Anonymous Functions

In this lesson, you will be introduced to anonymous functions and learn their syntax.

We'll cover the following

Function Literals

Remember when we learned about literals? They are defined as fixed values appearing directly in the source code. Literals don't need to be named, they can simply be used directly.

There are times when we only need to use functions once, or temporarily; only requiring the functionality of those functions. Naming them is an extra unnecessary step in this scenario.

What we need is something similar to literals. Functions that do not need to be named as their functionality is only required for a single instance.

Dart provides a solution known as **anonymous functions**.

> Anonymous functions are sometimes known as **lambda** functions or **closures**.

Syntax

Let's look at the syntax for an anonymous function.

```
(paramList) {
  functionBody
```

```
}
```

So, if we wanted to write an anonymous function which returns the cube of a number, it would look like this:

```
(x)  {
   x*x*x;
}
```

Here, *x* is the un-typed parameter of the anonymous function and *x* * *x* * *x* is the function body.

Learning by Example

Let's pass our anonymous cube function to the built-in method `forEach`. Our objective is to get the cube of every item in a list.

```
main() {
  var list = [1,2,3];
  list.forEach((item) {
   print(item*item*item);
  });
}
```

The anonymous function is invoked for each item in the list, `list`.

If the function contains only one statement in its function body, you can shorten it using arrow notation as discussed in a previous lesson.

```
main() {
  var list = [1,2,3];
  list.forEach(
    (item) => print(item*item*item));
}
```

In the next lesson, we will learn about nested functions.