# First App Project - Part 2

Assignment Goals:

- Use JavaScript coding best practices - variable definition, IIFE for scope, etc.
- Write a JavaScript application object with data and methods.
- Handle getting the HTTPObject in a cross browser way
- Make an Ajax request for data, check the status, and do callback on success
- Show a loading message until data is populated
- Show an error message if the data could not be loaded
- Ensure the area updated with the JSON data uses ARIA attributes for accessibility
- Save JSON data to browser storage
- Load JSON data from browser storage
- Clear local data from browser storage
- Use button events to trigger changes to the page

## GitHub Repository

https://github.com/htc-ccis2591/first-app/

This assignment is a continuation of Part 1, so you do not need to fork the repository again. Continue working on the files that you added and committed for part 1. Do not make a new repository. If you want to mark the files that you had submitted for part 1, you can look up how to create a tag on GitHub.

## Move Data to File

For this assignment you will make a request to get the JSON data instead of having it in your JavaScript file. Since we don't have a remote server, we will just make a request to get it from a file. Make a data file in your project and create a new file with a .json extension. Move your data (not the variable that it was assigned to) to the new file.

## Make Buttons for Data Load

Change your HTML page to have 4 buttons: Ajax Load, Local Load, Local Save, Local Clear. Give each button an id for easy access in JavaScript. You will set up an event handler for each button as follows:

### Ajax Load

This should load the page data by making an Ajax request (for the local json file). A loading message should be shown until the data is fetched. If an error occurs (test this by using the wrong file name), then an error message should be shown.

### Local Load

This should first check to see if HTML 5 local storage is supported in the browser and if not show an error message. If it is, then attempt to load your JSON data from local browser storage, but show a message if the data does not exist. (Optionally, disable the button if the data is not found so that it cannot be clicked on, and enable it once the data is saved.)

## Local Save

This should first check to see if HTML 5 local storage is supported in the browser, and if not show an error message.  If it is, then save your JSON data to local browser storage.  If no data is on the page yet, show an error.  Optionally you can disable this button until data is loaded.

## Local Clear

This should first check to see if HTML 5 local storage is supported in the browser and if not show an error message.  If it is, then clear your data from the  local browser storage.  If no data stored, do not show an error.

Recall that local storage was discussed in Chapter 5 of the textbook.  Ajax is discussed in Chapter 8.

## Tips

- Be sure to use all of the JavaScript coding best practices that we have discussed.
- Make functions for everything.
- Start with the JSON load, as you need the data from the file to save it to local storage.
- Save the data in a global (to your IIFE) variable.
- Make sure that you write cross-browser code to do the Ajax request as illustrated in the text.
- Be sure to test the use of your buttons in different orders to ensure that they are always working or showing errors correctly, regardless of the order in which they are clicked.

### Optional Bonus Items

Instead of showing messages if HTML5 local storage is not supported by the browser, you could do either of the following:

- Hide the buttons so that they are not shown. (Do not disable them, as it is expected that you can do something on the page to enable something that is disabled.  Hiding is more appropriate if that can never happen.)
- Research how to use cookies and utilize those as a backup option.

Add a form for search or sorting to your page.  (Note that I am not stressing either of these, as this is often done on the backend, before the ajax data is returned to the page.  Generally the amount of data is too large to do this effectively in the browser, as only a subset of the data is returned.)