**Assignment # 3**

1. What are the various types of operators in dart? Explain with Examples.

Types of Operator In Dart:

* Arithmetic Operators
* Equality and Relational Operators
* Type test Operators
* Bitwise Operators
* Assignment Operators
* Logical Operators

Arithmetic Operators

The following table shows the arithmetic operators supported by Dart.

|  |  |  |
| --- | --- | --- |
| S. No | Operators | Meaning |
| 1 | **+** | Add |
| 2 | **−** | Subtract |
| 4 | **\*** | Multiply |
| 5 | **/** | Divide |
| 6 | **~/** | Divide, returning an integer result |
| 7 | **%** | Get the remainder of an integer division (modulo) |
| 8 | **++** | Increment |
| 9 | **--** | Decrement |

Examples:

void main() {

var num1 = 101;

var num2 = 2;

var res = 0;

res = num1+num2;

print("Addition: ${res}");

res = num1-num2;

print("Subtraction: ${res}");

res = num1\*num2;

print("Multiplication: ${res}");

res = num1/num2;

print("Division: ${res}");

res = num1~/num2;

print("Division returning Integer: ${res}");

res = num1%num2;

print("Remainder: ${res}");

num1++;

print("Increment: ${num1}");

num2--;

print("Decrement: ${num2}");

}

Equality and Relational Operators

Relational Operators tests or defines the kind of relationship between two entities. Relational operators return a Boolean value i.e. true/ false.

Assume the value of A is 10 and B is 20

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| > | Greater than | (A > B) is False |
| < | Lesser than | (A < B) is True |
| >= | Greater than or equal to | (A >= B) is False |
| <= | Lesser than or equal to | (A <= B) is True |
| == | Equality | (A==B) is False |
| != | Not equal | (A!=B) is True |

Examples:-

void main() {

var num1 = 5;

var num2 = 9;

var res = num1>num2;

print('num1 greater than num2 :: ' +res.toString());

res = num1<num2;

print('num1 lesser than num2 :: ' +res.toString());

res = num1 >= num2;

print('num1 greater than or equal to num2 :: ' +res.toString());

res = num1 <= num2;

print('num1 lesser than or equal to num2 :: ' +res.toString());

res = num1 != num2;

print('num1 not equal to num2 :: ' +res.toString());

res = num1 == num2;

print('num1 equal to num2 :: ' +res.toString());

}

Bitwise Operators

The following table lists the bitwise operators available in Dart and their role −

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| Bitwise AND | a & b | Returns a one in each bit position for which the corresponding bits of both operands are ones. |
| Bitwise OR | a | b | Returns a one in each bit position for which the corresponding bits of either or both operands are ones. |
| Bitwise XOR | a ^ b | Returns a one in each bit position for which the corresponding bits of either but not both operands are ones. |
| Bitwise NOT | ~ a | Inverts the bits of its operand. |
| Left shift | a ≪ b | Shifts a in binary representation b (< 32) bits to the left, shifting in zeroes from the right. |
| Signpropagating right shift | a ≫ b | Shifts a in binary representation b (< 32) bits to the right, discarding bits shifted off. |

Examples:

void main() {

var a = 2; // Bit presentation 10

var b = 3; // Bit presentation 11

var result = (a & b);

print("(a & b) => ${result}");

result = (a | b);

print("(a | b) => ${result}");

result = (a ^ b);

print("(a ^ b) => ${result}");

result = (~b);

print("(~b) => ${result}");

result = (a < b);

print("(a < b) => ${result}");

result = (a > b);

print("(a > b) => ${result}");

}

Assignment Operators

The following table lists the assignment operators available in Dart.

|  |  |  |
| --- | --- | --- |
| S. No | Operator | Operator & Description |
| 1 | **=** | **(Simple Assignment )**  Assigns values from the right side operand to the left side operand  **Ex**: C = A + B will assign the value of A + B into C |
| 2 | **??=** | Assign the value only if the variable is null |
| 3 | **+=** | **(Add and Assignment)**  It adds the right operand to the left operand and assigns the result to the left operand.  **Ex**: C += A is equivalent to C = C + A |
| 4 | **-=** | **(Subtract and Assignment)**  It subtracts the right operand from the left operand and assigns the result to the left operand.  **Ex**: C -= A is equivalent to C = C – A |
| 5 | **\*=** | **(Multiply and Assignment)**  It multiplies the right operand with the left operand and assigns the result to the left operand.  **Ex**: C \*= A is equivalent to C = C \* A |
| 6 | **/=** | **(Divide and Assignment)**  It divides the left operand with the right operand and assigns the result to the left operand |

