Arrays

An **array** is a low-level data structure that holds an **ordered collection of elements**. Each position in the array has an **index**, starting with 0.

Confusingly, in some languages, there is a *high-level* data structure called an "array" which has a few additional features.

In a low level array, you must specify the size of your array when you instantiate it:

Java

```
// Low level arrays in Java

// instantiate an array to hold 10 integers
int gasPrices[] = new int[10];

gasPrices[0] = 346;
gasPrices[1] = 360;
gasPrices[2] = 354;
```

Arrays are efficient for looking up the element at an index, because if you know the address where an array starts in memory, it's simple math to find the address of any index. This gives arrays an $O(1)$ lookup time.

Low level arrays are the foundation of many other data structures, like dynamic arrays, stacks, and hash maps.

A **dynamic array (/concept/dynamic-array-amortized-analysis)** doesn't require you to specify the length and allows you to seamlessly (although sometimes with time and space costs) insert and delete elements at any index.

In Java, you can simply say:

```Java
List<Integer> gasPrices = new ArrayList<Integer>();


gasPrices.add(346);
gasPrices.add(360);
gasPrices.add(354);
```

Here, the details about the array's length are abstracted out for you. You can add as many prices as you'd like.

Fun fact: **strings** are almost always implemented as arrays of characters.

See also:

- Dynamic Arrays (/concept/dynamic-array-amortized-analysis)
- Linked Lists (/concept/linked-list)
- Queues (/concept/queue)