Arrays - Simple and multidimensional

There are two kinds of array, simple and multidimensional. A simple array is usually just called *an array* and a multidimensional array is usually called *an array of arrays*. Both those names serve as an indication of what they should store and what they represent.
These structures have two important features for easy access: *length* and *index*.
I would imagine that length is the most intuitive feature of the two, it defines amount of items and so it's a simple counting exercise. Length and index are often used in the same context, since index defines a specific position of an array, meaning you could get one single element the length defines the last value you could get.

> An important note to make at this moment is that an arrays' index begins at 0 and the length is absolute value. When getting the last item of an array bear in mind that note.

Let's see some length examples below:

```
var example1 = [1, 2, 3];
var example2 = [1, 2, 3, 4, 5];
var example3 = [1, 2, 3, [1, 2, 3, 4, 5]];
var example4 = [[1, 2], 3, [1, 2, 3, 4, 5]];
```

Both the `example1` and `example2` arrays are unremarkable, they have a length of 3 and 5 respectively and are quite the "basic example" archetype required to make a point, by contrast, `example3` and `example4` might surprise you. The `example3` has a length of 4 and `example4` has a length of 3, if this surprised you I should call your attention to every time I mentioned arrays above and wrote something like a "collection of items" - using the word "item" for the definition is very important and not one bit innocent. The example above teaches us two things about that "collection of items" expression:
1. Items is a generic name for "anything that's in a given array index"
2. Arrays with a greater amount of items can have a smaller length

Let's go back to `example3`, I claimed that it had a length of 4 and that indices started at 0. How can I get the last item of `example3` and what would it be. If the first index is 0 and the length is 4 the last index is 3:

```
//Getting the last index of example3
example3[3] -> [1, 2, 3, 4, 5]
```

I'm guessing this result isn't that surprising. The array stored inside the last index of `example3` is an item of the variable, even though it's an array by itself, one automatic conclusion would be that `example3` is a *2 level deep multidimensional array*. I guess this introduces the concept of *level*.
See examples below:

```
// One level deep -> Simple array
var oneLevel = [1, 2, 3, 4];
// Two or more levels deep -> Multidimensional array
var twoLevels = [1,  [2, 3, 4]];
var threeLevels = [1, [2,  [3, 4]]];
var fourLevels = [1, [2,  [3, [4]]]];
```

The examples above make it easy to see how many levels an array have, it might not be as easy but the way to access any given item is through indices. The index is an essential part of using arrays. Separating an array into items makes it easier to understand how much navigation you have to do inside the structure.
For each level of a given array you have to add an index.
Let's take the examples above and access number 4 as it's most deeply nested in all the examples:

```
// oneLevel:
//    length: 4
//    level: 1
oneLevel[3] -> 4
// twoLevels:
//    length: 2
//    level: 2
twoLevels[1] -> [2, 3, 4] -> twoLevels[1][2] -> 4
// threeLevels:
//    length: 2
//    level: 3
threeLevels[1] -> [2, [3, 4]] -> threeLevels[1][1] -> [3, 4] ->
threeLevels[1][1][1] -> 4
// threeLevels:
//    length: 2
//    level: 4
fourLevels[1] -> [2,  [3, [4]]] -> fourLevels[1][1] -> [3, [4]] ->
fourLevels[1][1][1] -> [4] -> fourLevels[1][1][1][0] -> 4
```

If you look closely at each example you'll see I'm always checking for the items *length* to get the *index* and each item has a subarray, so they had their own indices.