DART

The State of D art

Gilad Bracha

Google

DART

The State of Dart

Gilad Bracha

Google

Dart

Web apps should be as good or better than native apps

- For users
- For developers
  - Sophisticated web apps need not require superhuman feats of heroism

Dart

Constraints

- Compile to Javascript on the browser
- Familiar to mainstream programmers

Dart: Language, Platform, Tools, Engine(s)

Familiar, unsurprising language

```
class Point {
  var x, y;
  Point(a, b){x =a; y = b;};
  operator +(a) { return new Point(x + a.x, y + a.y);}
}
```

Dart: Language, Platform, Tools, Engine(s)

- There are umpteen frameworks for any given purpose on the web. They don't interoperate very well
- Dart provides a solid set of core libraries. The core APIs are now stable.

Dart: Language, Platform, Tools, Engine(s)

- Language is designed to be toolable
  - Dynamic, but structured enough to support analysis
  - Classes
  - No eval()
  - Optional types

```
class Point {
  var x, y;
  Point(a, b){x =a; y = b;};
  operator +(a) {return new Point(x + a.x, y + a.y);}
}
```

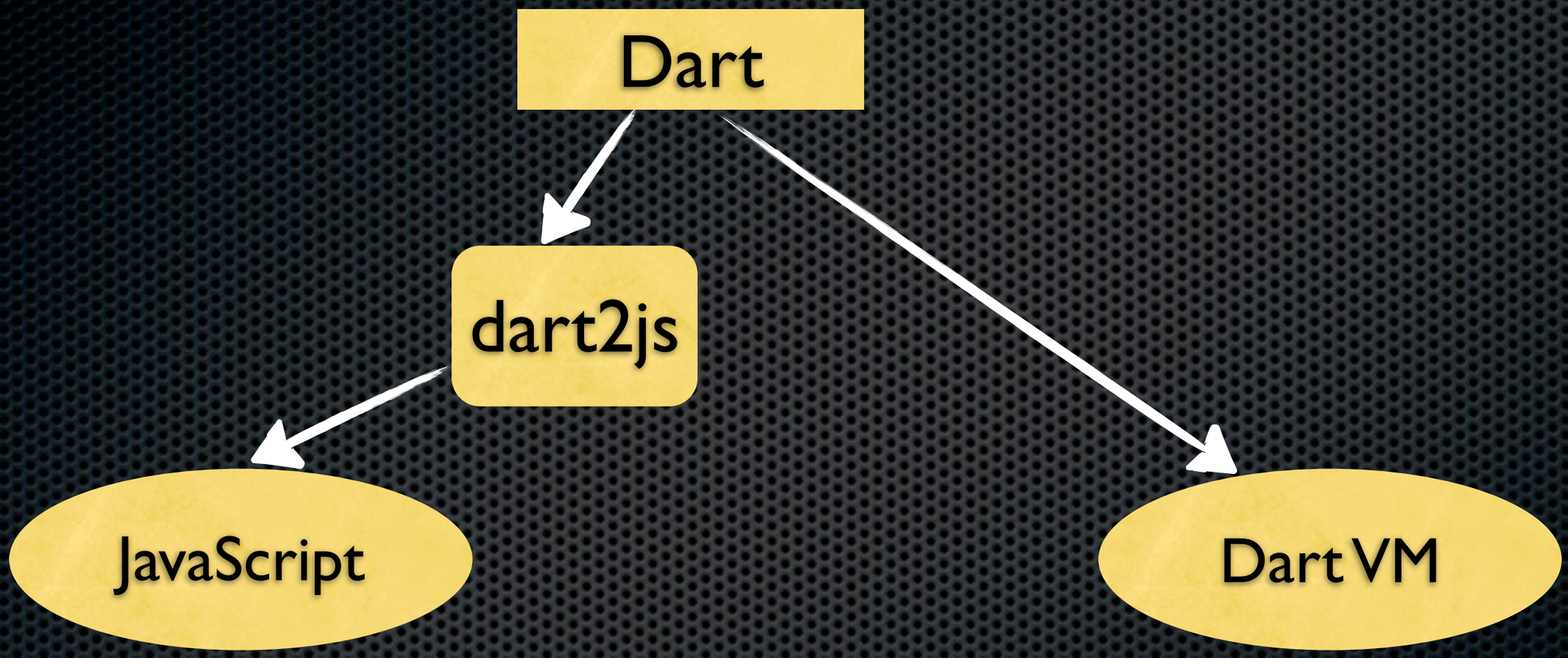
```
class Point {
  var x, y;
  Point(this.x, this.y);
  operator +(a) =>
    new Point(x + a.x, y + a.y);
}
```

```
class Point {
  int x, y;
  Point(this.x, this.y);
  Point operator +(Point a) =>
      new Point(x + a.x, y + a.y);
}
```

Dart: Language, Platform, Tools, Engine(s)

