Dart notes

Frank Braun

2018-11-08

Distilled from Dart language tour[1].

Basics

- `main()`: Required top-level function where app execution starts.
- `print()`: Print output.
- `$variableName` (or `${expression}`): include variable or expression's string equivalent inside string literal.
- `var`: Declare a variable without specifying its type.

Comments

- Single-line comments: `//`
- Multi-line comments: `/* ... */`
- Documentation comments: `///` or `/**`, links in square brackets (to classes, methods, fields, top-level variables, functions, and parameters in lexical scope).

Built-in types

- `int`: integer no larger than 64-bit, depending on the platform.
- `double`: 64-bit (double-precision) floating-point numbers.
- `int` and `double` are subtypes of `num`.
- `bool`: implemented by the two boolean literals `true` and `false`.
- String: sequence of UTF-16 codes. Use single or double quotes to create.
- Lists: `var list = [1, 2, 3];` (zero-based indexing).
- Maps: `var map = { 2: 'foo', 10: 'bar', 18: 'baz' };` or `Map()`.

[1] https://www.dartlang.org/guides/language/language-tour

Important concepts

- Every variable is an *object*, and every object is an instance of a *class*.
- Even numbers, functions, and `null` are objects.
- All objects inherit from the `Object` class.
- Strongly typed, but optional type annotations (type inference).
- `dynamic`: No type is expected.
- Generic types: `List<int>` or `List<dynamic>`.
- Dart supports:
  - Top-level functions (such as `main()`).
  - Functions tied to a class or object (static and instance methods).
  - Functions within functions (nested or local functions).
- Similarly, Dart supports:
  - Top-level variables.
  - Variables tied to a class or object (static and instance variables).
  - Instance variables are sometimes known as fields or properties.
- Private identifiers start with an underscore (`_`).
- Dart has both expressions (which have runtime values) and statements (which don't). For example, the conditional expression `condition ? expr1 : expr2` has a value of `expr1` or `expr2`. Compare that to an if-else statement, which has no value.

Language

- Uninitialized variables have an initial value of `null`.
- `assert` statement to disrupt normal execution if a boolean condition is false (disabled in production code).
- If you never intend to change a variable, use `final` or `const`, either instead of `var` or in addition to a type.
- A final variable can be set only once; a const variable is a compile-time constant. Const variables are implicitly final.
- A final top-level or class variable is initialized the first time it's used.
- To get the symbol for an identifier, use a symbol literal: `#foo`.

Functions

Functions are objects and have the type `Function`. Example with type annotations:

```
bool isNoble(int atomicNumber) {
  return _nobleGases[atomicNumber] != null;
}
```

Shorthand syntax for functions that contain just one expression:

```
bool isNoble(int atomicNumber) => _nobleGases[atomicNumber] != null;
```

The `=> expr` syntax is a shorthand for `{ return expr; }` (arrow syntax).