Dart Building Blocks Part 1

Key Points	Notes

1. What is Dart?

a. Dart is developed by Google for general purpose programming language, which uses C-based languages syntax and conventions (C++, C# & Java).

b. Dart is purely object-oriented, optionally typed (auto assigned data type), a class-based language that support functional and reactive programming (events).

2. Why using Dart?

a. Dart is the core language for flutter mobile programming and it allow fast prototyping for mobile application development.

b. Dart is mostly used for web development, which you can extend the web application to mobile application. Similar to React.js and React Native.

c. Dart uses high level programming (Java and JavaScript), which it is easy to use, fast and less learning curve to learn.

3. Core Syntax

a. You can use any text editor to code Dart and save the code as *.dart extension. Alternatively, you can use Dart online editor which is available at https://dartpad.dev

b. Sample of Dart programming code:

```
void main(){ // top level function where app execution
starts
  display();
}

void display(){ // normal function
  print("Hello World!");
}
```

Summary

Key Points	Notes

4. Variables

a. You can create variable in a few different ways:

i. `var name = 'Ali';`
ii. `dynamic name = 'Ali';`
iii. `String name = 'Ali';`

b. Default value in Dart.

i. Uninitialized variables have an initial value of null even with the assigned data types because everything in Dart is object.

c. Keywords.

i. Do NOT use these keywords as your variable or identifier because they are reserved words.

ii. List of Dart Keywords.

5. Data Types

a. Available Data Types in Dart:

Type	Example
int	23
double	23.12134
bool	true/false
String	"hello"+"world"
dynamic	23/23.12134/true/false/"hello"

b. Infer data types

i. Data types are determined or inferred based on the assigned value.

Examples:

1. `var pi = 3.412;`
2. `dynamic pi = 3.412;`

Summary

Key Points

Notes

6. Operators

a. Arithmetic (Number Operations)

Operator	Operation
+	Addition and string concatenation
-	Subtraction
*	Multiplication
/	Division
%	Remainder

b. Unary

Operator	Operation
++	Increment by 1
--	Decrement by 1
!	Invert the value of a Boolean

c. Equality and Relational (Boolean)

Operator	Operation
==	Equality
!=	Not Equal
> & <	Greater than & Less than
>= & <=	Greater than or equal & Less than or equal

d. Logical (Boolean)

Operator	Operation
||	OR (True if either Boolean expression is true)
&&	AND (True if all Boolean expressions are true)

Summary

Key Points	Notes

7. Strings

a. String in Dart is represented by **single quotes** or **double quotes**, which is similar to JavaScript.

Examples:

i. `var str = 'This is string';`
ii. `String str = "This is string";`
iii. `dynamic str = "This is string's string";`

b. **Values** and **expressions** can be embedded in strings to create new strings.

Example:

```
var firstName = 'Ahmad';
String lastName = "Albab";

dynamic fullName = "$firstName $lastName";
print(fullName); // print 'Ahmad Albab'
```

c. Combine **adjacent strings** either in multiple lines or next to each other.

Example:

```
//No concatenation required
var line1 = 'This is first string\t' 'this in second
string\n'
"this is third string in second line";
print(line1);


//With concatenation (+)
var line2 = 'This is 1' + "this is 2" + 'this is 3';
print(line2);
```

Summary

Key Points	Notes

	d. Escape sequence and delimit strings.
	
	Example:
	
	`//Escape sequence`
	`var firstWord = 'I don\'t have a car';`
	`print(firstWord);`
	
	`//Delimit`
	`var secondWord = "I don't have a car";`
	`print(secondWord);`
	
	e. Preserve string formatting in multiple lines.
	
	Example:
	
	`var car = """`
	`Volkswagen Golf Mk 8,`
	`2.0 TSI,`
	`4Motion.`
	`""";`
	`print(car);`
	
	f. Display string formatting.
	
