Dart

a *very* quick introduction

https://github.com/bseib/LightningishDart

google.com/+BrocSeib
broc.seib@gmail.com

GLOSSY lightning-ish talks
Apr. 26, 2014

What is Dart?

A programing language, developed by Google.

It runs client side in web browsers.

It can also run on server side.

Why Dart?

Do we need yet another language?

Large scale javascript development stinks.

You already know Dart. (Pretty much anyway.)

The usual suspects

Control Flow

while() {}
do {} while()
if () {} else {}
for (; ;)
for (x in list)
switch case

Built-in Types

double, int, num,
bool, String

Exception Handling

try {} catch() {} finally {}
throw

Classes

```
class Foo {
    String name;
    void printName() {
        print(name);
    }
}
```

Collections

Map, Set, List, Queue

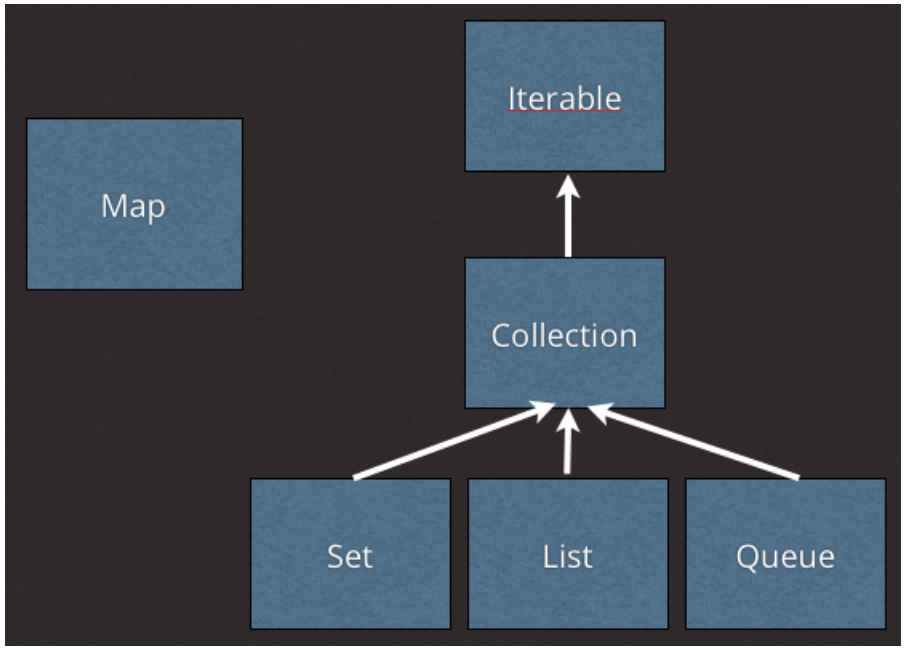

image source:
Seth Ladd's Dart tips

Literal assignments are supported:

```
var colors = ["red", "blue", "green"];
var histogram = { 'oak': 12, 'ash' : 34 };
```

What does Dart look like?

```
void main() {
  String favorite = "flying";
  say('riding');
  say('sailing');
  verbose(favorite);
}

say(var verb) {
  print("Look, I'm $verb!" + " Yay!");
}

verbose(verb) {
  print('''
You'll never believe it but you said you'd like to
be $verb and now all your hopes and dreams have come
true -- you are going to go ${verb.toUpperCase()} right now!
''');
}
```

Closures, fn => shorthand

```
main() {
  var hi = buildGreeter3("こんにちは, "); // Kon'nichiwa
  var result = hi("world");
  print(result);
}
```

`=> expr`

is shorthand for

`{ return expr; }`

```
buildGreeter3(String prefix) {
  return (name) => prefix + name + ".";
}
```

Cascade operator ..

```
main() {
  var shaggy = new CartoonCharacter()
    ..name = "Shaggy"
    ..likes = "food"
    ..catchphrase = "Zoiks!"
    ..act();
}

class CartoonCharacter {
  String name;
  String likes;
  String catchphrase;
  act() => print(catchphrase + " I really like " + likes + ". --" + name);
}
```

Optional parameters

```
main() {
  buildIceCream(["vanilla", "rocky road"]);
  buildIceCream(["chocolate", "mocha"], inBowl:true, hasSprinkles:true);
  buildIceCream(["vanilla bean", "pralines and cream", "mint chip"], hasSprinkles:true);
  payBill(1.23);
  payBill(4.56, true);
}

buildIceCream(flavors, {bool inBowl: false, bool hasSprinkles: false}) {
  print("your ice cream build starts with a " + (inBowl ? "bowl" : "cone") + " and is filled with:");
  for ( var f in flavors ) {
    print("$f ice cream, then");
  }
  if ( hasSprinkles ) {
    print("some sprinkles on top!");
  } else {
    print("that's it!");
  }
  print("");
}

payBill(amount, [bool hasLoyaltyCard]) {
  print("thanks for paying $amount" + ((hasLoyaltyCard!=null) ? " and being a loyal customer." : "."));
}
```

Generics

```
var colors = new List<String>();
colors.addAll(['Red', 'Blue', 'Green']);
colors.add(123); // fails in checked mode

var map = new Map<String, bool>();
// ...


class Cache<T> {
  T getByKey(String key);
  setByKey(String key, T value);
}
```

