General Assembly

STUDY GUIDE

SETS

What Is a Set?

Many data structures we've covered are lists, or collections of ordered values (e.g., arrays).

**Sets* are like similar to arrays, with one important difference: Sets hold unique sequences of values (i.e., a value cannot occur twice in a set).

Here's how sets compare to some of the data structures we've already met:

List type	Description
Array	**Any ordered sequence of values.**
Linked list	**A list of values, stored in nodes, referencing other nodes.**
Stack	**A list of ordered values in which the first item in is the last item out.**
Queue	**A list of ordered values in which the first item in is the first item out.**
Set	**Any unordered sequence of unique values.**

Why Sets?

Sets have a number of useful properties.
- Sets are great for storing unique collections of items, for example:
- Unique usernames in a database.
- A unique list of tags on all of your blog posts.
- Customer names.
- Sets can store a wide range of data types (e.g., numbers, Booleans, strings)
- A set won't allow you to add a value that's already present.
- Passing an array through a set object will automatically remove any duplicates in the array.

Common Set Methods

The following methods are commonly built into sets:

Method	Description
.length	**Returns the length of the set.**
.insert(value)	**Adds a value to the set, unless it's already present.**
.remove(value)	**Removes and returns a value from the set.**
.has(value)	**Returns true if the value is already in the set.**
.union(set)	**Returns a new set comprising the union between two previous sets (i.e., the combined values of both sets).**
.intersect(set)	**Returns a new set comprising the intersection between two previous sets (i.e., the values in both sets).**
.difference(set)	**Returns a new set comprising the difference between two sets (i.e., the values that only appeared in one of the two previous sets).**