Dart

basic Dart、Built-in types

Important concepts

- Everything is Object (numbers, functions, null), every object is an instance of a Class
- strongly typed
- Doesn’t have the keyword public, protected, and private . Use underscore(_) , it’s private to its library

Variables

- using var, rather than type annotations, for local variables.
  var name = ‘Bob’, String name = ‘Bob’

- Default value - null

Final vs Const

- Use const for variables that you want to be compile-time constants

- Final instance variables must be initialized before the constructor body starts

- Final should be used over const if you don't know the value at compile time, and it will be calculated/grabbed at runtime.

Numbers

- int - values no larger than 64 bits, depending on the platform
- double - 64-bit (double-precision) floating-point numbers
- both int and double are subtypes of num (abs() , ceil(), floor() and more func in dart: math)

Numbers

- String -> int: 1 = int.parse(‘1’)
  String -> double: 1.1 = double.parse(‘1.1’)

- int, double -> String: 1.toString(), 2.2.toString(), 3.14159.toStringAsFixed(2)

- assert((3 << 1) == 6) // 0011 << 1 == 0110
  assert((3 >> 1) == 1) // 0011 >> 1 == 0001
  assert((3 | 4) == 7) // 0011 >> 0100 == 0111

String

- var firstName = ‘Mark’;
  var lastName = ‘Chen’;
  "$firstName ${lastName}” //Mark Chen
  “$firstName” + “ ${lastName}” //Mark Chen
  "$firstName ${lastName.toUpperCase()}” //Mark CHEN

- var s1 = '''
  You can create
  multi-line strings
  like this one.
  ''';

Booleans

- var oops = 0 / 0;
  oops.isNaN // true

List(Array)

- 0 is the index of the first element
  `list.length - 1` is the index of the last element

- ```
  var list = [1, 2, 3];
  var list2 = [4, 5, 6];
   list = [...list, ...list2]; // [1, 2, 3, 4, 5, 6]
  ```

- ```
  var list;
  var list2 = [0, ...?list]; // [0]
  ```

List(Array)

- var nav = [‘Home’, ‘Furniture’, if(promoActive) ‘Outlet’];

- var listOfInts = [1,2,3];
  var listOfStrings = [for(var i in listOfInts) ‘a$i’];
  // listOfStrings = [a1, a2, a3]

Set

- an unordered collection of unique items
- var setA = {1, 2, 3, 3}; // {1, 2, 3, 3}
- var setB = <int>{};
  Set<int> setB = {};
  var setB = {}; create a map, not a set

Map

- an object that associates keys and values
- Both keys and values can be any type of object
- Each key occurs only once, but you can use the same value multiple times

Map

- var gifts = {
  // key: value
  ‘first’: ‘bird’,
  ‘second’: ‘dog’,
  ‘third’: ‘cat’
  };

- gifts[‘fourth’] = ‘bear’; // Add a key-value pair

- assert(gifts[‘fifth’] = null)