CS701 Module2 Assignment

General Rules for Homework Assignments

- You are strongly encouraged to add comments throughout the program. Doing so will help your facilitator to understand your programming logic and grade you more accurately.
- You must work on your assignments individually. You are not allowed to copy the answers from the others. *However*, you are encouraged to discuss approaches to the homework assignment with your facilitator.
- Each assignment has a strict deadline. However, you are still allowed to submit your assignment within 2 days after the deadline with a penalty. 15% of the credit will be deducted unless you made previous arrangements with your facilitator and professor. Assignments submitted 2 days after the deadline will not be graded.
- When the term *lastName* is referenced in an assignment, please replace it with your last name.

You are strongly encouraged to add comments into your program!

Create a new folder named HW2_*lastName.* Write the following programs in this folder.

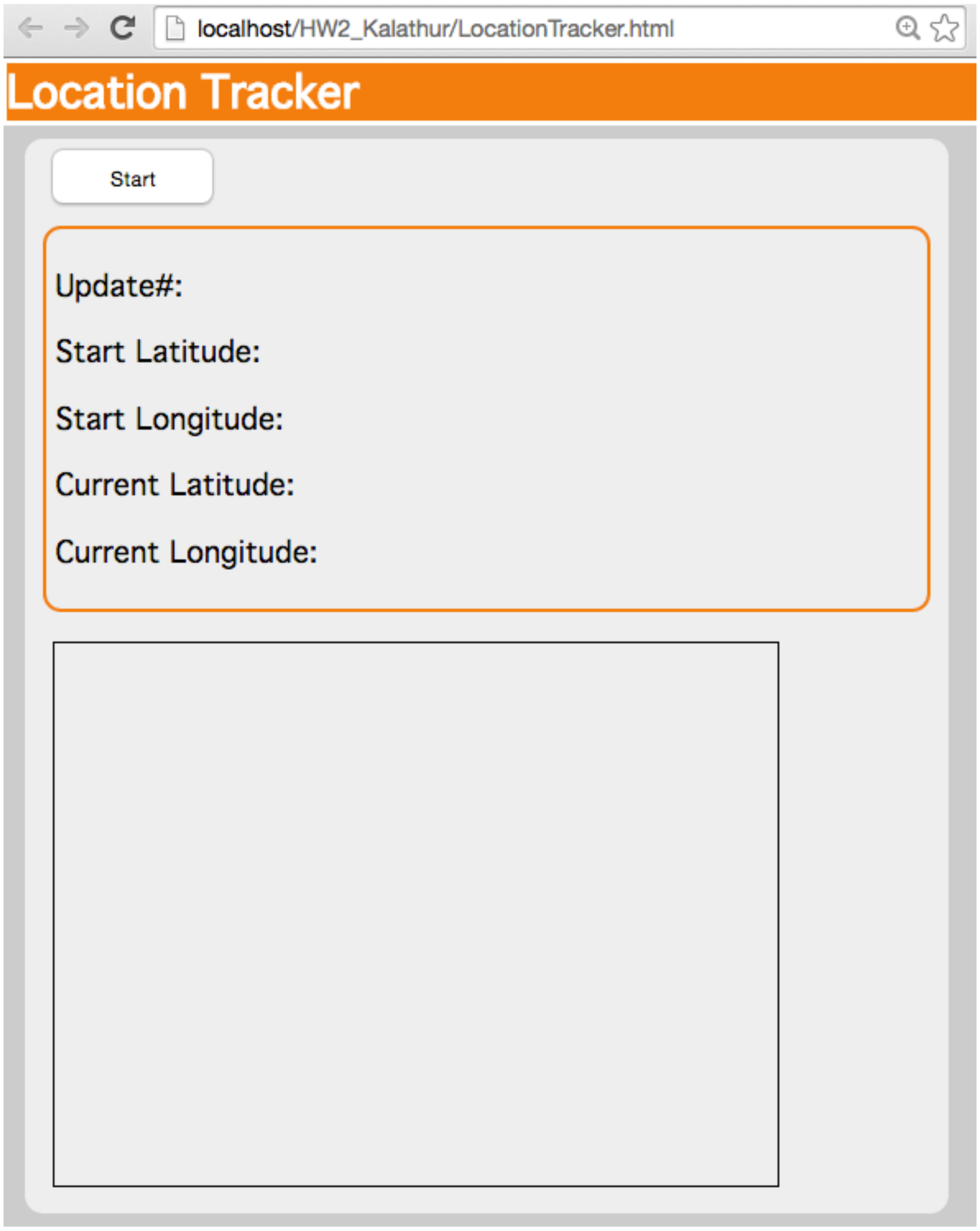
Scenario:

You decided to release the dove residing in your birdcage. But, you are keen in tracking the flight of the bird. So, you tied a sensor tag to the poor bird and decided to write a HTML5 application to monitor its path. Since you neither have the bird nor the sensor, you decided to simulate the path.

Part 1 – Geolocation (30 Points)

Using the HTML5 Geolocation API, write the LocationTracker.html and the associated Javascript file LocationTracker.js. You are free to use the html5.css from the samples.

