Dart Cheat Sheet

1. Comments

rest of line	`//…`
rest of line and multi-line	`/*…*/`
documentation	`///…`

2. Operators

unary postfix	`…++  …--  …()  …[]  .` `…?[]`[2] `…![]`[3] `…?.`[2] `…!.`[3]

unary prefix	`-…  !…  ~…  ++…  --…`
	`await …`

multiplicative	`*  /  %  ~/`
additive	`+  -`
shift	`<<  >>  >>>`[1]
bitwise	`&  ^  |`

relational and type test	`>=  >  <=  <`
	`as  is  is!`

equality	`==  !=`
logical	`&&  ||`
if null	`??`
tertiary	`expr ? expr : expr`
cascade	`..` `?..`[2] `!..`[3]

assignment	`=  *=  /=  %=  ~/=  +=  -=  <<=`
	`>>=  >>>=  &=  ^=  |=  ??=`

spread	`...` `...?`[2]

[1] Unsigned shift right
[2] Conditional access if not null
[3] Runtime error if null

3. Core Data Types

void	`void`
boolean	`bool`
64-bit integer	`int`
64-bit float	`double`
string[1]	`String`
dynamic (runtime)	`dynamic`
symbol	`Symbol`
collections	`List  Set  Map`
functions	`Function`
futures	`Future`
streams	`Stream`

[1] Sequence of UTF-16 code units

4. Declarations

explicit type	`type ident;`
	`type ident = expr;`
	`const type ident = expr;`
	`final ident;`
	`final ident = expr;`
	`late ident;`

inferred type	`var ident = expr;`
	`const ident = expr;`
	`final ident = expr;`

enumeration[1]	`enum ident {`
	`  ident,`
	`  ident,`
	`}`

generic contraints	`…<T extends Type>…`
type alias	`typedef ident = type;`

[1] Define at global scope. Use `index` getter for value.

5. Literals

decimal int	`123`
float	`123.45  1.0e4  8e5`
hex	`0x1234ABCD`
boolean	`true  false`

strings	`"abc"`
	`'abc'`
	`"""abc"""`
	`r"abc"`

interpolated string	`'$ident ${expr}'`
character	`\r  \n  \t`

unicode code point	`\u2665`
	`\u{1f606}`

symbol[1]	`#ident`

list	`[expr, expr, …]`
	`<type>[ … ]`

set	`{ expr, expr, …}`
	`<T, …>{…}`

map	`{ const: expr, …}`
	`<type, type>{…}`

[1] Symbols are not minified

6. Control Flow

if/then/else	`if (expr) {…}`
	`else if (expr) {…}`
	`else {…}`

for loop[2]	`for (stmt; expr; stmt) {…}`
	`for (decl in iter) {…}`

async for loop[2]	`async for (decl in stream) {…}`
while[2]	`while (expr) {…}`
do while[2]	`do {  } while (expr);`

try/catch/ finally	`try {…}`
	`on type {…}`
	`on type catch (ident) {…}`
	`catch (ident ) {…}`
	`catch (ident, ident`[3]`) {…}`
	`finally {…}`

switch[1]	`switch (expr) {`
	`  case const1:`
	`    break;`
	`  case const2:`
	`    break;`
	`  case const3:`
	`    continue label;`
	`  …`
	`  label:`
	`  default:`
	`    break;`
	`}`[1]

return	`return;`
	`return expr;`

continue	`continue;`
	`continue label;`

break	`break;`

[1] Local variables are scoped to case clause.
[2] Can use break & continue to alter control flow
[3] Stack trace

7. Functions, Closures & Generators

functions[1]	`type ident(arg, arg2) {`
	`  return expr;`
	`}`

async functions[1]	`Future<T> ident(…) async {`
	`  …`
	`}`

closure[2]	`() => expr`
	`arg => expr`
	`(arg, …) => expr`
	`() {}`
	`(arg, …) {`
	`  …`
	`  return expr;`
	`}`
	`(…) async {…}`

generic functions	`type ident<T, …>(…) {…}`

async. function	`Future<T> ident(…) async {…}`
	`(…) async => expr`

sync. generator	`Iterable<T> ident(…) sync* {`
	`  …`
	`  yield expr;`
	`}`

async. generator	`Stream<T> ident(…) async* {`
	`  …`
	`  yield expr;`
	`}`

recursive generator[2]	`yield* expr;`

[1] Methods have access to the `this` variable
[2] Can use for both `sync*` and `async*` generators

8. Function & Constructor Parameters

positional	`(type ident, type ident)`
optional positional[1]	`(type ident, [type? ident])`
named	`({type ident})`
default named	`({type ident=const})`
required named	`({required type ident})`
mixed positional & named	`(type ident, …, {type ident, …})`

[1] Cannot be used with named arguments

9. Additional List Operations

for	`[for (…) expr]`
if	`[if (expr) expr]`

spread	`[ident,`
	`...ident,`
	`...?ident]`

10. Imports & Exports

library[1]	`library ident;`

imports	`import 'file.dart';`
	`import 'package:ident/…';`
	`import 'dart:ident';`

exports	`export 'file.dart' show ident;`

alias/ deferred[2]	`import … as ident`
	`import … deferred as ident`

show/hide	`import … show ident`
	`import … hide ident`

[1] Only required for metadata & documentation
[2] Use `deferred` with `loadLibrary()`. `dart2js` only.

11. Classes

class/ generic class	`class Type {`
	`  fields`
	`  constructors`
	`  properties`
	`  methods`
	`}`
	`class Type<T,…> {…}`

static/const	`static decl`
	`const decl`
	`const static decl`

constructor[1]	`Type(…)`
	`Type(…) : super(…)`
	`Type(…) : super.ident(…)`
	`Type(…) : ident = expr, …`

call superclass	`{ …; super();}`
	`{ super(); …}`

assignment sugar	`Ident(this.ident, …)`
named constructor	`Ident.ident(…)`

properties	`type get ident {…}`
	`type get ident => expr`
	`type set ident {…}`

inheritence	`class Type extends Type {…}`
interface	`class Ident implements Ident {…}`

mix-in	`mixin Type {…}`
	`mixin Type on Type {…}`

callable class	`class … { call(…) {} }`
builtin metadata[2]	`@Override  @Deprecated`
abstract class	`abstract class Type {…}`

extension[3]	`extension Ident on Type {…}`
	`Ident(Type(…))`

[1] Constructors not inherited. Default constructor calls `super(…)`. Right hand side cannot access `this`.
[2] Custom metadata is just a simple `class`.
[3] Use "wrapper class" syntax only for name conflicts.

v0.3. Updated for Dart v2.16.0.