**DATATYPES AND FUNCTION**

**Datatypes in Dart**

A data type is an attribute of data, which tells the compiler, or interpreter how the programmer intends to use the data.

Dart supports the following data types:

* Number
* Strings
* Boolean
* Lists
* Maps
* Runes
* Null

**Numbers**

When you need to store numeric value on dart, you can use either int or double. Both int and double are subtypes of num. Int stores whole numbers while double stores decimal numbers.

Example:

void main() {

// Declaring Variables

int num1 = 100; // without decimal point.

double num2 = 130.2; // with decimal point.

num num3 = 50;

num  num4 = 50.4;

// For Sum

num sum = num1 + num2 + num3 + num4;

// Printing Info

print("Num 1 is $num1");

print("Num 2 is $num2");

print("Num 3 is $num3");

print("Num 4 is $num4");

print("Sum is $sum");

}

**String**

String helps you to store text data in your program. You can use single or double quotes to store string in dart.

Example

void main() {

// Declaring Values

String schoolName = "Powerlearn Project Academy";

String address = "AFRICA";

// Printing Values

print("My School name is $schoolName and the address is $address");

}

/\* Output:

My School name is Powerlearn Project Academy and the address is AFRICA

\*/

**Booleans**

In Dart, boolean holds either true or false value. You can write the bool keyword to define the boolean data type. You can use boolean if the answer is true or false. Consider the answer to the following questions:

* Are you asleep?
* Is the door open?
* Does a cat fly?
* Are you older than your father?

These all are yes/no questions. Its a good idea to store them in boolean.

void main() {

bool isMarried = true;

print("Married Status: $isMarried");

}

/\* Output:

Married Status: true

\*/

**Lists**

Dart List is similar to an array, which is the ordered collection of the objects. If you want to store multiple values without creating multiple variables, you can use a list.

void main() {

List<String> names = ["John", "James", "Peter"];

print("Value of names is $names");

print("Value of names[0] is ${names[0]}"); // index 0

print("Value of names[1] is ${names[1]}"); // index 1

print("Value of names[2] is ${names[2]}"); // index 2

print(names);

}

/\* Output:

Value of names is [John, James, Peter]

Value of names[0] is John

Value of names[1] is James

Value of names[2] is Peter

[John, James, Peter]

\*/

**Maps**

A map is a dynamic collection that represents a set of values ​as key-value pairs. Keys and values ​in the map can be of any type.

void main() {

// Creating a Map with String keys and int values

Map<String, int> ages = {'Alice': 30,

'Bob': 25,

'Charlie': 35,

};

print("Ages of students: $ages");

}

/\* Output:

 Ages of students: {Alice: 30, Bob: 25, Charlie: 35}

\*/

**Runes**

A rune can be defined as an integer used to represent any Unicode code point. As a Dart string is a simple sequence of UTF-16 code units, 32-bit Unicode values in a string are represented using a special syntax.

void main() {

  // Define a string with runes

  String runesString = "Runes in Dart: \u{1F600} \u{1F64B} \u{1F680}";

  // Print the string

  print(runesString);

  /\* Output

      Runes in Dart: 😀 🙋 🚀

  \*/

}

**Arithmetic Operations using Numbers**

Arithmetic operators are the most common types of operators. They perform operations like addition, subtraction, multiplication, 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 subtract = num2-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 $subtract");

 print("The multiplication is $mul.");

 print("The division is $div.");

 print("The integer division is $div2.");

 print("The modulus is $mod.");

}

/\* OUTPUT:

The addition is 13.

The subtraction is 7.

The unary minus is -7

The multiplication is 30.

The division is 3.3333333333333335.

The integer division is 3.

The modulus is 1.

\*/

**Functions in Dart**

Functions are the block of code that performs a specific task. They are created when some statements are repeatedly occurring in the program. The function helps reusability of the code in the program.

**Function Advantages**

* Avoid Code Repetition
* Easy to divide the complex program into smaller parts
* Helps to write a clean code

**Syntax of a function**

returntype functionName(parameter1,parameter2, ...){

// function body

}

**Return type**: It tells you the function output type. It can be void, String, int, double, etc. If the function doesn’t return anything, you can use void as the return type.

**Function Name**: You can name functions by almost any name. Always follow a lowerCamelCase naming convention like void printName().

**Parameters**: Parameters are the input to the function, which you can write inside the bracket (). Always follow a lowerCamelCase naming convention for your function parameter.

**Example 1**: Function That Prints Name

This is a simple program that prints name using function. The name of function is printName().

// writing function outside main function.

void printName(){

  print("My name is John James");

}

// this is our main function.

void main(){

  printName();

}

/\*

OUTPUT:

My name is John James

\*/

**Example 2**: Function To Find Sum of Two Numbers

This function finds the sum of two numbers. Here, the function accepts two parameters. i.e., num1 and num2, and the return type is void.

void add(int num1, int num2){

  int sum = num1 + num2;

   print("The sum is $sum");

}

void main(){

  add(10, 20);

}

/\*

OUTPUT:

The sum is 30

\*/

**Key Points**

* In dart, functions are also objects.
* You should follow the lowerCamelCase naming convention while naming function.
* You should follow the lowerCamelCase naming convention while naming function parameters.

