Rev. 08Aug2019

Exercise 50

USING LODASH

As great as JavaScript is, there are times we'd like it to have even more built-in capabilities. Sounds like time for some open-source, third-party software! Unzip the accompanying file into your **exercises** folder.

lodash is a library of JavaScript functions that fills in some of the gaps in native JavaScript's capabilities. A `<script>` tag loading lodash is already included in `index.html`.There's no code to write in this exercise, but there are some great things to learn.

Open `index.js`. The first bit of code creates an array, `attendees`, with names of people who will be attending some event. The next array, `contacted`, is a subset of `attendees` — people who have already been contacted. Our goal is to output `attendees` in the element with an `id` of `attendees`, output those contacted in an element with an `id` of `contacted`, and those who have *not* been contacted in an element with an `id` of `to-contact`.

The problem is this: we have an array for all attendees, another for those already contacted — but no array for those we need to contact. lodash to the rescue!

```
const toContact = _.difference(attendees, contacted)
```

lodash functions begin with an underscore. The `difference` function is passed two arrays. It returns another array of those elements that are *not* in both — in other words, those still needing to be contacted. Now we can loop over all three arrays, displaying the appropriate names for each group.

This is commonly known as "de-duping"

Next, we have different people with their three favorite flowers. Several people like the same flower. We want an array of all favorite flowers — with any duplicates removed.

To accomplish this, we first join all of the individual arrays by using the built-in JavaScript `concat` function.

Then, we use another lodash function, `uniq`, and pass it the array of all flowers (including duplicates). The `uniq` function returns an array with only unique elements in it — in other words, with no duplicates.

Next we have an array of objects. Each object has the name of a child and their age. Following the `kids` array, we have two constants, one for a minimum age and the other for a maximum age. Our task is to output "Welcome" or "Sorry", followed by the child's name. For this we use lodash's `inRange` function.

A "string" is the programming term for a text value. "Hello, world" is a string.

There are times when we get a *string* that we'd like to display on a web page, but we need it formatted with the first letter upper-cased, followed by the rest of the string in lower-case. The next snippet of code shows how we can accomplish this with lodash.

```
const string1 = 'FeLiCiA'
const string2 = 'FELICIA'
const string3 = 'felicia'
console.log(_.capitalize(string1))
console.log(_.capitalize(string2))
console.log(_.capitalize(string3))
```

Regular expressions (regexes) are built into JavaScript and allow us to do very sophisticated pattern-matching.

Let's say we have an array of product SKUs. We only want the ones that begin with 'IBM'. Normally, you'd use a regular expression for this, but an easier way is to use lodash's `startsWith` function.

```
const products = [
  'MBT8975',
  'RDS2614',
  'IBM6472',
  'IBM2398',
  'DEV9121',
  'IBM1887',
  'TRS6311'
]
```

Now, to `console.log` only the ones that start with `'IBM'`:

```
products.forEach( sku => {
  if( _.startsWith(sku, 'IBM')) {
    console.log(sku);
  }
})
```

These are just a few of the functions made available by lodash. For more information, check out: **https://lodash.com/docs/4.17.15**.