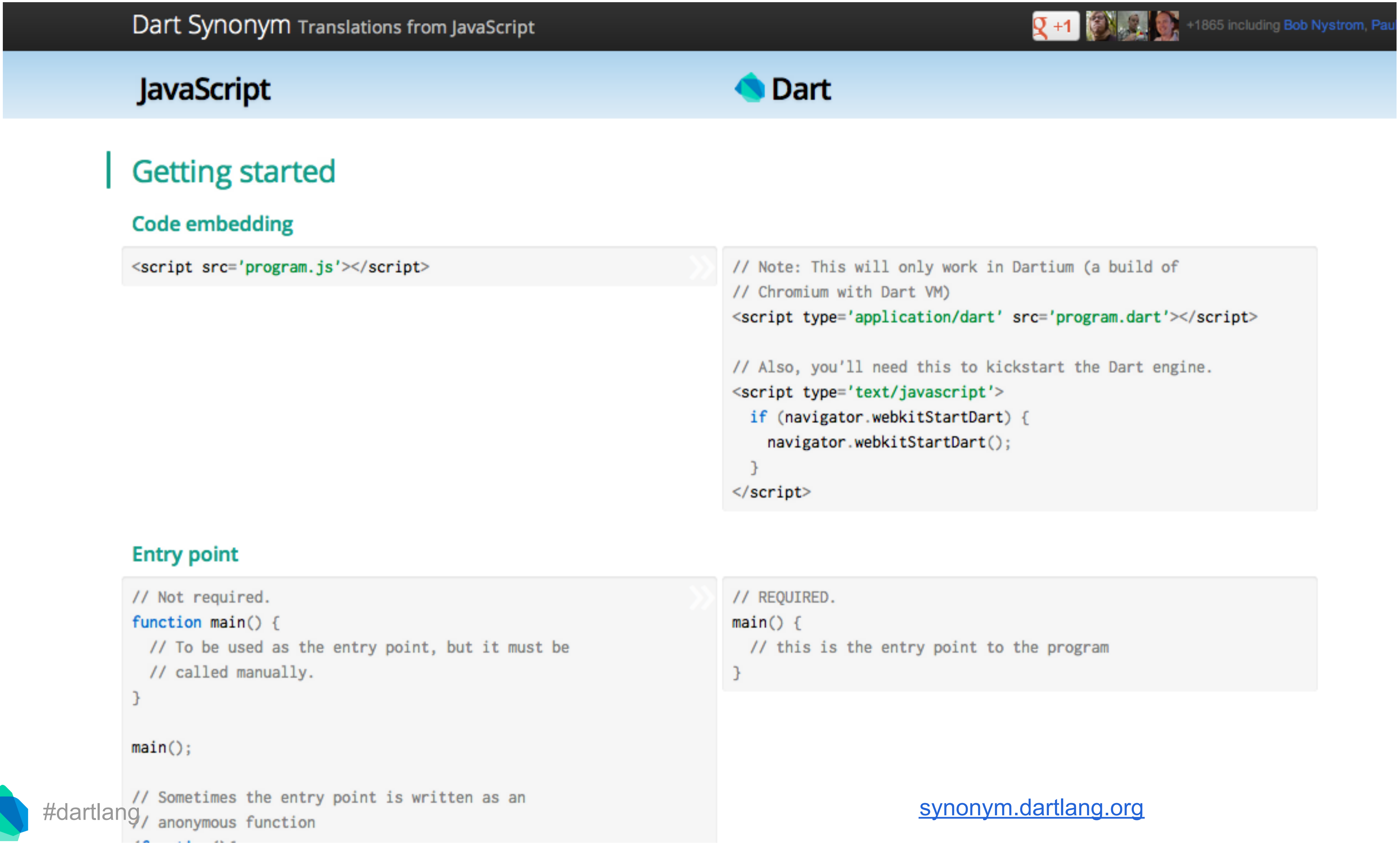
AssertionError
Clock
Expect
FallThroughError
Futures
Math
Object
Strings
TypeError

Interfaces

bool
Collection<E>
Comparable
Completer<T>
Date
double
Duration

#dartlang

Dart API docs



Dart Synonym Translations from JavaScript

JavaScript | Dart

Getting started

Code embedding

JavaScript:

```
<script src='program.js'></script>
```

Dart:

```
// Note: This will only work in Dartium (a build of
// Chromium with Dart VM)
<script type='application/dart' src='program.dart'></script>

// Also, you'll need this to kickstart the Dart engine.
<script type='text/javascript'>
  if (navigator.webkitStartDart) {
    navigator.webkitStartDart();
  }
</script>
```