The initial rendering of the HTML page is shown below with placeholders for the location details and the Google map.



Location Tracker

Update#:

Start Latitude:

Start Longitude:

Current Latitude:

Current Longitude:

When the *Start* button is clicked, get the current position using the Geolocaiton API and display the initial location in the map as shown below. After the initial location is displayed, use the *setInterval* method to invoke your method *updateMyLocation* every 5 seconds. The *Start* button is disabled from now on.

The *updateMyLocation* method simulates the changes in the latitude and longitude as follows. Generate two random numbers using Math.random() and divide each by 100. These two numbers represent the changes in the latitude and longitude. Modify the current location by adding the latitude value and subtracting the longitude value. This will make the bird only fly in the NorthWest direction. If you are on the Northern border or the Western border, feel free to keep the bird over the United States (or your country) by the appropriate changes to the current location. Update the current location in the HTML and also draw the path in the Google Map as shown in the following screencast:

https://mymedia.bu.edu/media/CS701_HW2_Clip1/1_y4orrxvm

Part2 – HTML5 Drag and Drop and Local Storage (40 points)

Democrat and Republican senators voting by their party lines through Drag and Drop.

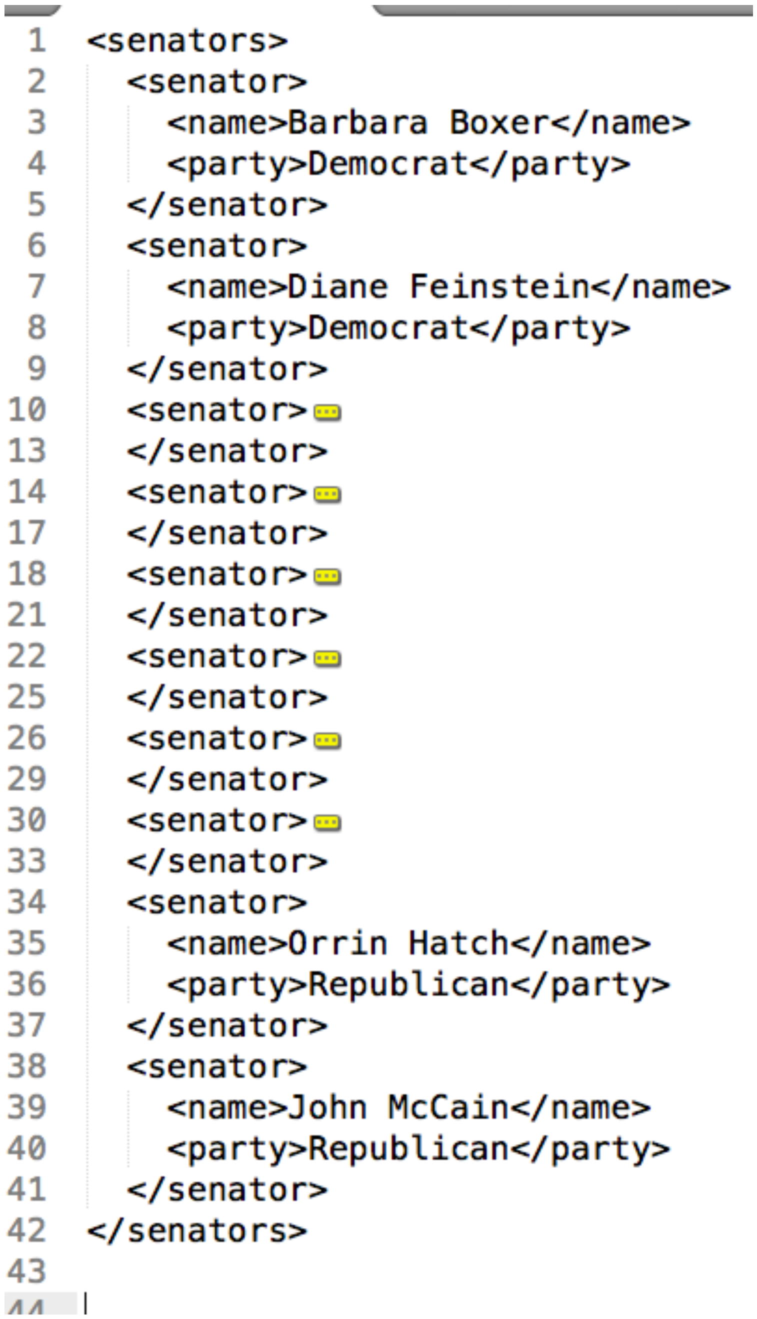
The application presents a list of senators and two areas representing Democrats and Republicans. The senators are dragged and dropped into their respective areas.

The initial list is loaded through AJAX from the partyList.xml file (provided in the samples). Each senator is converted to a JSON object keeping track of the properties *name*, *party*, and *voted* (true or false). The list of senators is then stored in local storage. Whenever a senator is dragged and dropped into their respective area, the JSON object is updated as voted and the list of senators is updated in the local storage. When the application is loaded, the local storage is first checked. If the data is there, the list of senators is loaded from the local storage, otherwise the AJAX call is made.

