Berlin GTUG

Dart: a modern web language

Florian Loitsch
Google

DART

Who am I?

Florian Loitsch, software engineer at Google

Projects

- **Scheme2Js** - Scheme-to-JavaScript compiler
- **Js2scheme** - JavaScript-to-Scheme compiler
- **V8** - high-performance JavaScript virtual machine
- **Dart** - structured programming for the web

DART

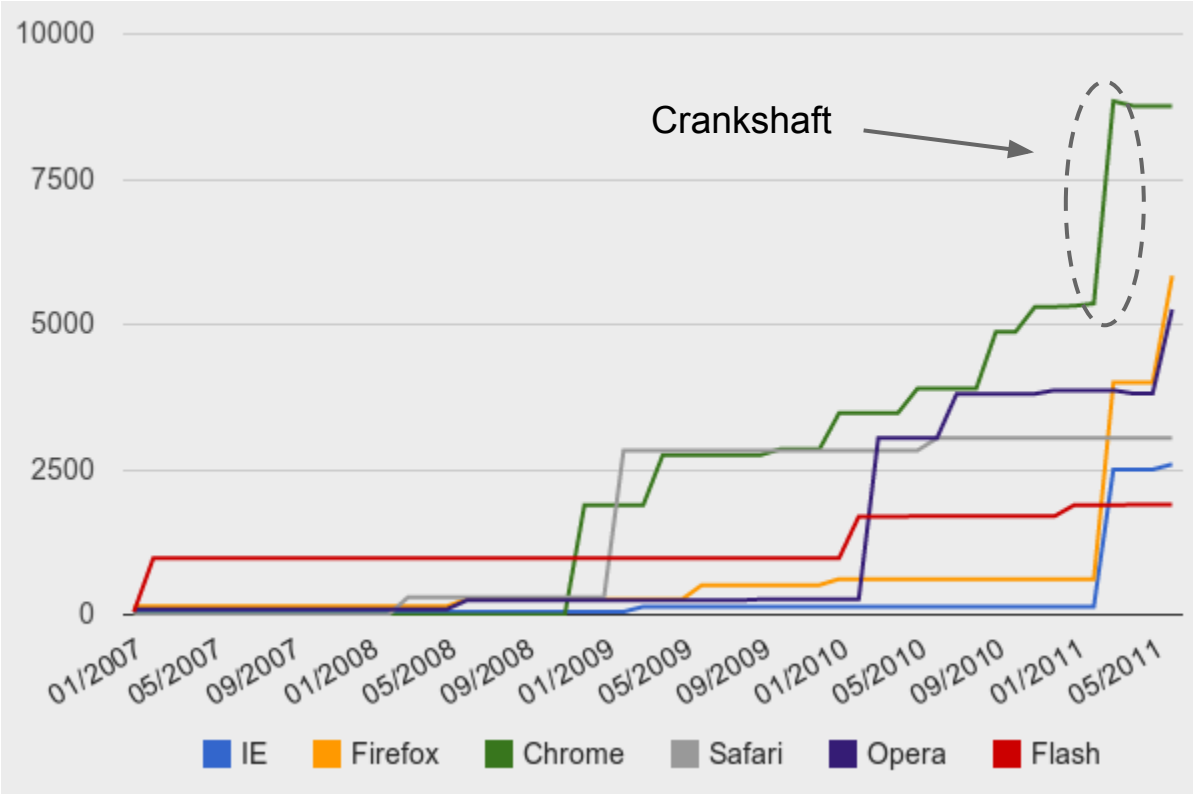
Motivation

Improve web development

The web is already pretty awesome

- It is easy to develop small applications
  - Code runs everywhere (phones, desktops)
  - No installation of applications
  - Deployment is almost trivial
- JavaScript is very flexible and supports incremental development

The rise of JavaScript



Credit: http://iq12.com/blog/

DART

Why is the web hard to program for?