	Example:
	
	`var str = "\r First line\n Second line\t Third line";`
	`print(str);`

Summary

Key Points	Notes

	8. Immutability (static)
	a. A `const` is used when a value is **known at compile time** (before the program runs or during compilation) such as integer or string literal but **cannot be re-assigned** after being initialized.
	
	Example:
	
	`const pi = 3.142;`
	`pi = 2.1; // error can't assign to the const variable`
	
	`var x = const [];`
	`x = [12]; // successfully assigned because x is not const variable`
	
	`const y = [];`
	`y = [12]; // error can't assign to the const variable`
	
	b. A `final` is used when a value is **unknown at compile time** but **cannot be re-assigned** after being initialized.
	
	Example:
	
	`final house = 'Terrace';`
	`house = 'apartment'; // error can't assign to the final variable`
	
	`final String houseType = 'T-A';`
	`print('$house with house type $houseType'); // Terrace with house type T-A`

Summary

Key Points	Notes

	9. Nullability (Null)
	a. Any variable declared without value assignment or initialization will be given a null value. A null value means nothing is stored in the variable.
	
	Example:
	
	`var x;`
	`int? y;`
	`String? z;`
	`double? a;`
	`dynamic b;`
	`print("x=$x, y=$y, z=$z, a=$a, b=$b"); // All null values`
	
	b. All Dart data types are derived from a type named `object`, which any uninitialized object will result a null value.
	
	c. A **null-aware operator** (`??`) is used when working with null values. This helpful for checking null values before performing any operation.
	
	Example:
	
	`int? x;`
	`var y = x ?? 1;`
	`print(y);`
	
	`x ??= y;`
	`print(x);`
	
	d. There is `?.` operator that protect programmer from accessing properties on **null objects**.
	
	Example:
	
	`print(car?.isAutomatic); // return null`

Summary

Key Points	Notes

10. Control Flow

a. Control flow includes **conditionals (if-else, ternary operator & switch)** that allow program to decide whether to execute or skip certain parts of the code. Conditionals in Dart are similar to other C-like languages.

Example:

```
//if-else
int total = 75;
if(total >= 70 && total <= 74){
  print(“B+”);
} else if(total >= 75 && total <= 79){
  print(“A-“);
} else if(total >= 80 && total <= 100){
  print(“A”);
}


//ternary operator
(total >= 70 && total <= 74)? print(“B+”) : print(“A-“);


//switch
String grade = “A-“;
switch (grade){
 case “A”:
  print(“80-100”);
  break;
 case “A-“:
  print(“75-79”);
  break;
 case “B+”:
  print(“70-74”);
  break;
}
```

Summary

Key Points	Notes

11. Loops

a. Loops allow program to repeat code execution to a certain number of times or based on certain conditions. This helpful for repetitive operations.

Example:

```
List<String> fruits = ['apple','orange','banana'];

//while loop
int i = 0; // initial counter value
while(i < fruits.length){  // loop condition
   print(fruits[i]);
   i++;    // update counter value
}

//do-while loop
int i = 0; // initial counter value
do{
  print(fruits[i]);
  i++;  // update counter value
} while (i < fruits.length); // loop condition

//for loop
for(int i = 0; i < fruits.length; i++){
  print(fruits[i]);
}

//for in loop
for(String fruit in fruits){
  print(fruit);
}

//for each loop
fruits.forEach((String fruit){
  print(fruit);
});
```

Summary

Key Points	Notes

```
//foreach loop
Map<int, String> person = {23:'Ali', 25:'Aminah'};
person.forEach((age,name) => print('$name is ${age} years
old'));

var person = {23:'Ali', 25:'Aminah'};
person.forEach((age,name) => print('$name is ${age} years
old'));
```

12. References

a. A tour of the Dart Language. Retrieved on October 2nd, 2020 from https://dart.dev/guides/language/language-tour
b. Howard, J. (2019). Dart Basics. Retrieved from https://www.raywenderlich.com/4482551-dart-basics#toc-anchor-014
c. Dart Programming Tutorial. Retrieved on October 2nd, 2020 from https://www.tutorialspoint.com/dart_programming/index.htm

Summary