Optional Parameters

When you want to leave the passing of arguments to functions as an option, you can use optional parameters. In this lesson, we will learn how to declare functions with optional parameters.

We'll cover the following

A function can have two types of parameters: **required** and **optional**. The required parameters are listed first, followed by any optional parameters. Optional parameters can be **named** or **positional**.

> Optional parameters can **either** be *named* or *positional*, but not both.

Named Parameters

In a function, named parameters are declared using curly brackets (`{}`).

```
(requiredParam, {optionalParam1, optionalParam2,...})
```

The optional named parameter is encompassed in curly brackets (`{}`) within the parentheses.

Named parameters are called as such because they need to be named during a function call.

Below is the syntax for specifying named parameters during a function call.

```
parameterName: value
```

In the example below we are printing a number and two strings. The number is a required parameter while the strings are optional named parameters.

```
printer(num n,{String s1, String s2}) {
  print(n);
  print(s1);
  print(s2);
}
```

Let's call the `printer` function while only specifying one of the optional parameters.

Make sure that the parameter name you are specifying is the same as the one you have specified as a parameter in the function declaration.

```
printer(num n,{String s1, String s2}) {
  print(n);
  print(s1);
  print(s2);
}

main() {
  printer(75, s1: 'hello');
}
```

Now, let's call `printer` without specifying any of the optional parameters.

```
printer(num n,{String s1, String s2}) {
  print(n);
  print(s1);
  print(s2);
}

main() {
  printer(75);
}
```

When we don't specify an optional parameter, Dart takes it as `null`.

Finally, below we are calling `printer` while specifying all the optional parameters.

```
printer(num n,{String s1, String s2}) {
  print(n);
  print(s1);
  print(s2);
}

main() {
```

```
  printer(75, s1: 'hello', s2: 'there');
}
```

Positional Parameters

Wrapping a set of function's parameters in square brackets (`[]`) marks them as optional parameters.

The syntax is as follows.

```
(requiredParam, [optionalParam])
```

When calling a function, we pass an optional positional parameter just as we would a required parameter.

In the example below, we have one required parameter and two optional parameters.

```
String mysteryMessage(String who, [String what, String where]){
  var message = '$who';
  if(what != null && where == null){
    message = '$message said $what';
  } else if (where != null){
    message = '$message said $what at $where';
  }
  return message;
}
```

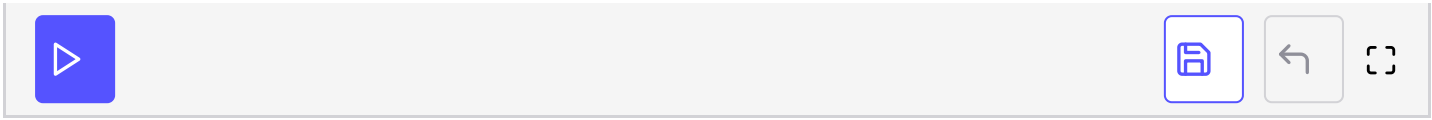
In the code snippet above, our return value is different if the optional parameter is passed during the function call. The purpose of **line 3** is to check if the optional parameter has been specified or not.

Let's call `mysteryMessage`, and only specify the first optional parameter.

```
String mysteryMessage(String who, [String what, String where]){
  var message = '$who';
  if(what != null && where == null){
    message = '$message said $what';
  } else if (where != null){
    message = '$message said $what at $where';
  }
  return message;
}

main() {
```

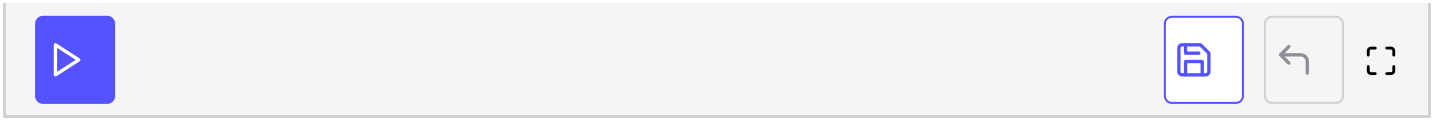
```
main() {
  var result = mysteryMessage('Billy', 'howdy');
  print(result);
}
```



Now, let's call `mysteryMessage` while specifying both optional parameters.

```
String mysteryMessage(String who, [String what, String where]){
  var message = '$who';
  if(what != null && where == null){
    message = '$message said $what';
  } else if (where != null){
    message = '$message said $what at $where';
  }
  return message;
}

main() {
  var result = mysteryMessage('Billy', 'howdy', 'the ranch');
  print(result);
}
```



It is important to mention that one of the main differences between positional and named parameters lies in the fact that you must use positional parameters in the order they are declared. In the example above, we cannot use `where` without using `what`. However, this is not the case with named parameters.

That's it for optional parameters. Let's move onto recursive functions in the next lesson.