- Writing large well-performing applications is hard
- Hard to reason about the program structure
- Startup performance is often really bad
- Difficult to document intent (lack of types)
- No support for modules, packages, or libraries

DART

Make it easier

- We want to improve the web platform
  - Better support for programming in the large
  - Faster application startup (especially on mobile)
  - More predictable and better runtime performance
  - JavaScript is a powerful tool but it has sharp edges
- Keep up the innovation momentum
  - The web is evolving at a fantastic pace!
  - The developer tools have to keep up

DART

JavaScript is full of ... surprises

- Lots and lots of implicit type conversions
- Most operations produce weird results when passed wrong or uninitialized values instead of failing in a recognizable way

Keep on truckin'

DART

No argument type checking

```
var x = 499;
x + null;
x + [];
x + undefined;
x - {};
```

No argument type checking

```
var x = 499;
x + null;  // => 499
x + [];
x + undefined;
x - {};
```

No argument type checking

```
var x = 499;
x + null;  // => 499
x + [];  // => 499
x + undefined;
x - {};
```

No argument type checking

```
var x = 499;
x + null;  // => 499
x + [];  // => 499
x + undefined;  // => NaN
x - {};
```

No argument type checking

```
var x = 499;
x + null;  // => 499
x + [];  // => 499
x + undefined;  // => NaN
x - {};  // => NaN
```

No array bounds checking

```
var array = new Array(32);
...
array[32];
array[-1];
array[.1];
array[null];
array[array];
```

No array bounds checking

```
var array = new Array(32);
...
array[32];  // => undefined
array[-1];  // => undefined
array[.1];  // => undefined
array[null];  // => undefined
array[array];  // => undefined
```

No array bounds checking

```
var array = new Array(32);
...
array[32];  // => void 0
array[-1];  // => void 0
array[.1];  // => void 0
array[null];  // => void 0
array[array];  // => void 0
```

No spell checking?

```
var request = new XMLHttpRequest();
...
request.onreadystatechange = function() {
  if (request.readystate == 4) {
    console.log('Request done!');
  }
};
```

No spell checking?

```
var request = new XMLHttpRequest();
...
request.onreadystatechange = function() {
  if (request.readyState == 4) {
    console.log('Request done!');
  }
};
```

JavaScript has improved but ...

- JavaScript has fundamental issues at the language level that impact productivity
- Performance has improved but mostly for a pretty static subset of JavaScript
- It remains very time consuming to build and maintain large web apps

The story of Dart

- A few years ago Lars Bak and Kasper Lund prototyped Spot
  - A new simple programming language for the web
  - Based on their experiences from JavaScript
- Spot was the prelude for the Dart project

What is Dart?

- Unsurprising object-oriented programming language
- Class-based single inheritance with interfaces
- Familiar syntax with proper lexical scoping
- Single-threaded with isolate-based concurrency
- Optional static types

First code

Let's try some Dart code:

- Classes
- Closures
- Optional types

http://try.dartlang.org

Conventional type checking

- Tries to prove that your program obeys the type system
- Considers it a fatal error no proof can be constructed
- In Dart, you are innocent until proven guilty...

```
List<Apple> apples = tree.pickApples();
printFruits(apples);

void printFruits(List<Fruit> fruits) {
  for (Fruit each in fruits) print(each);
}
```

DART

Optional static types

- Static types convey the intent of the programmer
- Checkable documentation for code and interfaces
- Avoids awkward variable naming or comment schemes
- Type annotations have no effect on runtime semantics

Experiments with Types

Let's explore a few illustrative examples.

A good time to ask questions!

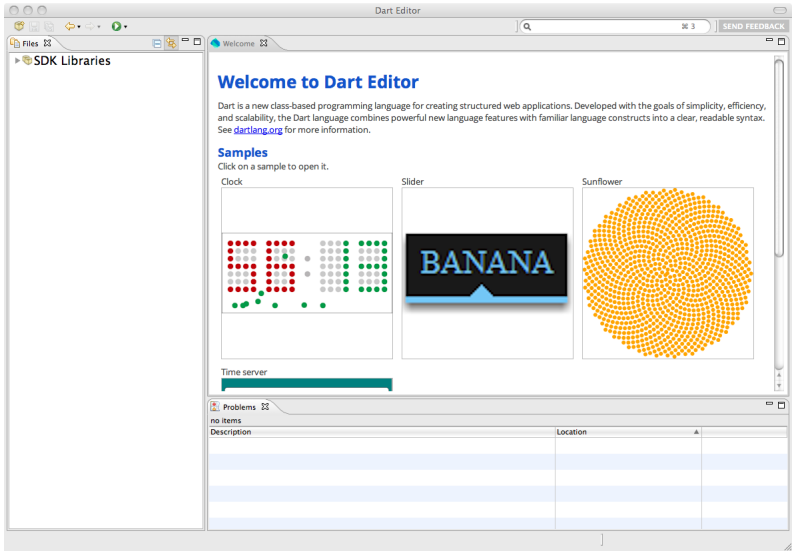
But there is more!