Entry point

JavaScript:

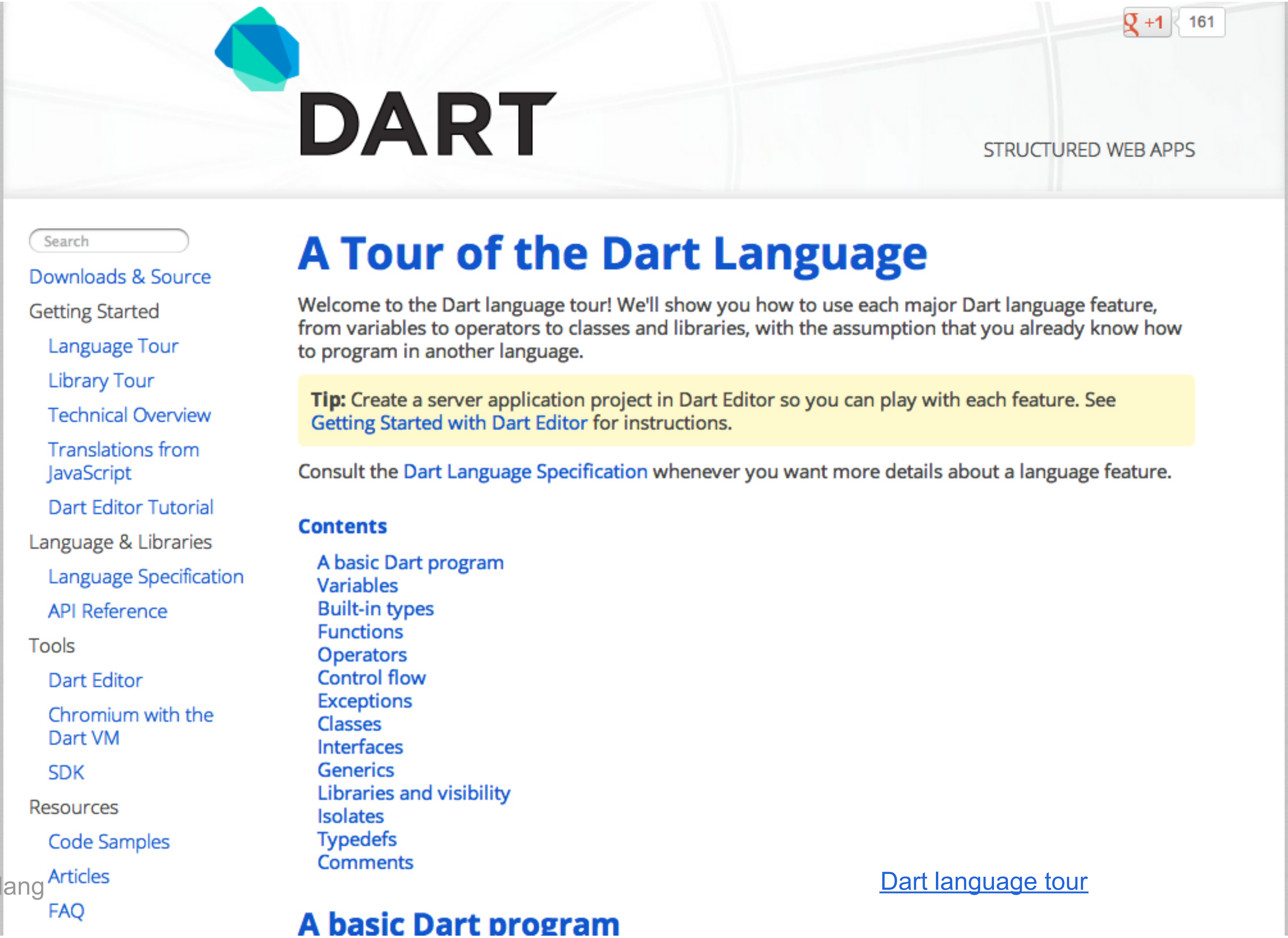
```
// Not required.
function main() {
  // To be used as the entry point, but it must be
  // called manually.
}

main();

// Sometimes the entry point is written as an
// anonymous function
```

