Comments

Comments are the set of statements that are ignored by the dart compiler during program execution. They are used to explain the code so that you or other people can understand it easily.

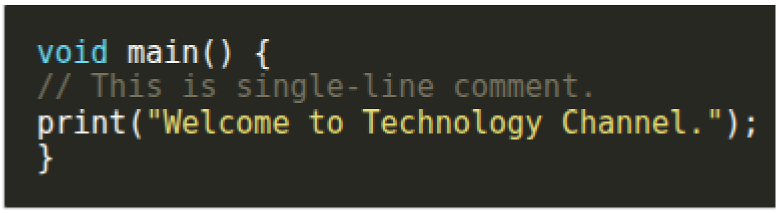
Advantages Of Comments

- You can describe your code.
- Other people will understand your code more clearly.

Types Of Comments

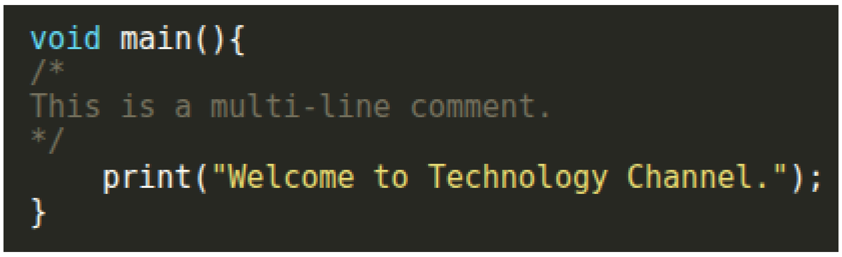
- Single-Line Comment: For commenting on a single line of code. E.g. // This is a single-line comment.
- Multi-Line Comment: For commenting on multiple lines of code. E.g. /* This is a multi-line comment. */
- Documentation Comment: For generating documentation or reference for a project/software package. E.g. /// This is a documentation comment

Single-Line Comment In Dart



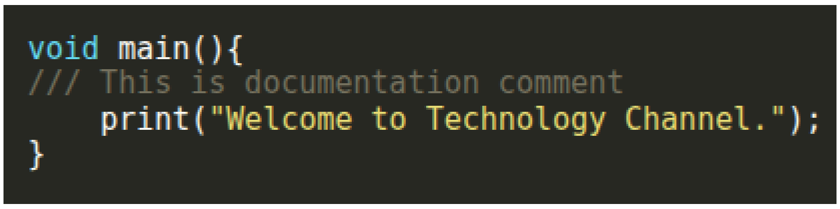
```
void main() {
// This is single-line comment.
print("Welcome to Technology Channel.");
}
```

Multi-Line Comment In Dart



```
void main(){
/*
This is a multi-line comment.
*/
    print("Welcome to Technology Channel.");
}
```

Documentation Comment In Dart



```
void main(){
/// This is documentation comment
    print("Welcome to Technology Channel.");
}
```

Operators In Dart

Operators are used to perform mathematical and logical operations on the variables. Each operation in dart uses a symbol called the operator to denote the type of operation it performs. Before learning operators in the dart, you must understand the following things.

- Operands : It represents the data.
- Operator : It represents how the operands will be processed to produce a value.

Note: Suppose the given expression is 2 + 3. Here 2 and 3 are operands, and + is the operator.

