Dart: Iterable Collections

Written by Thapanapong Rukkanchanunt

Outline

- Collections vs Iterable
- Read from collections
- Checking conditions
- Filtering
- Mapping

Collections vs Iterable

- Collection is a group of elements
  - List, Set, Map
  - Access element by `collection[index]`
- Iterable is collection but access is sequential
  - Iterable<int> iter = [1,2,3];
  - Access element by `iterable.elementAt(index)`

Read from Collections

- For-in Loop can be used on collections
- `.first` and `.last` access first and last element of iterables
  - `.last` can be slow because it goes through all elements
- `firstWhere()` will find the first element in a collection but you need to provide predicate
- Predicates are functions that return true or false.
- `singleWhere()` is the like `firstWhere()` except only one element must satisfy the predicate

3 Ways to write a predicate

- As an expression using arrow syntax (=>)
  - `collection.firstWhere((item) => item.length > 5);`
- As a block between brackets with return statement
  - `collection.firstWhere((item) { return item.length > 5});`
  - Notice that you don't have to provide function name or return type
- As a function: pass function name
  - `collection.firstWhere(function_name);`

StateError

- When using `firstWhere()` and `singleWhere()`, if the program cannot find element (either it's not there or more than one satisfies predicate in `singleWhere()`), the method will throw `StateError`
- Use orElse: to catch the error
  - `collection.firstWhere(function_name, orElse: () => null);`

Checking conditions

- `.any()` returns true if at least one element satisfies the predicate.
  - `if (collection.any(predicate)) { print('any'); }`
- `.every()` returns true if all elements satisfy the predicate.
  - `if (collection.every(predicate)) { print('all'); }`

Filtering

- `.where()` returns all elements that satisfies the predicate.
  - `var evens = numbers.where((number) => number.isEven);`
  - If nothing is found, the method return empty Iterable.
- `.takeWhile()` returns all elements before the one that satisfies the predicate.
  - `var cols = numbers.takeWhile((number) => number.isEven);`
- `.skipWhile()` returns all elements after and including the first one that doesn’t satisfy the predicate.
  - `var cols = numbers.skipWhile((number) => number.isEven);`

Extra: Number property

```
+int x = 5
+print(x.isFinite);
+print(x.isInfinite);
+print(x.isNan);
+print(x.isNegative);
```

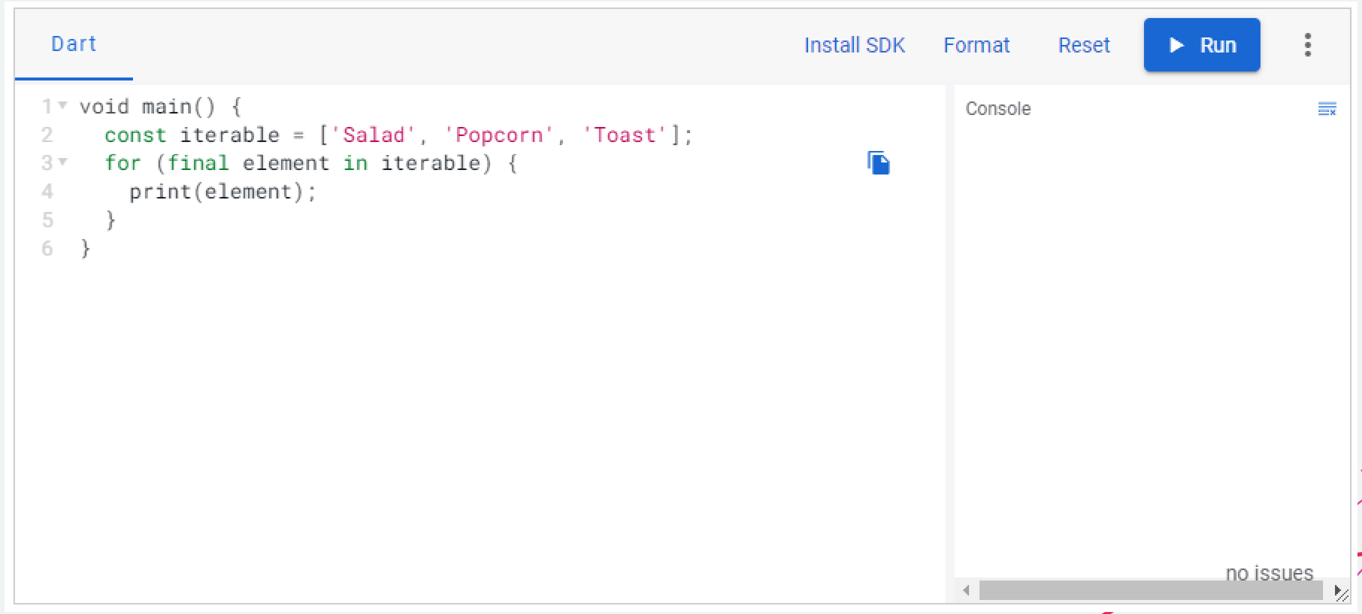
```
+print(x.sign);
+print(x.isEven);
+print(x.isOdd);
```

Mapping

- `.map()` applies a function over each element and return new one
  - `Iterable<String> output = numbers.map((number) => number.toString());`
- The return value of `.map()` is Iterable.

Lab #02 Iterable

- Complete Iterable Collections codelab
  - https://dart.dev/codelabs/iterables
- Inform staff after you complete all exercises
  - Examples can help if you get stuck on any exercises



```dart
void main() {
  const iterable = ['Salad', 'Popcorn', 'Toast'];
  for (final element in iterable) {
    print(element);
  }
}
```