Dart:

```
// REQUIRED.
main() {
  // this is the entry point to the program
}
```

#dartlang

synonym.dartlang.org



DART

STRUCTURED WEB APPS

+1 161

Search

Downloads & Source

Getting Started

Language Tour

Library Tour

Technical Overview

Translations from JavaScript

Dart Editor Tutorial

Language & Libraries

Language Specification

API Reference

Tools

Dart Editor

Chromium with the Dart VM

SDK

Resources

Code Samples

Articles

FAQ

A Tour of the Dart Language

Welcome to the Dart language tour! We'll show you how to use each major Dart language feature, from variables to operators to classes and libraries, with the assumption that you already know how to program in another language.

Tip: Create a server application project in Dart Editor so you can play with each feature. See Getting Started with Dart Editor for instructions.

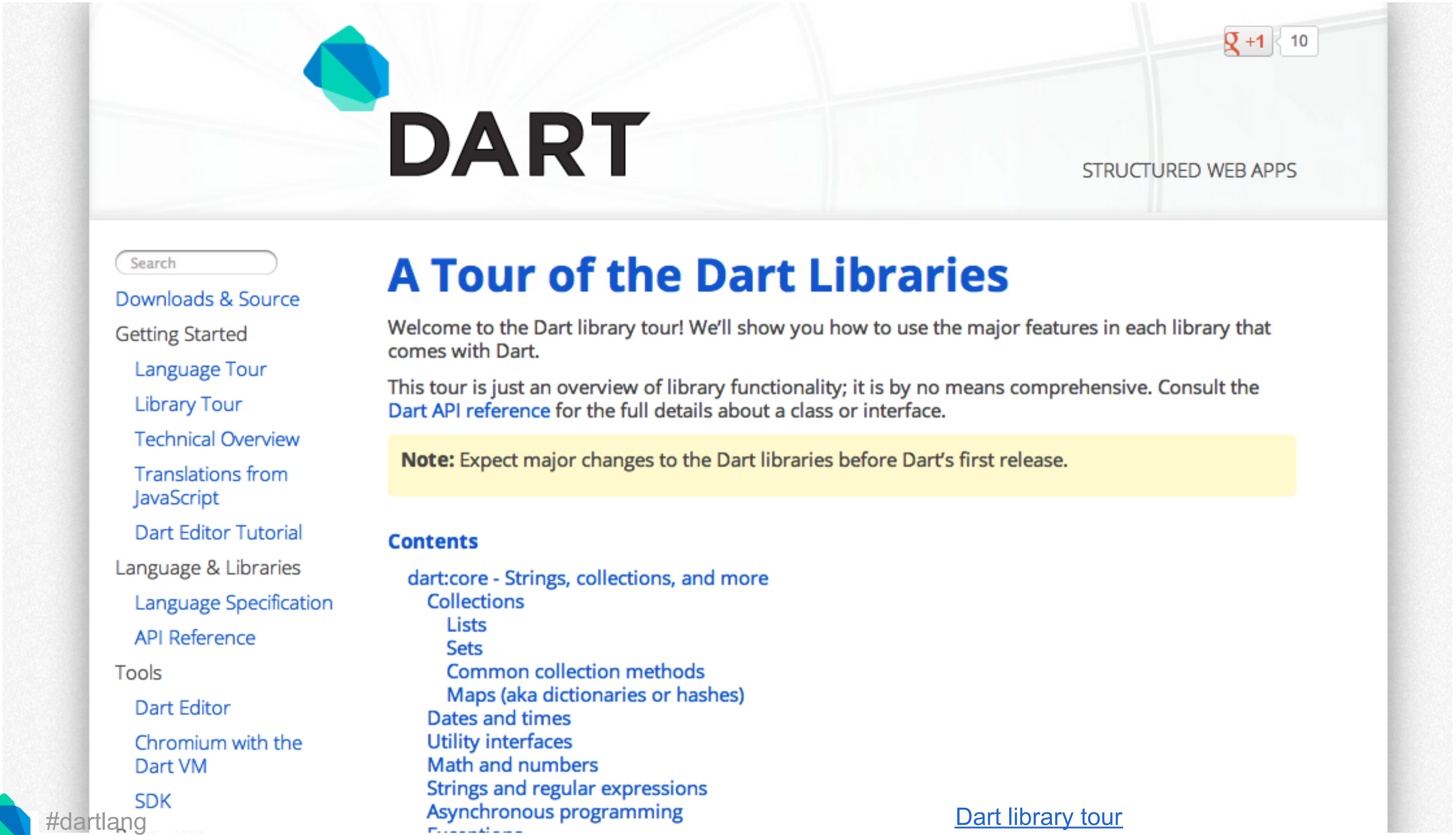
Consult the Dart Language Specification whenever you want more details about a language feature.

