The if Statement

In this lesson, we will go over how to use the if statement.

We'll cover the following

Introduction

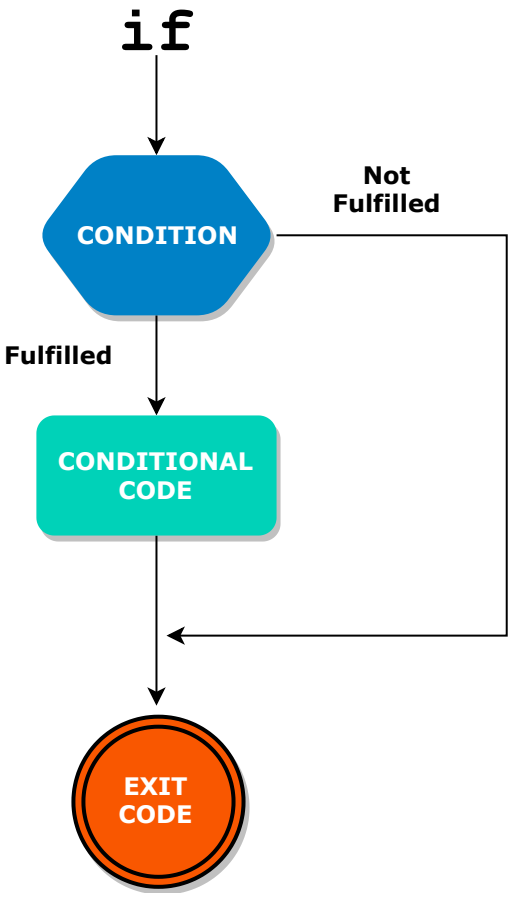
The `if` statement allows you to incorporate conditions in your code which need to be *fulfilled* before the code can execute.

Moving away from the world of computers, let's imagine you wake up in the morning and get ready to go out. When you reach the door, you're not sure if it's raining outside or not. If it is raining outside, you will take an umbrella with you. If it is not raining, you wouldn't want to take an umbrella with you, so you do nothing and just leave. Your final decision is conditional based on the weather outside. This is the same way conditional statements work.

Conditional statements are incredibly powerful as they take the state of the program into consideration and act accordingly. This provides a decision-making capability to the programmer.

Control Flow

Let's look at the control flow of `if`.



The above flow is showing that if the condition has been fulfilled, the compiler will execute the conditional code and if the condition has not been fulfilled, the compiler will exit that block of code without executing the conditional code.

Syntax

Before we see the `if` expression in action, let's go over the syntax and see how to write a block of code with `if` using Dart.

```
if (condition) {
  conditional code
}
```

The syntax starts with the `if` keyword followed by parentheses (`()`). In the parentheses, you would write your condition. For instance, the condition in our real-life example would be *raining outside*. After the closing parentheses (`)`), we insert an opening curly bracket (`{`) after which we go to the next line and write our conditional code. This is code that will execute if the condition holds true. In

our real-life example, this would be *take umbrella*. After we write the conditional

code, we go to the next line and insert the closing curly bracket (`}`) and end our `if` statement.

`if` in Action

In our example below, we want to empty a list only **if** it is not empty.

You can check if a collection is empty using the `isEmpty` and `isNotEmpty` properties. `isEmpty` is `true` when a collection is empty and `isNotEmpty` is `true` when a collection is not empty.

> Remember that properties can be accessed using the dot operator (`.`).

Let's look at how we would code the example in Dart.

```
main() {
  var testList = [2,4,8,16,32];
  print(testList);

  if(testList.isNotEmpty){
    print("Emptying List");
    testList.clear();
  }

  print(testList);
}
```

In the code above, the condition is using the `isNotEmpty` property to check if `testList` is not empty (**line 5**). If it is not empty, then the conditional code is executed which first prints **"Emptying List"** (**line 6**) and then clears all the items contained in `testList` (**line** 7).

> **Try it Yourself:** While the code above is using a non-empty list, try an empty list and see how the output would change.

Let's further explore the `if` statement in the next lesson.