Dart comes with a lot of developer tools:

- DartEditor: Eclipse based Dart editor
- Dartium: Chromium with embedded Dart VM
- dart2js: Dart-to-JavaScript compiler

Let's see it in action

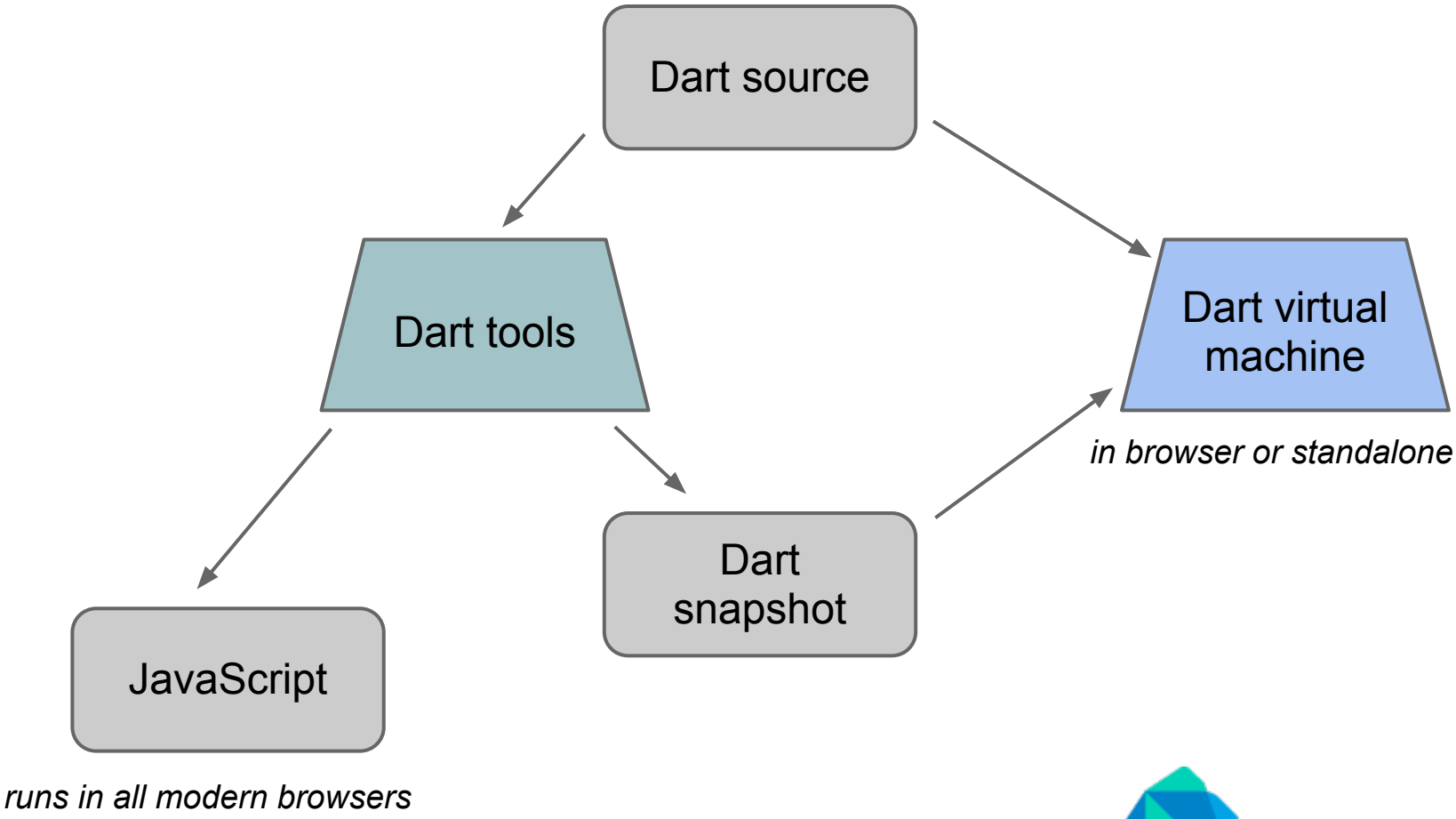
- Let's build a simple web application with the Eclipse-based Dart Editor



