List: The Dart Array

Lists are probably the most common data structure used in Dart. In the following lesson, you will learn how to create a List.

We'll cover the following

Introduction

Arrays are one of the most common and most popular data structures provided by a programming language. That being said, there are no arrays in Dart. Instead, Dart has **lists** which provide more or less everything an array provides.

Lists are an *ordered* collection of objects. This means that every element in a list has a fixed position. Use a List when you need to access objects by index.

> Lists are of the type `List`.

Creating a List

There are multiple ways to create a list. Let's look at the more common ones below.

Using Literals

The simplest way is using literals along with square brackets (`[]`).

The general syntax is as follows:

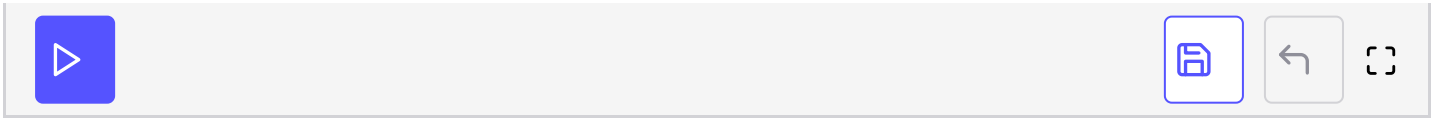
```
var listName = [elem 1, elem 2,...,elem n]
```

We start off with the `var` keyword followed by a unique identifier (`listName`). After the equals sign (`=`), we insert the opening square bracket (`[`) which is followed by the elements we want in our list, with each element being separated by a comma (`,`). After the last element, we insert the closing square bracket (`]`).

Let's look at how this would look in actual code.

```
main() {
  var simpleList = [1,2,3];

  print(simpleList);
}
```



On **line 2** of the code snippet above, we are declaring and initializing a list with three elements.

When we print the list using the simple print method, the complete list, including square brackets, is displayed.

> Remember how we said that `List` is a type when we were discussing data types? Well in the code above, Dart infers that `simpleList` has a type `List<int>` (a `List` with elements of type `int`).

Using a Constructor

You can also declare a list using a *List constructor*. A **List constructor** creates an object using the `List` keyword followed by parenthesis (`()`).

The general syntax is as follows:

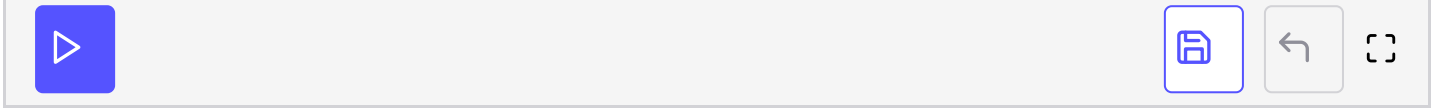
```
var listName = List()
```

When we create a list using the syntax above, we end up with an empty list. Let's look at an example below.

```
main() {
```

```
  var listOfVegetables = List();


  print(listOfVegetables);
}
```



When we print `listOfVegetables` in the code snippet above, we get an empty list (square brackets with no elements).

Specifying the Type

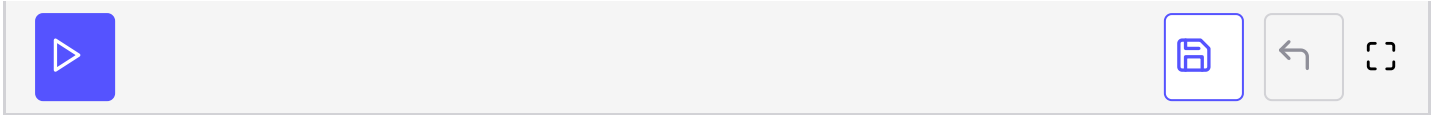
Instead of depending on Dart's type inference, we can specify the type that a list should contain.

```
var listName = List<dataType>()
```

Let's make sure our `listOfVegetables` can only store strings.

```
main() {
  var listOfVegetables = List<String>();

  print(listOfVegetables is List<String>);
}
```



On **line 4** of the code snippet above, we are checking if the type of `listOfVegetables` is `List<String>` using the `is` operator. When you press RUN, `true` should be displayed as the output.

Now that we know how to create lists, let's look at them in a bit more detail in the next lesson.