

Working with Lists

In this lesson, you will be introduced to some of the properties and methods of a list.

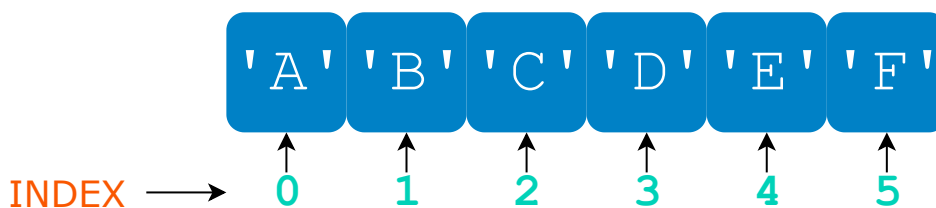
We'll cover the following

- Indexing
- Accessing an Element
- Finding the Length of a List
- Adding a Single Element
- Adding Multiple Elements
- Removing a Single Element
- Removing All Elements
- The map() Method

As discussed, a List is a type of object that has particular properties and particular methods that it can perform. Let's look at some of them below.

Indexing

Lists use zero-based indexing. This means that the first element of a list is located at the **0th** index.



Since each element has its own position, a list can contain duplicates of a single element because each duplicate is still unique in its position. For instance, we can have a list with five identical elements as shown below.



Accessing an Element

To access an element at a particular index we can use square brackets (`[]`).

The general syntax is as follows:

`listName[index]`

Let's look at an example below.

```
main() {
  var listOfVegetables = ['potato', 'carrot', 'cucumber'];

  print(listOfVegetables[1]);
}
```



In the code snippet above, we are accessing the second element of the list `listOfVegetables`, which is the element at the index 1.

Finding the Length of a List

The length of a list is simply the number of elements in that list. To find the length of a list, we can access the `length` property. To access any property we use the dot operator (`.`).

The basic syntax is as follows:

`listName.length`

Let's find the length of our `listOfVegetables`.

```
main() {
  var listOfVegetables = ['potato', 'carrot', 'cucumber'];

  print(listOfVegetables.length);
}
```



When you press RUN, the number 3 should be displayed because `listOfVegetables` has three elements.

Adding a Single Element

We can add a single element to the end of an already existing list using the `add` method. The only condition is that the element you add must be of the same type as the elements of the list.

The `add` method has a single parameter which is the element you want to add to a list. The type of the parameter depends on the list you call the method on.

The general syntax is as follows:

`listName.add(element)`

Let's add another vegetable to our `listOfVegetables`.

```
main() {  
    var listOfVegetables = ['potato', 'carrot', 'cucumber'];  
  
    listOfVegetables.add('cabbage');  
  
    print(listOfVegetables);  
}
```

When you press RUN, you'll see that the updated list now also has *cabbage* as an element.

Adding Multiple Elements

We can add multiple elements to an already existing list using the `addAll` method. Again, the only condition is that the elements you add must all be of the same type as the elements of the list.

The `addAll` method also has a single parameter which is a list. The list should contain the elements you want to add to an already existing list. The type of the parameter is `List<dataType>`, where the data type depends on the list you call the

method on.

In conclusion, `addAll` basically merges the elements of two lists into one.

The general syntax is as follows:

```
listName.addAll([elem 1, elem 2, ..., elem n])
```

OR

```
listName.addAll([otherListName])
```

Let's add some more vegetables to our `listOfVegetables`.

```
main() {  
  var listOfVegetables = ['potato', 'carrot', 'cucumber', 'cabbage'];  
  
  listOfVegetables.addAll(['broccoli', 'zucchini']);  
  
  print(listOfVegetables);  
  
  var vegetablesToAdd = ['okra', 'capsicum'];  
  
  listOfVegetables.addAll(vegetablesToAdd);  
  
  print(listOfVegetables);  
}
```



Removing a Single Element


To remove a single element from an already existing list, we can use the `removeAt` method which removes the element at the specified index.

The `removeAt` method has a single parameter which is the index of the element you want to remove. The type of the parameter is `int`.

The general syntax is as follows:

```
listName.removeAt(index of element to be removed)
```

Let's remove some vegetables from `listOfVegetables`.



```
main() {
  var listOfVegetables = ['potato', 'carrot', 'cucumber', 'cabbage', 'broccoli', 'zucchini'];


  listOfVegetables.removeAt(0);
  print(listOfVegetables);

  listOfVegetables.removeAt(2);
  print(listOfVegetables);
}
```



If you know which element you want to remove but don't know its index, you can use the `indexOf` method we discussed in a previous lesson to find the index of an element.

Let's find the index of 'carrot' and remove it from our `listOfVegetables`.



```
main() {
  var listOfVegetables = ['carrot', 'cucumber', 'zucchini'];

  var carrotIndex = listOfVegetables.indexOf('carrot');
  listOfVegetables.removeAt(carrotIndex);

  print(listOfVegetables);
}
```




The final `listOfVegetables` now only contains two elements, namely 'cucumber' and 'zucchini'.

Removing All Elements

To remove all the elements from a list, we can simply call the `clear` method which takes no parameters.

Let's remove all the elements from `listOfVegetables`.



```
main() {
  var listOfVegetables = ['cucumber', 'zucchini'];

  listOfVegetables.clear();

  print(listOfVegetables);
}
```

}
When you press RUN, an empty list should be displayed.

The `map()` Method

The `List` type has a method known as `map()`, not to be confused with the `Map` collection.

`map()` maps all the items of a list to an expression or statement. For instance, we could have a list of integers and we want to calculate the square of each integer in the list. `map()` could be used to solve such a problem.

Let's look at the syntax below.

`listName.map((iterator) => statement)`

Iterator can have any name. It is basically a variable that takes the value of each item in the list one by one. The iterator starts equal to the first item in the list and will then apply that item to the statement. Then, it will be assigned the second item in the list and apply that item to the statement. This process will continue until there are no more elements in the list.

In the example below, we have our list of vegetables and we want to print "I love vegetable", for each of the items.

The code in Dart is as follows:

```
main() {  
  var listOfVegetables = ['carrot', 'cucumber', 'zucchini'];  
  var mappedVegetables = listOfVegetables.map((vegetable) => 'I love $vegetable');  
  print(mappedVegetables);  
}
```

You might have noticed that the output is not a list, as it does not have square brackets. To transform the result of `map()` to a list we can use the `toList()` method.

Let's modify the code above so that the output is a list.

```
main() {  
  var listOfVegetables = ['carrot', 'cucumber', 'zucchini'];  
  var mappedVegetables = listOfVegetables.map((vegetable) => 'I love $vegetable').toList();  
  print(mappedVegetables);  
}
```



When you run the code above, you will see a list as the output. This was done by using the `toList()` method on **line 3**.

In the next lesson, you will be challenged to use what you've learned about lists.