**Types of Functions**

Functions are the block of code that performs a specific task. Here are different types of functions:

* No Parameter And No Return Type
* Parameter And No Return Type
* No Parameter And Return Type
* Parameter And Return Type

**Function With No Parameter And No Return Type**

In this function, you do not pass any parameter and expect no return type. Here is an example of it:

void main() {

  printName();

}

void printName() {

  print("My name is John James.");

}

/\*

OUTPUT:

My name is John James.

\*/

Here printName() is a function which prints name on your screen.

**Function With Parameter And No Return Type**

In this function, you do pass the parameter and expect no return type. Here is an example of it:

Here printName(String name) is a function which welcome person

void main() {

  printName("John");

}

void printName(String name) {

  print("Welcome, ${name}.");

}

/\*

OUTPUT:

Welcome, John.

\*/

In this program, printName(String name) is the function which has keyword void. It means it has no return type, and the pair of parentheses is not empty but this time that suggests it to accept an parameter.

**Function With No Parameter And Return Type**

In this function, you do not pass any parameter but expect return type. Here is an example of it:

Here InstructorName() is a function which returns Instructor's name. In the entire program, anyone can use this function to find the name of the Instructor.

void main() {

// Function With No Parameter & Return Type

  String name = InstructorsName();

  print("The Name from function is $name.");

}

String InstructorsName() {

  return "Allan Lenkaa";

}

/\*

OUTPUT:

The Name from function is Allan Lenkaa.

\*/

In this program, InstructorsName() is the function which has String keyword before function name, means it return String value, and the empty pair of parentheses suggests that there is no parameter that is passed to the function.

**Function With Parameter And Return Type**

In this function, you do pass the parameter and also expect return type. Here is an example of it:

// this function add two numbers

int add(int a, int b) {

  int sum = a + b;

  return sum;

}

void main() {

  int num1 = 10;

  int num2 = 20;

  int total = add(num1, num2);

  print("The sum is $total.");

}

/\*

OUTPUT:

The sum is 30.

\*/

In this program, int add(int a, int b) is the function with int as the return type, and the pair of parenthesis has two parameters, i.e., a and b.

THE TYPES OF FUNCTIONS DISCUSSED

The code snippets below indicate all the functions. Use the comments as your guidelines.

// parameter and return type

int add(int a, int b) {

  var total;

  total = a + b;

  return total;

}

// parameter and no return type

void mul(int a, int b) {

  var total;

  total = a \* b;

  print("Multiplication is : $total");

}

// no parameter and return type

String greet() {

  String greet = "Welcome";

  return greet;

}

// no parameter and no return type

void greetings() {

  print("Hello World!!!");

}

void main() {

  var total = add(2, 3);

  print("Total sum: $total");

  mul(2, 3);

  var greeting = greet();

  print("Greeting: $greeting");

  greetings();

}

/\*

OUTPUT:

Total sum: 5

Multiplication is : 6

Greeting: Welcome

Hello World!!!

\*/

**Anonymous Functions**

Nameless functions or functions without a name. This type of function is known as an **anonymous function**, **lambda**, or **closure**. An anonymous function behaves the same as a regular function, but it does not have a name with it. It can have zero or any number of arguments / parameters with an option type annotation.

**Syntax**

Below is the syntax of the anonymous function

(parameter){

  // body of the function

}

**Knowledge Panel:**

* You can assign an anonymous function to a variable
* You can pass an anonymous function as a parameter / argument

**Example 1**:

In this example, you will learn to use an anonymous function to print all list items. This function invokes each fruit without having a function name.

void main() {

  const fruits = ["Apple", "Mango", "Banana", "Orange"];

  fruits.forEach((fruit) {

    print(fruit);

  });

}

/\*

OUTPUT:

Apple

Mango

Banana

Orange

\*/

**Example 2:**

In this example, we will use an anonymous function to print all list items.

void main() {

  // list of cars

  List cars = ["BMW", "BENZ", "AUDI", "TOYOTA"];

  // iteration with anonymous function as a parameter

  cars.forEach((car) {

    print(car); // printing an item

  });

}

**Arrow Function**

If you want to declare a function in one line; In Dart we have a fat arrow function that can enable you. The function is represented by **=>** symbol.

**Syntax**

Below is the syntax for the arrow function

returnType functionName(parameters) => expression;

**Knowledge Panel**

Note: The arrow function is used to make your code short. **=>** expr syntax is a shorthand for { return expr; }.

**Example 1**: Calculation of simple interest without Arrow Function

This program finds simple interest without using the arrow function.

// function that calculate interest

double calculateInterest(double principal, double rate, double time) {

  double interest = principal \* rate \* time / 100;

  return interest;

}

void main() {

  double principal = 5000;

  double time = 3;

  double rate = 3;

  double result = calculateInterest(principal, rate, time);

  print("The simple interest is $result.");

}

/\*

  OUTPUT:

  The simple interest is 450.

\*/

**Example 2**: Calculation of simple interest WITH Arrow Function

// arrow function that calculate interest

double calculateInterest(double principal, double rate, double time) =>

    principal \* rate \* time / 100;

void main() {

  double principal = 5000;

  double time = 3;

  double rate = 3;

  double result = calculateInterest(principal, rate, time);

  print("The simple interest is $result.");

}