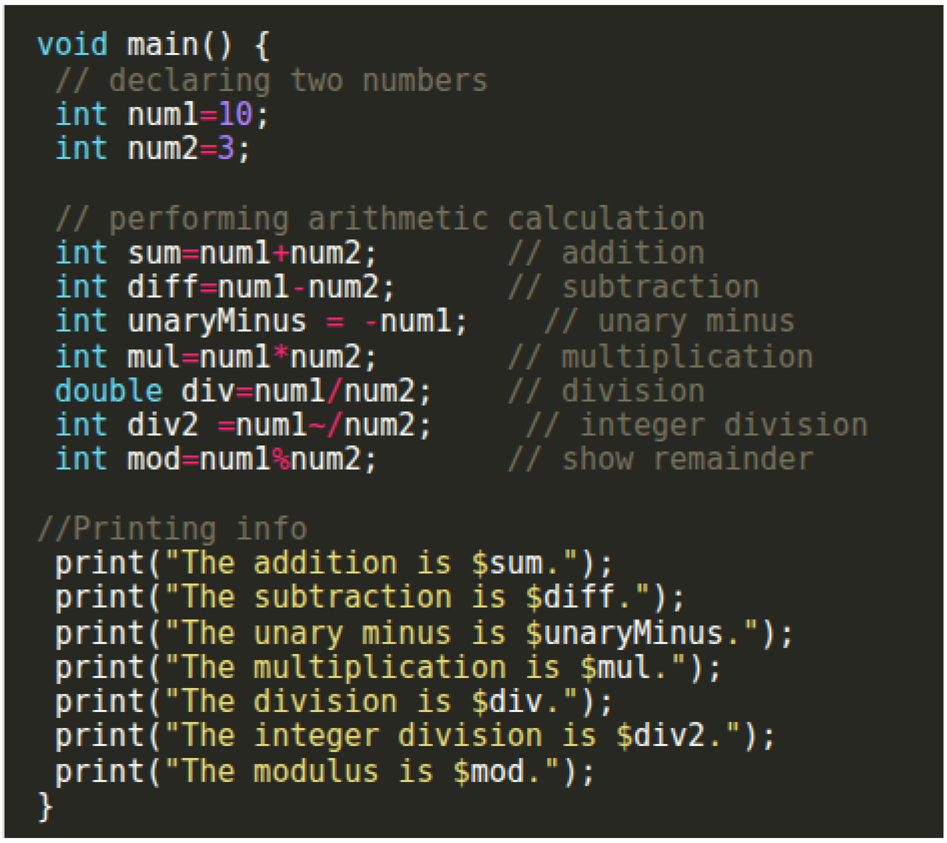
Types Of Operators

There are different types of operators in dart. They are as follows:

- Arithmetic Operators
- Increment and Decrement Operators
- Assignment Operators
- Logical Operators
- Type Test Operators

Arithmetic Operators

Operator Symbol	Operator Name	Description
`+`	Addition	For adding two operands
`-`	Subtraction	For subtracting two operands
`-expr`	Unary Minus	For reversing the sign of the expression
`*`	Multiplication	For multiplying two operands
`/`	Division	For dividing two operands and give output in double
`~/`	Integer Division	For dividing two operands and give output in integer
`%`	Modulus	Remainder After Integer Division



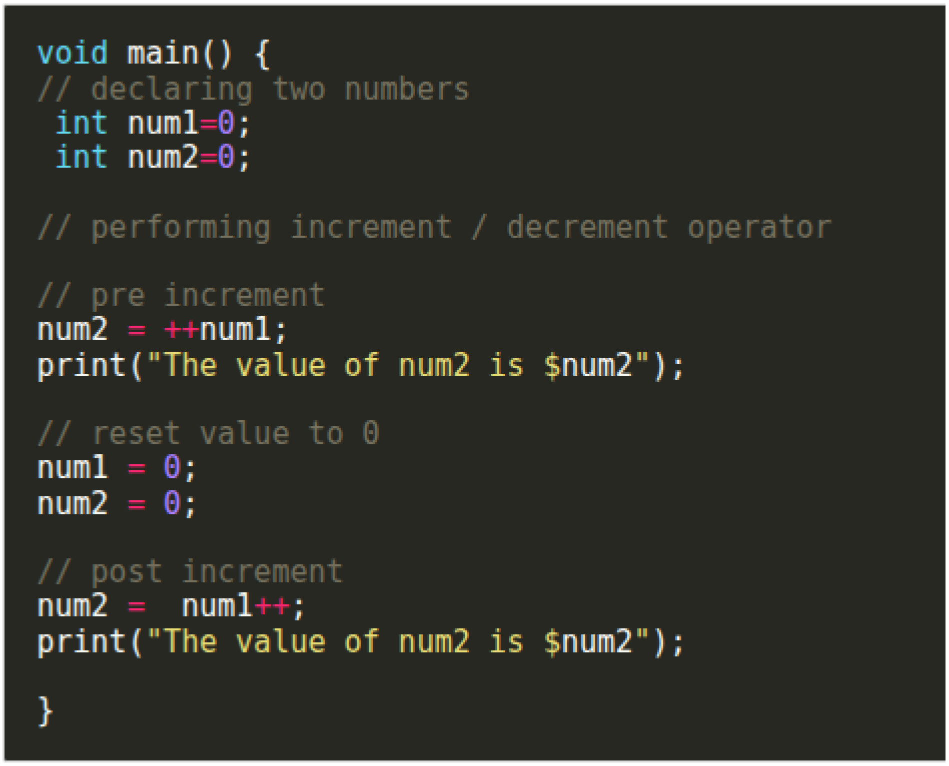
```
void main() {
  // declaring two numbers
  int num1=10;
  int num2=3;

  // performing arithmetic calculation
  int sum=num1+num2;         // addition
  int diff=num1-num2;        // subtraction
  int unaryMinus = -num1;      // unary minus
  int mul=num1*num2;         // multiplication
  double div=num1/num2;      // division
  int div2 =num1~/num2;       // integer division
  int mod=num1%num2;         // show remainder

 //Printing info
  print("The addition is $sum.");
  print("The subtraction is $diff.");
  print("The unary minus is $unaryMinus.");
  print("The multiplication is $mul.");
  print("The division is $div.");
  print("The integer division is $div2.");
  print("The modulus is $mod.");
 }
```