What did you just see?

- Immediacy through fast save-refresh cycle
- Integrated development and debugging
- Same source code runs on server and client

Deployment and execution



DART

Dart virtual machine

- Dart has been designed for performance
  - Simplicity gives more performance headroom
  - Enforced structure leads to better predictability
  - Virtual machine performs better than V8 at launch
- Works standalone or embedded in browser
  - Experimental Dart-enabled build of Chromium
  - SDK includes preliminary server-side libraries

```
$ dart hello.dart
```

Snapshots

- Snapshots contain serialized program structures
  - Cyclic graph of classes, interfaces, and statics
  - Can be read in without parsing source code
  - Improve startup performance by more than 10x
- Snapshots can be generated server-side or client-side
  - Platform independent format sent over the wire
  - Can be cached locally in browser app cache

DART

Dart-to-JavaScript

- Compiler is implemented in Dart
  - Generates JavaScript that runs in modern browsers
  - Built for future optimizations (type inferencing, etc.)
  - Uses *tree shaking* to cut down on code size

```
$ dart2js --out=hello.js hello.dart
```

Flavour of generated JavaScript

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

```
Isolate.$defineClass("Point", "Object", ["x", "y"], {
  toString$0: function() {
    return '(' + $.toString(this.x) + ',' +
                 $.toString(this.y) + ')';
  }
});
```

DART

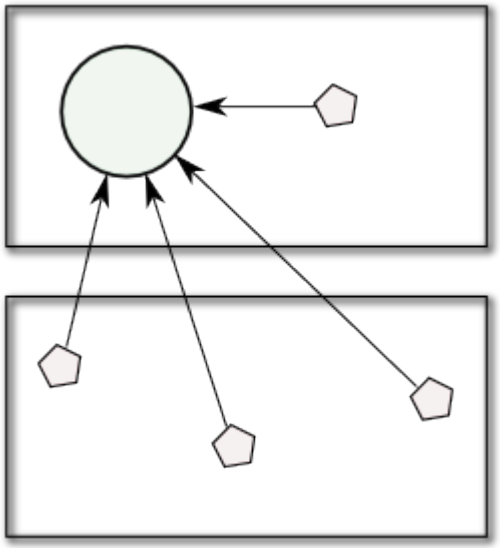
Isolates

Isolates are lightweight units of execution:

- Run in their own address space like processes
- Nothing is shared - nothing needs synchronization
- All communication takes place via messaging passing
- Supports concurrent execution

Communication

- ReceivePorts:
  - enqueues incoming messages
  - can not leave their isolate
  - can be created on demand



- SendPorts:
  - created by a ReceivePort
  - dispatches messages to its ReceivePort
  - can be transferred (across Isolate boundaries)
  - Unforgeable, transferable capability

DART

Open source

- Dart is available under a BSD license
- Developed in the open (code reviews, build bots, etc.)

Online resources

- **Primary site** - http://www.dartlang.org/
- **Code** - http://dart.googlecode.com/
- **Libraries** - http://api.dartlang.org/
- **Specification** - http://www.dartlang.org/docs/spec/

Summary

- Dart is an unsurprising, object-oriented language that is instantly familiar to most
- Dart allows you to write code that tools and programmers can reason about
- Dart applications runs in all modern browsers through translation to JavaScript

Dart was designed with performance in mind.

Dart allows rapid prototyping and structured development.

Thank you!

Dart is open source and instantly familiar to lots of programmers.

Dart runs everywhere JavaScript does.