Rev. 08Aug2019

Exercise 31

A LITTLE ABOUT ARRAYS

JavaScript objects are a great choice when you have different bits of information related to a single subject:

```
let states = {
  ak: 'Alaska',
  al: 'Alabama',
  az: 'Arizona'
}
```

How would you access a state like Connecticut?

states.northeast.ct

And, we've even seen that we can have objects *within* objects:

```
let statesByRegion = {
  northeast: {
    ct: 'Connecticut',
    de: 'Delaware',
    me: 'Maine'
  }
}
```

This is known as "drilling down" into an object

That works well when we know upfront what they keys to an object will be. Consider, though, a different situation. Students enroll in a course. You might think: let's create a course object. Good. Maybe like this:

```
let course = {
  title: 'Screenwriting for Beginners',
  teacher: 'Charlie Kaufman',
  students: ???
}
```

What do we do about students? They come; some stay, some drop out. It's very dynamic. How would you have keys for that? Or consider student grades for a class: what key should we use for that? For these — and many more — situations, we need something purpose-made for *groups* of items. And that thing is an *array*.

Let's take the case where we have a week of information about high temperatures. Here they are: 84, 89, 78, 79, 91, 95, 90. And what sort of things would we want to know, given that data. Here are some things we might want to ask the data:

- What was the highest temperature?
- What was the lowest?
- What was the average temperature?
- How many days was the temperature 90 degrees or higher?

Arrays have the unique ability of making questions like that easy to answer. We'll get to that shortly in another exercise. For now, let's just get some familiarity with creating arrays.

```
// create an array of temperatures
let temps = [84, 89, 78, 79, 91, 95, 90]
```

Let's look at another example:

```
let diamondShapes = ['round', 'princess', 'emerald', 'asscher', 'marquise', 'oval',
'radiant', 'pear', 'heart', 'cushion']
```

The pattern appears to be an opening square bracket: `[` — followed by a comma-delimited list of items, which is then terminated with a closing square bracket: `]`

- [ ] Try it yourself. In `index.js`, create an array of colors.

When we want to get information from a JavaScript *object*, we use the name of the object and the name of they key, separated by a dot: `states.al`.

When we want to get information from a JavaScript *array*, we use the name of the array and then the item number in the array wrapped in square brackets: `temps[0]`. Quite a few exercises ago, I mentioned that we typically start numbering from a starting point of 0. That always throws people off: why not start with 1? Over time, I believe you'll understand the rationale behind that decision. For now, an easy way to think about how numbering works is to think of the number as an *offset* from the first item.

Thinking in that way, what is the offset of the first item in an array? Well, it has no offset: it's the first, so the number is 0. What about the second item in an array? It's offset from the first item by 1, so it's number is 1.

Looking at the `temps` array again...

```
let temps = [84, 89, 78, 79, 91, 95, 90]
```

...what is the order number for `91`? That's the same as asking, "What's the offset for `91`?" It's offset by 4 from the first item (84), so we can access it as `temps[4]`. That's just a different way of thinking of numbering from 0.

We're going to go a little deeper into arrays in the next exercise. For now, complete the code in `index.js`.