- VM now at roughly 2x V8 performance. Fastest dynamic language implementation ever
- Compilation to Javascript roughly on par with handwritten JS

Dart Execution



Overview

- Dart: Civilized Programming for the Web
- Class based, Purely Object Oriented, Optionally Typed, Mixin-based Inheritance, Message-passing Concurrency, Mirror-based Reflection

Everything is an Object

- As in Smalltalk, Ruby, Scala and others
- No autoboxing, no hidden coercions

Segue: Coercions

```
function (x){return x;}(false) ? print(“true”): print(“false”);
```

Segue: Coercions

- false is true:

```
function (x){return x;}(false) ? print(“true”): print(“false”);
```

Segue: Coercions

- false is true:

autobox

function (x){return x;}(false) ? print(“true”): print(“false”);)

Segue: Coercions

- false is true:

coerce object to bool

```
function (x){return x;}(false) ? print(“true”): print(“false”);
```

Segue: Coercions

- false is true:

  function (x){return x;}(false) ? print(“true”): print(“false”);

- Ignorance is Strength:
  - All this was known since PL/1 (1960s)
  - Those who ignore history are condemned to repeat it (cf. Javascript, or worse, PHP)

Representation Independence

- Fields are never accessed directly in Dart
- Instead, accessors (getters and setters) are created and used implicitly
  - x.a means invoke x's getter method

    get a { ... }
  - x.a = v means invoke x's setter method

    set a(x) { .... }

Representation Independence

Now, if I decide to change the representation of *x* objects to compute *a* instead, I can declare getters and setters as needed without breaking my clients. Even my subclasses can't tell.

Optional Types

Types Annotations

- Are syntactically optional
- Have no semantic effect

Types don’t effect Runtime

```
class LazyFields {
  var _x;
  get x =>  _x == null ? _x = complicated(): _x;
}
```

Types don’t effect Runtime

```
class LazyFields {
    int _x;
    int get x =>  _x == null ? _x = complicated(): _x;
}
```

Types are Interfaces

Dart has no interface declarations, but classes induce implicit interfaces that are reified.

```
abstract class Pair {get first; get second;}

class ArrayPair implements Pair {
  var rep = [];
  ArrayPair(a, b) { rep[0] =  a; rep[1] = b;}
  get first => rep[0]; get second => rep[1];
}
```

Mirrors

Classic OO Reflection

o.getClass().getMethods();

Mirrors are Different

Mirrors are objects that reflect *other* objects.

reflect(o).type.methods;

If you don’t have the right mirror, you cannot reflect, addressing difficulties in deployment, distribution, security

Mirrors are Different

reflect(o).type.methods;

Mirrors are Different

InstanceMirror

reflect(o).type.methods;

Mirrors are Different

ClassMirror

⟷

reflect(o).type.methods;

Mirrors are Different

Map<Symbol, MethodMirror>

reflect(o).type.methods;

Beyond Introspection

- Mirror builders for creating new/modified code
- Stack mirrors and Activation mirrors to support debugging

Caveats

- Web apps often optimized for size by eliminating symbols (minification) and unused code (tree shaking)
- Reflecting on code that has been optimized away is not possible
- Options:
  - ~~Do not optimize? Ouch.~~
  - Provide mechanism to selectively preserve reflective information

Caveats

- Web apps often optimized for size by eliminating symbols (minification) and unused code (tree shaking)
- Reflecting on code that has been minifed is possible
  - Using **constant** symbols, we can ensure that symbols correspond to minified names
  - Mapping back to real names entails space overhead

A History of Mirrors

- Mirror-based Reflection
  - Originated in Self
  - Used in Strongtalk, Java (JDI & APT), Newspeak, Scala
  - Now in Dart
  - Caveat Emptor: WIP! Coming to dart2js soon!

More on Mirrors

- Blog:
  - gbracha.blogspot.com/2010/03/through-looking-glass-darkly.html
- OOPSLA 2004 paper: bracha.org/mirrors.pdf
- 2010 Video:
- www.hpi.uni-potsdam.de/hirschfeld/events/past/media/100105_Bracha_2010_LinguisticReflectionViaMirrors_HPI.mp4

noSuchMethod()

```
class Proxy {
  final mirror;
  Proxy(forwardee): mirror = reflect(forwardee);
  noSuchMethod(Invocation i) => mirror.delegate(i);
}
```

Libraries

- LIbraries are Dart's modularity mechanism
- Libraries group classes, functions and variables in a top level scope
- Libraries are units of encapsulation, but not security
  - Privacy is per library. Private members are prefixed with _
- So far, libraries are NOT objects :-(
- Familiar import mechanism

Dart

- Class based, Purely Object Oriented, Optionally Typed, Mixin-based Inheritance, Mirror-based Reflection, Message-passing Concurrency
- Supports Software Engineering, Good Performance, Toolability
- Status: Open Source (Apache), Open Development

- http://www.dartlang.org