Increment and Decrement Operators

With increment and decrement operators, you can increase and decrease values. If ++ is used at the beginning, then it is a prefix. If it is used at last, then it is postfix.

Operator Symbol	Operator Name	Description
`++var`	Pre Increment	Increase Value By 1. var = var + 1 Expression value is var+1
`--var`	Pre Decrement	Decrease Value By 1. var = var - 1 Expression value is var-1
`var++`	Post Increment	Increase Value By 1. var = var + 1 Expression value is var
`var--`	Post Decrement	Decrease Value By 1. var = var - 1 Expression value is var

++var increases the value of operands, whereas var++ returns the actual value of operands before the increment.

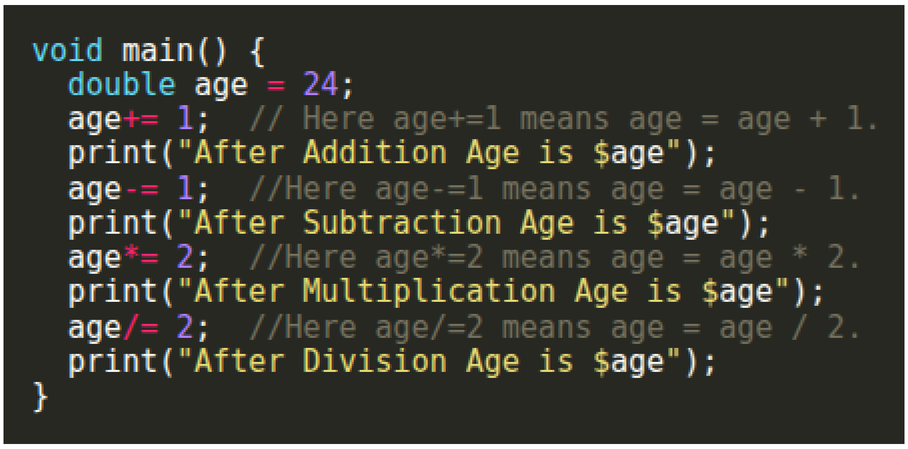


```
void main() {
// declaring two numbers
 int num1=0;
 int num2=0;

// performing increment / decrement operator

// pre increment
num2 = ++num1;
print("The value of num2 is $num2");

// reset value to 0
num1 = 0;
num2 = 0;

// post increment
num2 =  num1++;
print("The value of num2 is $num2");

}
```

Assignment Operators

It is used to assign some values to variables. Here, we are assigning 24 to the age variable.

Operator Type	Description
`=`	Assign a value to a variable
`+=`	Adds a value to a variable
`-=`	Reduces a value to a variable
`*=`	Multiply value to a variable
`/=`	Divided value by a variable

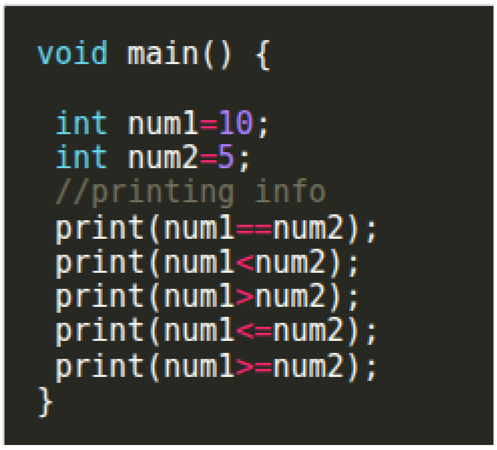


```
void main() {
  double age = 24;
  age+= 1;  // Here age+=1 means age = age + 1.
  print("After Addition Age is $age");
  age-= 1;  //Here age-=1 means age = age - 1.
  print("After Subtraction Age is $age");
  age*= 2;  //Here age*=2 means age = age * 2.
  print("After Multiplication Age is $age");
  age/= 2;  //Here age/=2 means age = age / 2.
  print("After Division Age is $age");
}
```

Relational Operators

Relational operators are also called comparison operators. They are used to make a comparison.

