The goal of this assignment is to get some practice with the XMLHttpRequest object and use it to make HTTP requests to the server for additional resources.

You will be using various shared resources that can be found at this URL:

```
http://students.engr.scu.edu/~adiaztos/resources/<resource_name>.php
```

Anytime you are asked to retrieve a resource, use the URL with the name of the resource at the end as shown.

To get started, download the starter files under Lab 7 in the files. Copy these files to your webpages directory

Screenshots of what each part should look like can be found under the Examples folder in the Lab 7 files.

For part's 1 and 2, use the native DOM and XMLHttpRequest JavaScript APIs, **NOT** any library methods (i.e. jQuery).

Part 1 - XMLHttpRequest

Here you will be retrieving 3 different resources from the server and adding their content to the HTML document.

This takes three steps:

1. Create an XMLHttpRequest object
2. Set an event handler function for the *onreadystatechange* event
3. Open the XMLHttpRequest connection giving it the GET method and the full URL to the resource
4. Sending the request using the send method

The resources are: sample1.php, sample2.php, and sample3.php

For sample1 and sample2, simply set whatever the response text is as the innerHTML of the respective divs with id sample1 and sample2.

For sample3 however, the response is a JSON object that looks like this:

```
{
 "friends": [
 {
 "id": 0,
 "name": "Jonny Todd"
 },
 {
 "id": 1,
 "name": "Flora Joyce"
 },
 {
 "id": 2,
 "name": "Owens Ford"
 },
 {
 "id": 3,
 "name": "Phoebe Downs"
 },
 {
 "id": 4,
 "name": "Lois Odom"
 }
 ]
}
```

Because this response is in JSON, which is plain text, we need to convert it to a JavaScript object like so.

```
var response = JSON.parse(this.responseText);
```

The goal is to create a list of friends from this object. This means you need to create an **unordered list** element, *ul*, and append it to your page. You will need to get the friends property, who's value is an array. Then iterate over each friend object, creating a **list item**, *li*, for each friend and setting the inner text of that element to the friend's name. Don't forget to append the list to the div with id *sample3*.

Part 2

In this part, you are given a simple contacts UI in part2.html, your job is to write the script, part2.js, to 1, populate the contacts, and 2, allow someone to search through the contacts. Both of the functionalities are going to take advantage of XMLHttpRequests.

If you notice at the top of the script, we are getting the element with id *template*. This element is only going to be used to clone new elements with the correct contact information from the request.

The first step to sending the request is to create an XMLHttpRequest object. We will only need one for this part.

Secondly, assign an event handler to the onreadystatechange event of the XMLHttpRequest object. In this event handler, you will do two things:

1. The first thing is to create an object from the response text using JSON.parse.
2. Once we have a JavaScript object, we can pass that to the function *populateContacts* which will handle processing the data.

Then, open a GET request, to the *contacts.php* resource and send the request. The response looks something like this (notice it's an array with objects inside it):

```
[
 {
 "id":"404",
 "name":"Abelard McMonnies",
 "email":"amcmonniesb4@booking.com"
 },
 {
 "id":"590",
 "name":"Abner Dellatorre",
 "email":"adellatorrega@squidoo.com"
 },
 {
 "id":"407",
 "name":"Ada Empson",
 "email":"aempsonb7@miitbeian.gov.cn"
 },
 {
 "id":"335",
 "name":"Adara Glazyer",
 "email":"aglazyer97@addtoany.com"
 },
...
]
```

Similarly to sample3 in part 1, you will traverse an object and create a new node for each contact. Taking advantage of the template element we can create new nodes like this:

```
var node = template.cloneNode(true);
```

Because the new node is a clone of the template, we need to make sure to update some of its properties, most importantly the id. Luckily, each contact object provides a unique *id* for each contact, so you can set the new node's id to the *id* of the contact.

Next, we need to update the number for the contact. Select the span with id *index* inside the template clone and give it an index number. The should range from 1, 2, 3, etc..

Because index also has an id, we need to give it a unique id. You can get the current *id*, *index*, and concatenate it with the contact's id to ensure it is unique.

Next, we need to set the name and email. If you look at the template, the name and email fields are just input fields. The name and email for the contact will be assigned as the *value* for that input, (don't worry about someone changing the input field, because the inputs have the attribute 'readonly', they can't be edited)

For selecting elements, querySelectorAll will probably be the easiest method, but remember, it returns a collection, not a single element.

Finally, attach the cloned node to the template's parent (already selected at the start of the script).

You should be able to test your contacts list at this point.

To implement the search functionality, finish the search function. You will need to get the value inside the text field since that will be the string we are searching for in the list. Rather than doing the searching yourself, you can send the search string as a query parameter to the contacts.php resource. To do that, make sure when you open the request you use the POST method, and that your URL will look something like this:

```
http://students.engr.scu.edu/~adiaztos/resources/contacts.php?query=Al
```

The string after *query=* is the value from the search field which we just concatenated to the rest of the URL.

Note: If you use the same XMLHttpRequest object from before to send your search request, you won't need to do anything else. The same event handler will be invoked, and the now filtered contacts list will be populated using your populateContacts function.

Don't forget to attach the search function as the click handler for the **search button**.

You should now have a fully working contacts list, with search functionality.

Part 3

For parts 3 and 4 you will use jQuery.

This part is similar to part 1 of Lab 5. Using jQuery, do the following:

1. Change all the text to uppercase before adding it to the paragraph with id "allUpper"
2. Change all the text to lowercase before adding it to the paragraph with id "allLower"
3. Change the color of the text to "red" and add it to the paragraph with id "redText"
4. Add the text to the paragraph with id "flashyText", and also add the class "flashy" to the paragraph

Part 4

This part is similar to part 1.

For sample1 and sample2 use the jQuery *load* method to request the content and set it as the content of the divs with id sample1 and sample2 respectively. With the *load* method, this is possible all in one call.

For sample3, since we need a little bit more manipulation, use the *$.get* method to get sample3.php and create the friends list.

Remember the jQuery syntax for creating new elements looks like this:

```
var list = $("<ul></ul>");
```