Final Assessment: Part one

Working with the DOM

In this portion we'll exercise working with the DOM as we pull values from a set of form fields, convert those to HTML and append them to the existing DOM.

1. Edit a new file called index.html. Put in a <form> with <input> elements for:
   - First
   - Last
   - Email
   - Cell
   - Photo URL

2. Add a button to the form with an id of "addPersonButton" and add a <section> tag at the end that will eventually be used to hold your output.

3. Create index.js and include it in index.html in a script tag. Create a function in index.js called addPerson().

4. In index.js, add an event listener to the button. When the user clicks it, you should call addPerson(). (Hints: Use getElementById() or querySelector() to get a reference to the button and then use addEventListener() to wire up the click event).

5. Inside addPerson, create variables for first, last, email, cell, and photo url. Populate the variables with the values typed into the form's text boxes. Use the debugger or console.log() them to make sure you've retrieved their values.

6. Create a new person object out of those variables. You'll see a file in starters called persons.json. This can serve as a guide for the shape of this person object.

7. In addPerson(), get a reference to the output <section> of your page and add a "person card" inside of it. You can use the person-card.html file in the starters folder if you like. If not, maybe something like this ...

```
<div class="personCard">
  <img src="${person.picture.large}" alt="${person.name.first}" />
  <p>${person.name.first} ${person.name.last}</p>
  <p>Email: ${person.email}</p>
  <p>Cell: ${person.cell}</p>
</div>
```

8. Test your code. If you do this right, your person will appear on your page when the form is filled out and the button is clicked.

9. When the user clicks "Add person", also clear out the form fields so they can enter another. (Hint: set the value property of each form field or call the reset() method for the form).

10. Let's do a little refactoring. Change your button's click method to add your new person to an array of person objects. Then change your code to loop through that persons array and append your person card to the output <section>. This way, each time your user presses the button, it will add multiple persons to the page.

Bonus! If you have extra time, alter your code to put your new person at the top of the list instead of the bottom. (Hint: Array.unshift()).

Final Assessment: Part two

Ajax

At this point you have a page that displays a list of persons. But this list is being typed in by hand from the form. Let's read those persons from an API by making an Ajax call.

Examining the API

11. First, open a browser and navigate to https://randomuser.me/api/?results=5
12. Take a look at what is being returned. It may be helpful to open the browser's developer tools and study the network tab. If you notice, the value coming back is pure JSON. Remember that URL because we'll come back and use it in just a minute.

Crafting a call to the API

13. Open your index.js file and create a new method called getPersons().
14. Call getPersons() on an event. You can either:
    a. Add a "Get Persons" button and call getPersons() on button click or
    b. Call getPersons() on the DOMContentLoaded event.
15. Run and test, making sure that getPersons() is indeed being called.
16. In getPersons(), do a fetch from the URL we visited above.

Processing the API response

17. Remember from the lecture that fetch returns a promise. So, after your fetch call, do this:

```
.then(res => {
      const responseObject = res.json();
      console.log(responseObject);
      return responseObject;
      })
```

That will convert the request string to a true JSON object.

18. Run and test again.
19. Take a look in the developer tools again and examine the data being returned. You should have an object that has our list of people inside it. What is the name of that key? ____
20. Chain another .then() function that will take the responseObject you saw in the step above and return only the property you discovered above. (Hint: you'll add one line of code and it will be another .then())
21. Run and test. If you've done it right, every time you refresh, you'll see a new list of random people in the developer tools.

Displaying the data

22. Now that you have a list of persons, display those on your index.html page. (Hint: Re-use the portion of your code that displays a list of persons on the page).

Refactoring

23. Change your getPersons method to receive in a "numberOfPeople" parameter with a default of 5. Use it in getPersons to fetch that number of people instead of the hardcoded number you have in there already.
24. Go through and pretty up your code, formatting it, tightening it, using good shortcuts wherever possible (like destructuring, spreading, object property shorthands, etc).