SNS COLLEGE OF TECHNOLOGY

Coimbatore-35
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A++' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER APPLICATIONS

23CAT607- CROSS-PLATFORM APP DEVELOPMENT

I YEAR II SEM

UNIT 2 – **FLUTTER BASICS**

TOPIC 5 – DART Functions

Dart Function

Dart function is a set of codes that together perform a specific task.

Use the functions multiple times by calling.

The function provides the flexibility to run a code several times with different values. A function can be called anytime as its parameter and returns some value to where it called.

Advantages of Functions

1. It increases the module approach to solve the problems.
2. It enhances the re-usability of the program.
3. We can do the coupling of the programs.
4. It optimizes the code.
5. It makes debugging easier.
6. It makes development easy and creates less complexity.

Defining a Function

A function can be defined by providing the name of the function with the appropriate parameter and return type. A function contains a set of statements which are called function body.

Syntax:

```
return_type func_name (parameter_list):
{
  //statement(s)
 return value;
}
```

Denotes the list of the parameter

Appropriate and valid identifier.

Data type such as void, integer, float, etc

Returns a value after complete its execution.

Calling a Function

Example - 1

```
int mul(int a, int b){
    int c;
    c = a+b;
    print("The sum is:${c}");
}
```

Invoke the defined function inside the main() function body

Syntax:

fun_name(<argument_list>);
or
variable = function_name(argument);

Example :
mul(10,20);

Passing Arguments to Function

When a function is called, it may have some information as per the function prototype is known as a parameter (argument).

Actual Parameter - A parameter which is passed during a function definition is called the actual parameter.

Formal Parameter - A parameter which is passed during a function call is called the formal parameter.

Return a Value from Function

A function always returns some value as a result to the point where it is called. The return keyword is used to return a value.

Syntax: **return** <expression/values>

Example - **return** result;

Dart Function with parameter and return value

```
void main() {
  print("Example of add two number using the function");
  // Creating a Function

  int sum(int a, int b){
          // function Body
          int result;
          result = a+b;
          return result;
}
// We are calling a function and storing a result in variable c
var c = sum(30,20);
print("The sum of two numbers is: ${c}");
}
```

```
Example of add two number using the function
The sum of two numbers is: 50
```

Output

Dart Function with No Parameter and Return Value

Syntax:

```
return_type func_name()
{
      //Statement(s);
      return value;
}
```

Example

```
void main(){
// Creating a function without argument
String greetings(){
  return "Welcome to Function";
}
// Calling function inside print statement

print(greetings());
}
```

Output

```
Welcome to Function
```

Dart Function with No Parameter and without a Return Value

Syntax:

```
func_name() {
 //statement
}
Or
void fun_name() {
  //statement(s)
}
```

The function has no return type.

Example - 3

```
// Creating a function without argument
void greetings()
{
  print("Welcome to JavaTpoint");
}
void main() {
 print("The example of Dart Function");
 // function callling
 greetings();
}
```

Output

```
The example of Dart Function Welcome to JavaTpoint
```

The main() function

The main() function is the top-level function of the Dart. It is the most important and vital function of the Dart programming language. The execution of the programming starts with the main() function. The main() function can be used only once in a program.

Syntax:

```
void main() {
  // main function body
}
```

```
void main()
{
 print("Welcome To JavaTpoint");

 }
```

Output

```
Welcome To JavaTpoint
```

Dart Return Value

The return keyword is used to represent the return statement. If the return statement not specified, then the function returns null.

Syntax:

```
return <expression/value>;
```

Dart value with Return Value

Syntax:

```
return_type function_name()
{
  //statement(s);
 return value;
}
```

Example -

```
void main() {
 int mul(int a, int b){
     int c = a*b;
     return c;
}
print("The multiplication of two numbers: ${mul(10,20)}");
}
```

Output

```
The multiplication of two numbers: 200
```

Dart Recursion

Dart Recursion is the method where a function calls itself as its subroutine. It is used to solve the complex problem by dividing it into sub-part. A function which is called itself again and again or recursively, then this process is called recursion.

Base condition in recursion

```
void main() {
  int factorial(int num){

 if(num<=1) { // base case
      return 1;
 else{
       return n*fact(n-1);
 }
}
}
}
```

Syntax:

```
void recurse() {
  //statement(s)
 recurse();
//statement(s);
}
void main(){
   //statement(s)
  recurse();
 //statement(s)
}
```

Example

```
int factorial(int num){

 //base case of recursion.
 if(num<=1) { // base case
     return 1;
}
 else{
                                  return    num*factorial(num-
1);   //function call itself.
 }
}
void main() {
 var num = 5;
 // Storing function call result in fact variable.
 var fact = factorial(num);
 print("Factorial Of 5 is: ${fact}");
}
```

Output:

```
Factorial Of 10 is: 120
```

THANK YOU