Example:

void main() {

var a = 12;

var b = 3;

a+=b;

print("a+=b : ${a}");

a = 12; b = 13;

a-=b;

print("a-=b : ${a}");

a = 12; b = 13;

a\*=b;

print("a\*=b' : ${a}");

a = 12; b = 13;

a/=b;

print("a/=b : ${a}");

a = 12; b = 13;

a%=b;

print("a%=b : ${a}");

}

Logical Operators

Logical operators are used to combine two or more conditions. Logical operators return a Boolean value. Assume the value of variable A is 10 and B is 20.

|  |  |  |
| --- | --- | --- |
| Operator | Description | Example |
| && | **And** − The operator returns true only if all the expressions specified return true | (A > 10 && B > 10) is False. |
| || | **OR** − The operator returns true if at least one of the expressions specified return true | (A > 10 || B > 10) is True. |
| ! | **NOT** − The operator returns the inverse of the expression’s result. For E.g.: !(7>5) returns false | !(A > 10) is True. |

Example # 1

void main() {

var a = 10;

var b = 12;

var res = (a<b)&&(b>10);

print(res);

}

Example # 2

void main() {

var a = 10;

var b = 12;

var res = (a>b)||(b<10);

print(res);

var res1 =!(a==b);

print(res1);

}

1. What is a difference between these operators “?? And?”

Conditional Expressions

Dart has two operators that let you evaluate expressions that might otherwise require ifelse statements −

condition ? expr1 : expr2

If condition is true, then the expression evaluates **expr1** (and returns its value); otherwise, it evaluates and returns the value of **expr2**.

expr1 ?? expr2

If **expr1** is non-null, returns its value; otherwise, evaluates and returns the value of **expr2**

1. What are the data types supported in Dart? Explain with Examples.

Data Types

In Dart language, there is the type of values that can be represented and manipulated in a programming language. The data type classification is as given below:

|  |  |  |
| --- | --- | --- |
| Data Type | Keyword | Description |
| Number | int, double, num | Numbers in Dart are used to represent numeric literals |
| Strings | String | Strings represent a sequence of characters |
| Booleans | bool | It represents Boolean values true and false |
| Lists | List | It is an ordered group of objects |
| Maps | Map | It represents a set of values as key-value pairs |

Examples = Number : -

**void** main() {

   // declare an integer

**int** num1 = 2;

   // declare a double value

**double** num2 = 1.5;

   // print the values

   print(num1);

   print(num2);

   var a1 = num.parse("1");

   var b1 = num.parse("2.34");

   var c1 = a1+b1;

   print("Product = ${c1}");

}

Examples = String : -

**void** main() {

    String string = 'Geeks''for''Geeks';

    String str = 'Coding is ';

    String str1 = 'Fun';

    print (string);

    print (str + str1);

}

Examples = Boolean: -

**void** main() {

  String str = 'Coding is ';

  String str1 = 'Fun';

**bool** val = (str==str1);

  print (val);

}

Examples = List: -

**void** main()

{

    List gfg = **new** List(3);

    gfg[0] = 'Geeks';

    gfg[1] = 'For';

    gfg[2] = 'Geeks';

    print(gfg);

    print(gfg[0]);

}

Examples =Map: -

**void** main() {

  Map gfg = **new** Map();

  gfg['First'] = 'Geeks';

  gfg['Second'] = 'For';

  gfg['Third'] = 'Geeks';

  print(gfg);

}

1. Find 5 new methods of List and String.

Method of List and String

Useful List methods in Dart

* sublist():
* shuffle():
* reversed:
* asMap():
* whereType():
* getRange():
* replaceRange():
* firstWhere():