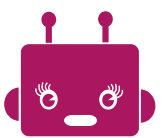
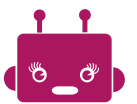
Dart for Java developers



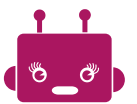
What to focus on

- Language
  - Syntax
  - Code structure
- Libraries
  - Basic
  - Platform-dependent
- Tools



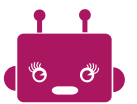
Core

- .dart file is a “module”
- modules live in packages
- top-level functions, variables, constants and classes
- names starting with “_” (underscore) are private to the module
- everything is an object, except for *null*
- null-safety (with „?” for nullable)
- generics: reified, accessible at runtime; no erasure
- type inference (var)
- final: assign-once; const: compile-time constant.



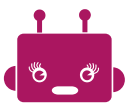
Imports

- import from
  - built-in libraries (*import 'dart:html';*),
  - local filesystem (*import './foo.dart;'*) or
  - external package managed by build tool, like pub (*import 'package:test/test.dart';*)
- aliases in imports (as, show, hide)



Asynchrony

- async, await; Future and Stream classes
- code executes in the same isolate
- sync and async generator functions
- isolates as “threads” without race conditions, see concurrency



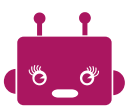
Generators

- Synchronous: return *Iterable*

```
Iterable<int> naturalsTo(int n) sync* {
  int k = 0;
  while (k < n) yield k++;
}
```

- *Asynchronous: return Stream*

```
Stream<int> asynchronousNaturalsTo(int n) async* {
  int k = 0;
  while (k < n) yield k++;
}
```



kamilachyla.com

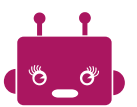
Classes

- callable classes
- typedefs
- function types

```dart
class WannabeFunction {
  String call(String a, String b, String c) => '$a $b $c!';
}

var wf = WannabeFunction();
var out = wf('Hi', 'there,', 'gang');

void main() => print(out);
```

```dart
typedef ListMapper<X> = Map<X, List<X>>;
Map<String, List<String>> m1 = {}; // Verbose.
ListMapper<String> m2 = {}; // Same thing but shorter and clearer.
```

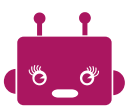
```dart
Iterable<T> where(bool Function(T) predicate) => ...
```



kamilachyla.com

Operator overloading

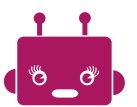
```
class Vector {
  final int x, y;

  Vector(this.x, this.y);

  Vector operator +(Vector v) => Vector(x + v.x, y + v.y);
  Vector operator -(Vector v) => Vector(x - v.x, y - v.y);

  @override
  bool operator ==(Object other) =>
      other is Vector && x == other.x && y == other.y;

  @override
  int get hashCode => Object.hash(x, y);
}

void main() {
  final v = Vector(2, 3);
  final w = Vector(2, 2);

  assert(v + w == Vector(4, 5));
  assert(v - w == Vector(0, 1));
}
```



kamilachyla.com

Getters, setters

```
class Rectangle {
  double left, top, width, height;

  Rectangle(this.left, this.top, this.width, this.height);

  // Define two calculated properties: right and bottom.
  double get right => left + width;
  set right(double value) => left = value - width;
  double get bottom => top + height;
  set bottom(double value) => top = value - height;
}

void main() {
  var rect = Rectangle(3, 4, 20, 15);
  assert(rect.left == 3);
  rect.right = 12;
  assert(rect.left == -8);
}
```



kamilachyla.com

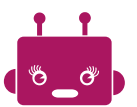
Extension methods

- On specific type

```
extension NumberParsing on String {
  int parseInt() {
    return int.parse(this);
  }

  double parseDouble() {
    return double.parse(this);
  }
}
```

- Generic

```
extension MyFancyList<T> on List<T> {
  int get doubleLength => length * 2;
  List<T> operator -() => reversed.toList();
  List<List<T>> split(int at) => [sublist(0, at), sublist(at)];
}
```

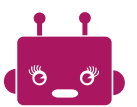


kamilachyla.com

Mixins

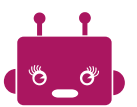
- Allow code reuse across class hierarchies
- Use *mixin* keyword to not allow instantiation
- Implement: extends Object, no constructor
- Can be restricted (with *on*) to required superclass

```
class Musician {
  // ...
}
mixin MusicalPerformer on Musician {
  // ...
}
class SingerDancer extends Musician with MusicalPerformer {
  // ...
}
```



Standard library

- Nice overview: https://dart.dev/guides/libraries/library-tour
- Must know (details: dart API reference):
  - dart:core
  - dart:async
  - dart:convert
  - dart:io (on dart vm, for scripts/cmdline apps)
  - dart:html (on web)
- Other libs: https://pub.dev/



Tools

- dartpad - learn syntax, embed
- dart-tool - cmdline tool
  - create, format, analyze, test, document, compile Dart code
- pub - Dart package manager
- webdev- tool to build and serve Dart web apps

More at: https://dart.dev/tools#general-purpose-tools