Operator Symbol	**Operator Name**	**Description**
`>`	Greater than	Used to check which operand is bigger and gives result as boolean
`<`	Less than	Used to check which operand is smaller and gives result as boolean
`>=`	Greater than or equal to	Used to check which operand is bigger or equal and gives result as boolean
`<=`	Less than or equal to	Used to check which operand is smaller or equal and gives result as boolean
`==`	Equal to	Used to check operands are equal to each other and gives result as boolean
`!=`	Not equal to	Used to check operand are not equal to each other and gives result as boolean

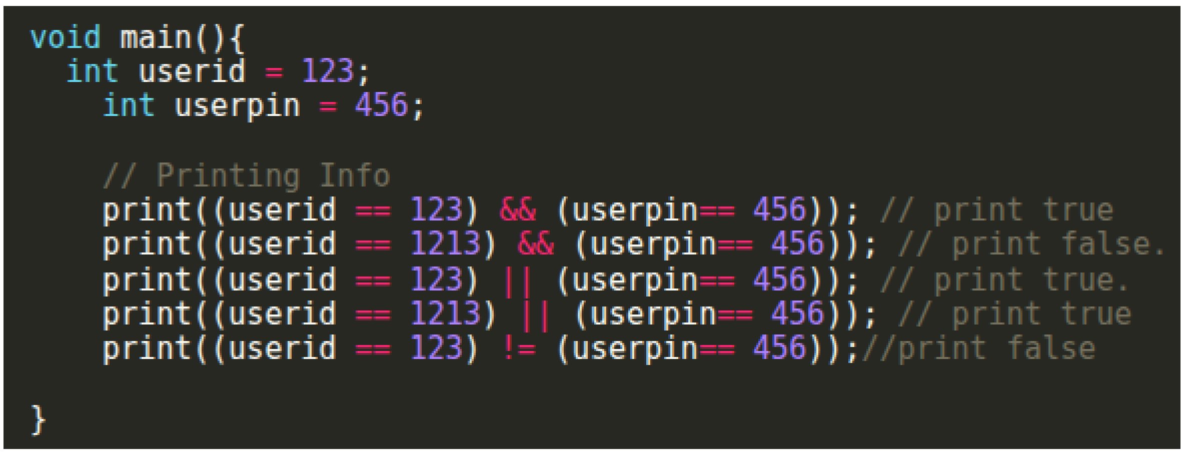


```
void main() {

  int num1=10;
  int num2=5;
  //printing info
  print(num1==num2);
  print(num1<num2);
  print(num1>num2);
  print(num1<=num2);
  print(num1>=num2);
 }
```

Logical Operators

It is used to compare values.

Operator Type	**Description**
`&&`	This is ‘and’, return true if all conditions are true
`||`	This is ‘or’. Return true if one of the conditions is true
`!`	This is ’not’. return false if the result is true and vice versa

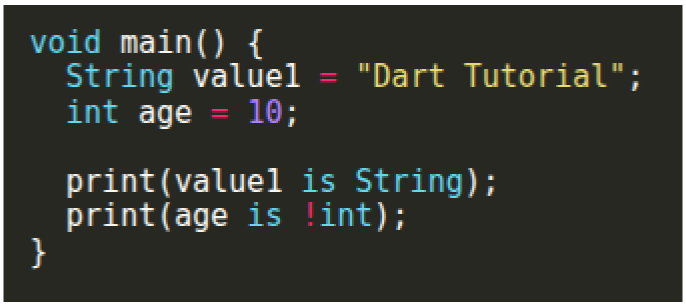


```
void main(){
  int userid = 123;
    int userpin = 456;

    // Printing Info
    print((userid == 123) && (userpin== 456)); // print true
    print((userid == 1213) && (userpin== 456)); // print false.
    print((userid == 123) || (userpin== 456)); // print true.
    print((userid == 1213) || (userpin== 456)); // print true
    print((userid == 123) != (userpin== 456));//print false

}
```

Type Test Operators

In Dart, type test operators are useful for checking types at runtime.

Operator Symbol	Operator Name	Description
`is`	is	Gives boolean value true if the object has a specific type
`is!`	is not	Gives boolean value false if the object has a specific type



```
void main() {
  String value1 = "Dart Tutorial";
  int age = 10;

  print(value1 is String);
  print(age is !int);
}
```