Dart Basics

1. **Dart Single line Comment:**
   Dart single line comment is used to comment a line until line break occurs. It is done using a double forward-slash (//).
   // This is a single line comment.

1. **Dart Multi-Line Comment:**
   Dart Multiline comment is used to comment out a whole section of code. It uses '/*' and '*/' to start and end a multi-line comment respectively.

1. **Dart Documentation Comment:**
   Dart Documentation Comments are a special type of comment used to provide references to packages, software, or projects.Dart supports two types of documentation comments "///"(C# Style) and "/**…..*/"(JavaDoc Style). It is preferred to use "///" for doc comments as many times * is used to mark list items in a bulleted list which makes it difficult to read the comments. Doc comments are recommended for writing public APIs.

Conditions to write variable name or identifiers

- Variable name or identifiers can't be the **keyword**.
- Variable name or identifiers can contain alphabets and numbers.
- Variable name or identifiers can't contain spaces and special characters, except the **underscore(_)** and the **dollar($)** sign.
- Variable name or identifiers can't begin with number.

Dart programming languages keywords

- In Dart programming languages keywords are categorized as Reserved Words, Contextual Keywords, Built-in Identifiers Words, and Keywords for Asynchrony Support.

- **1. Reserved Words**

  Reserved Words: These are main keywords in dart and they are: ***assert, break, case, catch, class, const, continue, default, do, else, enum, extends, false, final, finally, for, if, in, is, new, null, rethrow, return, super, switch, this, throw, true, try, var, void, while, with***

  2. Contextual Keywords

  Contextual Keywords: These Keywords have meaning only in specific places. They're valid identifiers everywhere. Contextual Keywords are: ***async, hide, on, show, sync***

Dart programming languages keywords

- **Built-in Identifier Words**
- Built-in Identifier Words: These are used to simplify the task of porting JavaScript code to Dart, these keywords are valid identifiers in most places, but they can't be used as class or type names, or as import prefixes. These words are: ***abstract, as, covariant, deferred, dynamic, export, extension, external, factory, function, get, implements, import, interface, library, mixin, operator, part, set, static, typedef***

- **keywords for Asynchrony Support**
- Keywords for Asynchrony Support: These are newer, limited reserved words related to the asynchrony support. And these keywords are: **await, yield**

Dart Data Types

- Dart supports the following built-in Data types.
- **Number**
  - **int** marks = 80;
  - **double** pi = 3.14;
- **Strings**
- **Boolean**
- **Lists**
- **Maps**

Dynamic type variable in Dart:

- This is a special variable initialised with keyword dynamic. The variable declared with this data type can store implicitly any value during running the program. It is quite similar to var datatype in Dart, but the difference between them is the moment you assign the data to variable with var keyword it is replaced with the appropriate data type.
- Syntax: dynamic variable_name;

```
void main() {
    dynamic g = "ttttt";
    print(g);
   // Reassigning the data to variable and printing it
   g = 3.14157;
   print(g);

}
```

Dart Data Types

- **Dart Lists**
- In Dart, The list is a collection of the ordered objects (value). The concept of list is similar to an array. An array is defined as a collection of the multiple elements in a single variable. The elements in the list are separated by the comma enclosed in the square bracket[]. The sample list is given below.
- var list = [1,2,3]

- **Dart Maps**
- The maps type is used to store values in key-value pairs. Each key is associated with its value. The key and value can be any type. In Map, the key must be unique, but a value can occur multiple times. The Map is defined by using curly braces ({}), and comma separates each pair.
- var student = {'name': 'Joseph', 'age':25, 'Branch': 'Computer Science'}

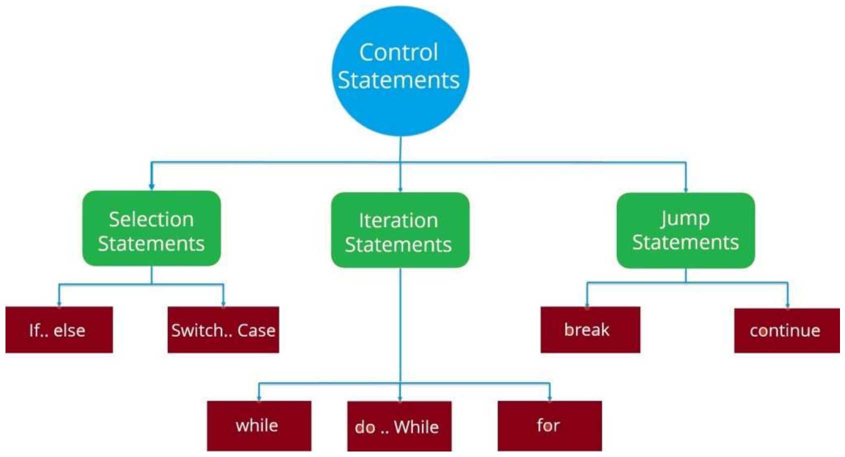
Dart main() Function

- This is the most vital part of each and every Dart program, it is mandatory for every Dart program to have a top-evel **main()** function definition. There can be only one **main()** function in Dart program. It serves as the entry point for every Dart program or app; the execution of a Dart program or application starts with the **main()** function..

- void main()

```
 {
   print("Hello World!");
  }
```

Dart Control Flow Statement Types



Dart Control Flow Statement Types

```
void main()
{
   var num = 75;
   print(" Dart If Statement");

            if(num > 50){
     print("Number Greater than 50");
   }

}
```

```
void main() {

   int dayOfWeek = 5;
   print(" Dart Switch Case statement.");
   switch(dayOfWeek){


   case 1:{
   print("Today is Monday."); }
       break;

   case 5:{
   print("Today is Tuesday."); }
   break;
     default:{
   print("Invalid Weekday."); }
   break;
   }


}
```

Dartpad and Dart Langauge

- Dartpad is a browser-based text editor that can execute Dart code, created by the Dart team. It's a "playground" or "scratchpad" of sorts, useful for testing small pieces of code. If you're just starting out with Dart, you can use Dartpad to get your feet wet without having to install the Dart SDK and IDE plugins.

- DartPad is an **open source tool that lets you play with the Dart language in any modern browser**. Many pages in this site — especially codelabs — have embedded DartPads.

Assignment 1

- In this assignment, you require to run all dart language examples given in lectures D L1 and D L2 using Dartpad

- Open https://dartpad.dev/?