Private variables

```
library cartoon;

class CartoonCharacter {
  String name;
  String likes;
  String catchphrase;
  String _actingLines; // outsiders can't change this
  CartoonCharacter(this.name, this.likes, this.catchphrase);
  act() {
    _actingLines = catchphrase + " I really like " + likes + ". --" + name;
    _sayLines();
  }
  _sayLines() { // outsiders can't call this directly
    print(_actingLines);
  }
}

import 'cartoon.dart';

main() {
  var shaggy = new CartoonCharacter("Shaggy", "food", "Zoiks!");
  shaggy.act();
//  shaggy._actingLines = "ventriloquy strikes!";
//  shaggy._sayLines();
}
```

Getters and Setters

```dart
main() {
  var car = new Car();
  car.isEngineRunning = true;
  print(car.isEngineRunning); // true
}

class Car {
  bool isEngineRunning;
}
```

Getters and Setters

```dart
main() {
  var car = new Car();
  car.isEngineRunning = true;
  print(car.isEngineRunning); // true
}

class Car {
  Engine engine;

  bool get isEngineRunning {
    return engine.isRunning;
  }

  void set isEngineRunning(bool isRunning) {
    engine.isRunning = isRunning;
  }
}
```

Mixins

A la carte inheritance. Pick what you want.

```
// Wings is a mixin
// FlyingPig is called a mixin application
abstract class FlyingPig = Pig with Wings;
```

A class declaration + convention makes a mixin.

- no constructor
- super class is Object
- never call super()

Mixin Example

```
void main() {
  var wilbur = new Pig("Wilbur");
  wilbur.findTruffles();

  var pigxie = new FlyingPig("Pigxie");
  pigxie.flyTo("southern France");
  pigxie.findTruffles();
}


class FlyingPig extends Pig with Wings {
  FlyingPig(name) : super(name);
  void findTruffles() {
    super.findTruffles();
    print("but had jet lag.");
  }
}

class Pig {
  var name;
  Pig(this.name);
  void findTruffles() {
    print("$name found some truffles");
  }
}

/**
 * Wings is a Mixin. So follow the conventions of a mixin:
 * No constructor, super class is Object, never call super()
 */
class Wings {
  void flyTo(location) {
    print("(flying to $location...)");
  }
}
```

Isolates

Dart is single threaded

Concurrent tasks have private, isolated memory

```
import 'dart:isolate';

void main() {
  for (int i=0;i<10;i++) {
    Isolate.spawn(runMe, i);
  }
}

void runMe(i) {
  print("i am here: ($i)");
}
```

Mirror reflection

Classic OO way to do reflection:

```
o.getClass().getMethods();
```

Mirror way to do reflection:

```
reflect(o).type.declarations;
```

What to do with Dart?

A compelling use for Dart is client apps.

Powerful libraries for rich web apps:

- Polymer
- AngularDart

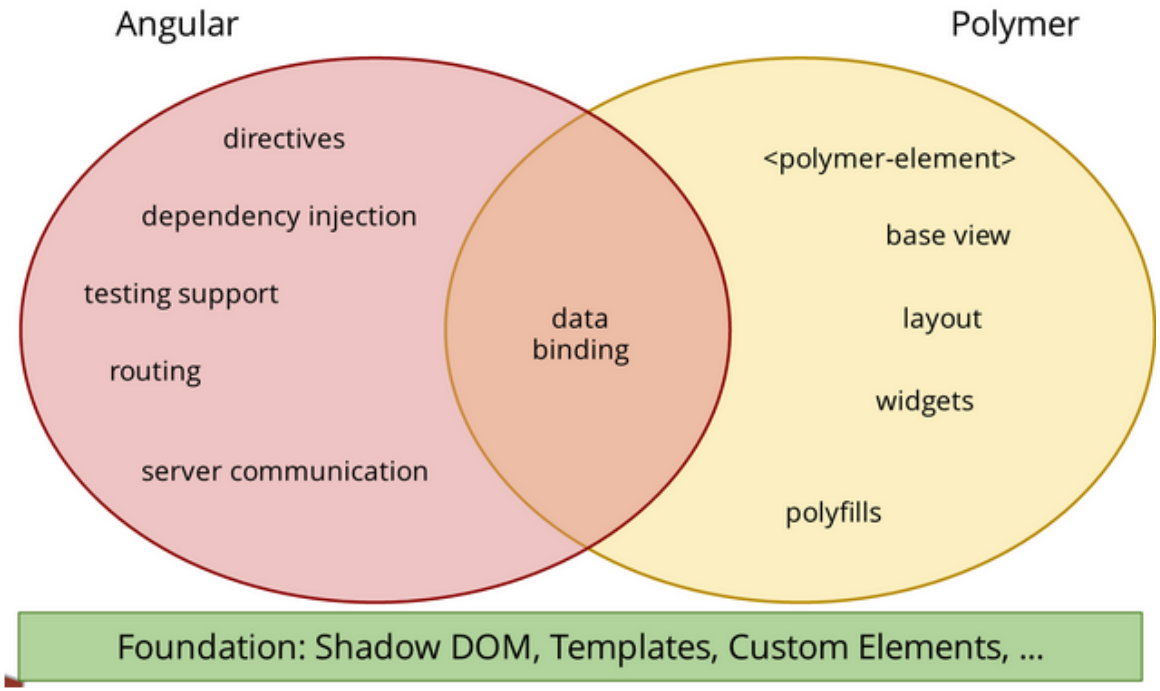

image source: Seth Ladd's blog

// thanks!

exit(0);

google.com/+BrocSeib
broc.seib@gmail.com