DART PROGRAMMING - FUNCTION

THANAKORN YARNGUY

DART-FUNCTION

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

The main objective of the function is **DRY(Don't Repeat Yourself).**

DART-FUNCTION

syntax

```
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.

DART-FUNCTION

example

```
void add(int num1, int num2){
  int sum = num1 + num2;
   print("The sum is $sum");
}
void main(){
  add(10, 20);
}
```

DART-FUNCTION

Key Points

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

DART-FUNCTION

Types Of Function

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

DART-FUNCTION

No Parameter & No Return Type

```
void main() {
  printName();
}

void printName() {
  print("My name is John Doe.");
}
```

DART-FUNCTION

Function With Parameter And No Return Type

```dart
void main() {
  printName("John");
}

void printName(String name) {
  print("Welcome, ${name}.");
}
```

DART-FUNCTION

Function With No Parameter And Return Type

```
// Function With No Parameter & Return Type
void main() {
  String name = primeMinisterName();
  print("The Name from function is $name.");
}

String primeMinisterName() {
  return "John Doe";
}
```

DART-FUNCTION

Function With Parameter And Return Type

```
// 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.");
}
```

DART-FUNCTION

Parameter In Dart

Positional Parameter In Dart

In positional parameters, you must supply the arguments in the same order as you defined on parameters

```
void printInfo(String name, String gender, [String title =
"sir/ma'am"]) {
  print("Hello $title $name your gender is $gender.");
}

void main() {
  printInfo("John", "Male");
  printInfo("John", "Male", "Mr.");
  printInfo("Kavya", "Female", "Ms.");
}
```

DART-FUNCTION

Named Parameter In Dart

Dart allows you to use named parameters to clarify the parameter's meaning in function calls. **Curly braces {}** are used to specify named parameters.

```
void printInfo({String? name, String? gender}) {
  print("Hello $name your gender is $gender.");
}

void main() {
  // you can pass values in any order in named parameters.
  printInfo(gender: "Male", name: "John");
  printInfo(name: "Sita", gender: "Female");
  printInfo(name: "Reecha", gender: "Female");
  printInfo(name: "Reecha", gender: "Female");
  printInfo(name: "Harry", gender: "Male");
  printInfo(gender: "Male", name: "Santa");
}
```

DART-FUNCTION

Use Of Required In Named Parameter

function **printInfo** takes two named parameters. You can see a **required** keyword, which means you must pass the person's name and gender. If you don't pass it, it won't work.

```
void printInfo({required String name, required String
gender}) {
  print("Hello $name your gender is $gender.");
}

void main() {
  // you can pass values in any order in named parameters.
  printInfo(gender: "Male", name: "John");
  printInfo(gender: "Female", name: "Suju");
}
```

DART-FUNCTION

Optional Parameter In Dart

Dart allows you to use optional parameters to make the parameter optional in function calls. **Square braces []** are used to specify optional parameters.

```
void printInfo(String name, String gender, [String? title]) {
  print("Hello $title $name your gender is $gender.");
}

void main() {
  printInfo("John", "Male");
  printInfo("John", "Male", "Mr.");
  printInfo("Kavya", "Female", "Ms.");
}
```

DART-FUNCTION

Anonymous Function In Dart

not every function needs a name. If you remove the return type and
the function name, the function is called **anonymous function.**

Syntax

```
(parameterList){
// statements
}
```

```
void main() {
// Anonymous function
  var cube = (int number) {
    return number * number * number;
  };

  print("The cube of 2 is ${cube(2)}");
  print("The cube of 3 is ${cube(3)}");
}
```

DART-FUNCTION

Arrow Function In Dart

Dart has a special syntax for the function body, which is only one line. The arrow function is represented by **=>** symbol. It is a shorthand syntax for any function that has only one expression.

<u>Syntax</u>

returnType functionName(parameters...) => expression;

```
int add(int n1, int n2) => n1 + n2;
int sub(int n1, int n2) => n1 - n2;
int mul(int n1, int n2) => n1 * n2;
double div(int n1, int n2) => n1 / n2;

void main() {
  int num1 = 100;
  int num2 = 30;

  print("The sum is ${add(num1, num2)}");
  print("The diff is ${sub(num1, num2)}");
  print("The mul is ${mul(num1, num2)}");
  print("The div is ${div(num1, num2)}");
}
```

DART-FUNCTION

Scope In Dart

The scope is a concept that refers to where values can be accessed or referenced. Dart uses curly braces **{}** to determine the scope of variables.

Global Scope

You can define a variable in the global scope to use the variable anywhere in your program.

```
String global = "I am Global. Anyone can access me.";
void main() {
  print(global);
}
```