Contents

A basic Dart program

#dartlang

[Dart language tour](#)



+1 10

DART

STRUCTURED WEB APPS

Search

Downloads & Source

Getting Started

Language Tour

Library Tour

Technical Overview

Translations from JavaScript

Dart Editor Tutorial

Language & Libraries

Language Specification

API Reference

Tools

Dart Editor

Chromium with the Dart VM

SDK

A Tour of the Dart Libraries

Welcome to the Dart library tour! We'll show you how to use the major features in each library that comes with Dart.

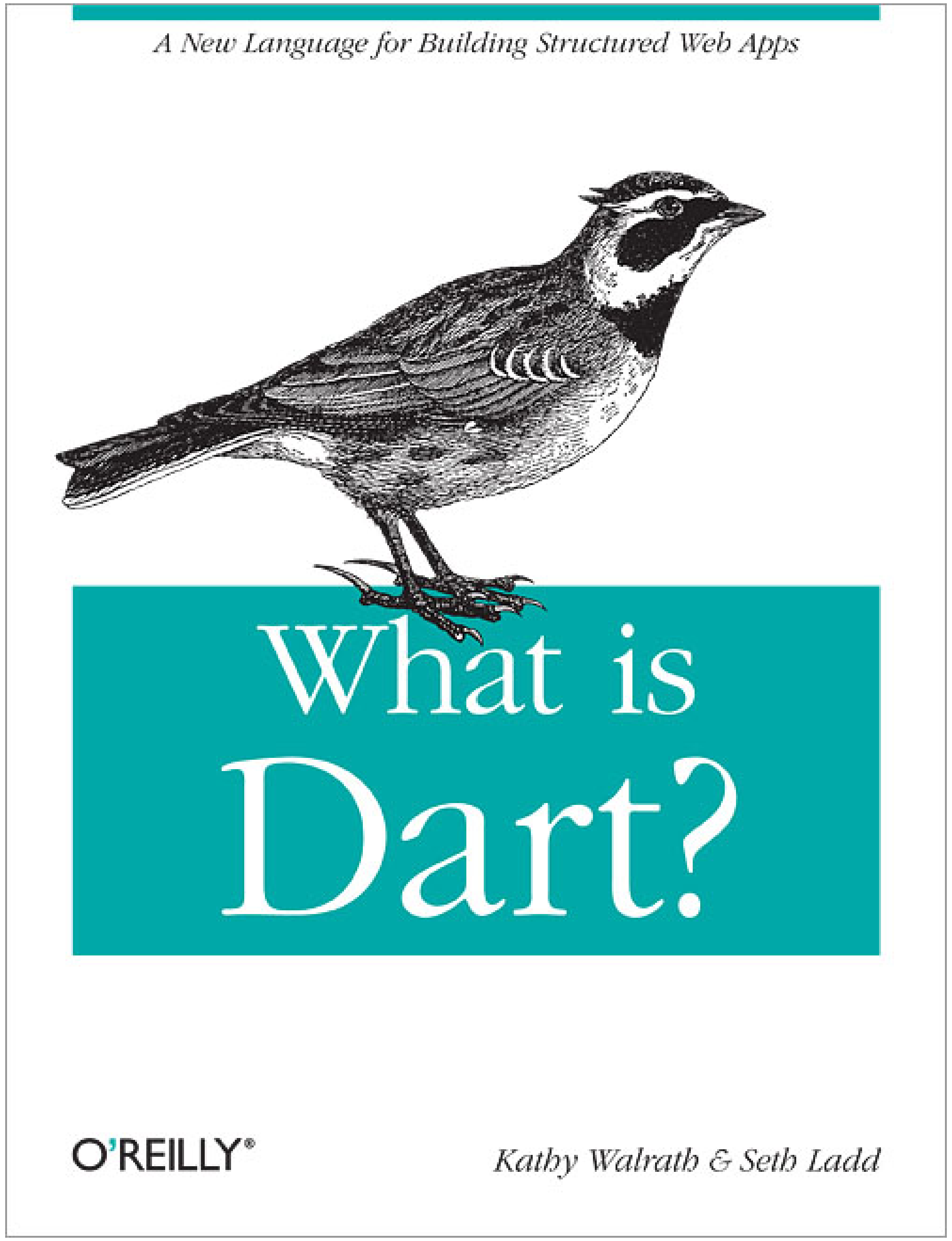
This tour is just an overview of library functionality; it is by no means comprehensive. Consult the Dart API reference for the full details about a class or interface.

