Note: The HTML code are donated by a previous student and we (the instructors) adopted the code as the template for this assignment.

All of your function definitions for this assignment should be put into one JavaScript file, call script.js. To test this file, you can import it on any of the three pages, and try activating the functions from the console.

**Index Page**

This is the page where you can see a list of index cards. To test these functions, try them on that page.

1. Create a function called cleanUpIndex that **removes** all of the DOM nodes that are unique to the Index page. This function takes no parameters.
2. Create a function called createSingleIndex that **creates** a DOM node that represents a single index card for the Index page. This function will take in a single object, where that object represents a single contact (see examples below). It should output a DOM node. It does not need to attach the DOM node to the rest of the DOM.
3. Create a function called renderIndex that **creates** all of the DOM nodes that are unique to the Index page. This function will take in a single parameter, which is an array containing many contacts, each of which represents a single contact. An example array of contacts is provided below. But be warned, I will supply the array, and I may change the details, like how many contacts there are. Obviously the reason you wrote createSingleIndex is because I thought it would help you write this function. Unlike createSingleIndex, which just sort of creates some stuff in isolation, this function must actually put DOM nodes onto the web page.

/\* Here is an example of a contact list array, with two contacts already populated \*/  
let contactList = [  
  {  
    name: "Roberta Dobbs",  
    phone: "778-555-1234",  
    address: "101 Main St, Anytown, USA",  
    email: "subgenius@slack.example.com",  
  },   
  {  
    name: "Bugs Bunny",  
    phone: "123-867-5309",  
    address: "Warner Brothers Animation Lot",  
    email: "whatsup@doc.example.com",  
  },  
]

**How you can tell it's working**

If you go to an index page, and you open the browser console, you should be able to:

* call cleanUpIndex() to delete all the index cards
* call renderIndex(contactList) to put them all back
  + the page should be exactly the same as it was before!
* rinse and repeat

**View Page**

This is the page where we view a single contact that we created previously.

1. Create a function called cleanUpView that **removes** all of the DOM nodes that are unique to the View page. This function takes no parameters.
2. Create a function called renderView that **creates** all of the DOM nodes that are unique to the View page. This function will take in a single parameter, which is an object that represents a single contact. Much like renderIndex, this function is responsible for actually modifying the web page.

**How you can tell it's working**

If you go to an index page, and you open the browser console, you should be able to:

* call cleanUpView() to the part of the page that shows the individual contact
* call renderView(contactList[0]) to put it back
  + the page should be exactly the same as it was before!
* rinse and repeat

**Create Page**

I hope the pattern is pretty obvious at this point.

1. Create a function called cleanUpCreate that **removes** all of the DOM nodes that are unique to the Create page. This function takes no parameters.
2. Create a function called renderCreate that **creates** all of the DOM nodes that are unique to the Create page. This function will take in a single parameter, which is an object that represents a single contact. Much like renderIndex, this function is reponsible for actually modifying the web page.

**How you can tell it's working**

If you go to an index page, and you open the browser console, you should be able to:

* call cleanUpCreate() to delete the form and all the inputs and so on
* call renderCreate() to put it all back
  + the page should be exactly the same as it was before!
* rinse and repeat

**Grading**

So you should have written 7 functions for this. Each function will be graded on a 2-point scale, for a total of 14 points for functionality.

* 2 points for flawless
* 1 point for mostly working but not flawless
* 0 points for not really working

There will be an additional 3 points for code clarity, style, layout, variable names, etc.

* 3 points: it is presentable to a professional standard
* 2 points: minor lapses
* 1 points: this looks like you did not make an effort

There will be 2 points for good git commit history, including branching at least once, merging at least once, good commit messages (no more "first commit" nonsense), not all commited at once.

* 2 points: multi-commits present, merge present, no stupid commit messages
* 1 point: tolerable but weak

That's a total of 19 points, because nice round numbers are overrated.

**Submission**

Push to GitHub, and put a link in the D2L dropbox. I will expect to see the file script.js right in the top-level directory of the main branch of your repo (can we all use main, just so I can write scripts to get the marks back faster, please).

The following may be useful:

Remove an element from the document:

const element = document.getElementById("demo");  
element.remove();

Remove the first element from a list:

const list = document.getElementById("myList");  
list.removeChild(list.firstElementChild);

Append an item to a list:

const node = document.createElement("li");  
const textnode = document.createTextNode("Water");  
node.appendChild(textnode);  
document.getElementById("myList").appendChild(node);