Note: Expect major changes to the Dart libraries before Dart's first release.

Contents

#dartlang

Dart library tour



A New Language for Building Structured Web Apps

What is Dart?

O'REILLY®

Kathy Walrath & Seth Ladd

#dartlang

Join the community

#dartlang

Third party Dart libraries

- Dart-crypto
- PureMVC
- Buckshot UI
- Logging
- Offline
- Vector math
- Box2D
- Nintendo emulator
- JSONP
- Mustache templates
- Cordova/Phonegap
- Three.dart
- MongoDB driver
- Redis driver
- AWS libs
- Z-machine
- Flash DisplayList
- mod_dart for Apache
- Dart on Heroku
- Many more...

Join the Dart conversation

- Twitter: @dart_lang
- G+: +Dart: Structured web apps
- Hashtag: #dartlang
- Blogs: http://dartosphere.org
- IRC: #dart
- Mailing list: misc@dartlang.org
- Stack Overflow: Tag dart

Get started today!

- Like rich code editors?
  - Download the **Dart Editor** bundle
- Like vi/emacs/Sublime?
  - Download the standalone **SDK**
- Learn Dart at **dartlang.org** and **api.dartlang.org**
- Send feedback!

The future

Dart is not done

- In progress
  - Reflection support
  - Simplifications to equality
  - Package manager
  - Method cascades
  - Shipping Dart in Chrome
  - UI libs for apps
- In discussion
  - class mixins
  - more...

Summary

Today you learned that Dart:

- Compiles to modern JavaScript
- Is easy to learn
- Has type annotations that fit your style
- Helps you avoid common web programming puzzlers
- Runs on the client and the server
- Ships an editor
- Is in active development
- Has an active community

Dart is structured web programming compatible with today's web.

Please try it and give us feedback!

dartlang.org

#dartlang

Thank You!

Try Dart today at ***dartlang.org***

@sethladd

+Seth Ladd

FluentConf office hours : 1:45pm, see you there!

#dartlang

Types, from Closure to Dart

```
// Closure compiler code

/**
 * @param {String} name
 * @return {String}
 */
makeGreeting = function(name) {
  /** @type {String} */
  var greeting = 'hello ' + name;
  return greeting;
}
```

```
// Dart code

String MakeGreeting(String name) {
  String greeting = 'hello $name';
  return greeting;
}
```

Isolate use cases

- Concurrency
- Security
- Mashups
- Inter-app communication
- Intra-app communication

You can: Handle Web socket connections

Dart

```
void main() {
  HttpServer server = new HttpServer();
  WebSocketHandler wsHandler = new WebSocketHandler();
  server.addRequestHandler((req) => req.path == "/ws", wsHandler.onRequest);

  wsHandler.onOpen = (WebSocketConnection conn) {
    conn.onMessage = (message) {
      conn.send("Echo: $message");
    };

    conn.onClosed = (int status, String reason) {
      print('closed with $status for $reason');
    };
  };

  server.listen('127.0.0.1', 8000);
}
```

You can: Run in browsers without Dart

Dart

```
<!DOCTYPE html>
<html>
  <body>
    <script type="application/dart" src="app.dart"></script>
    <script type="text/javascript"
       src="http://dart.googlecode.com/svn/bleeding_edge/client/dart.js"
>
    </script>
  </body>
</html>
```

1. Checks if Dart VM exists. If not:
   a. Dart script is removed and replaced with JS script
2. Starts program when DOMContentLoaded fires

You can: Be more specific with collections

Dart

```
List<String> fruits = <String>['apples', 'oranges'];

assert(fruits is List<String>);

~~fruits.add(42);~~ // static warning, runtime
exception!
```