When the data exists in the local storage and the application is loaded, the senators already dragged into their respective areas should also be populated. Make sure you test this case.

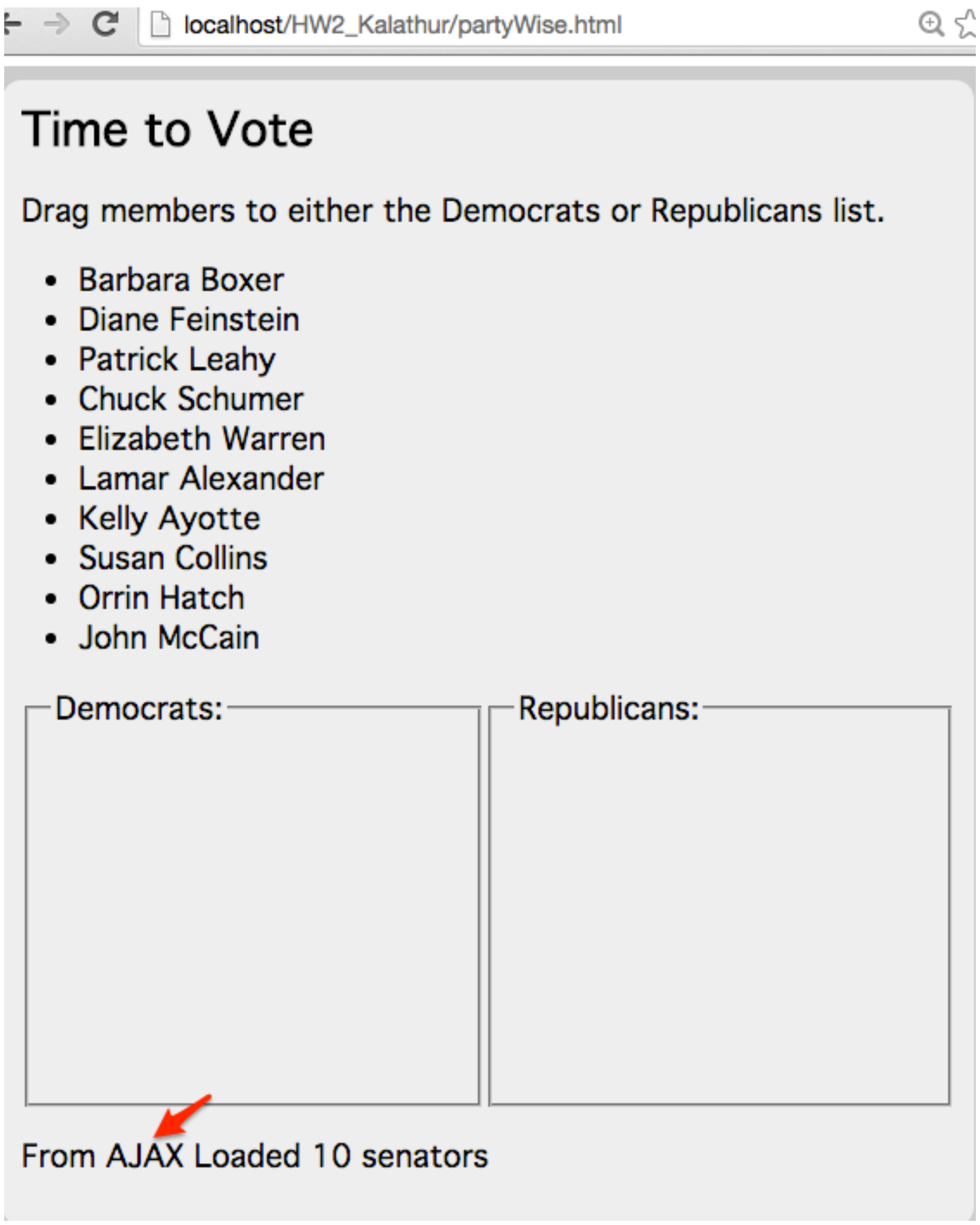
Write the partyWise.html and partyWise.js for the application's functionality.

The structure of partyList.xml is shown below.



```xml
<senators>
  <senator>
    <name>Barbara Boxer</name>
    <party>Democrat</party>
  </senator>
  <senator>
    <name>Diane Feinstein</name>
    <party>Democrat</party>
  </senator>
  <senator>
  </senator>
  <senator>
  </senator>
  <senator>
  </senator>
  <senator>
  </senator>
  <senator>
  </senator>
  <senator>
  </senator>
  <senator>
    <name>Orrin Hatch</name>
    <party>Republican</party>
  </senator>
  <senator>
    <name>John McCain</name>
    <party>Republican</party>
  </senator>
</senators>
```

The initial screen when the application is loaded is shown below. AJAX call is made to load the senator list.


localhost/HW2_Kalathur/partyWise.html

Time to Vote

Drag members to either the Democrats or Republicans list.

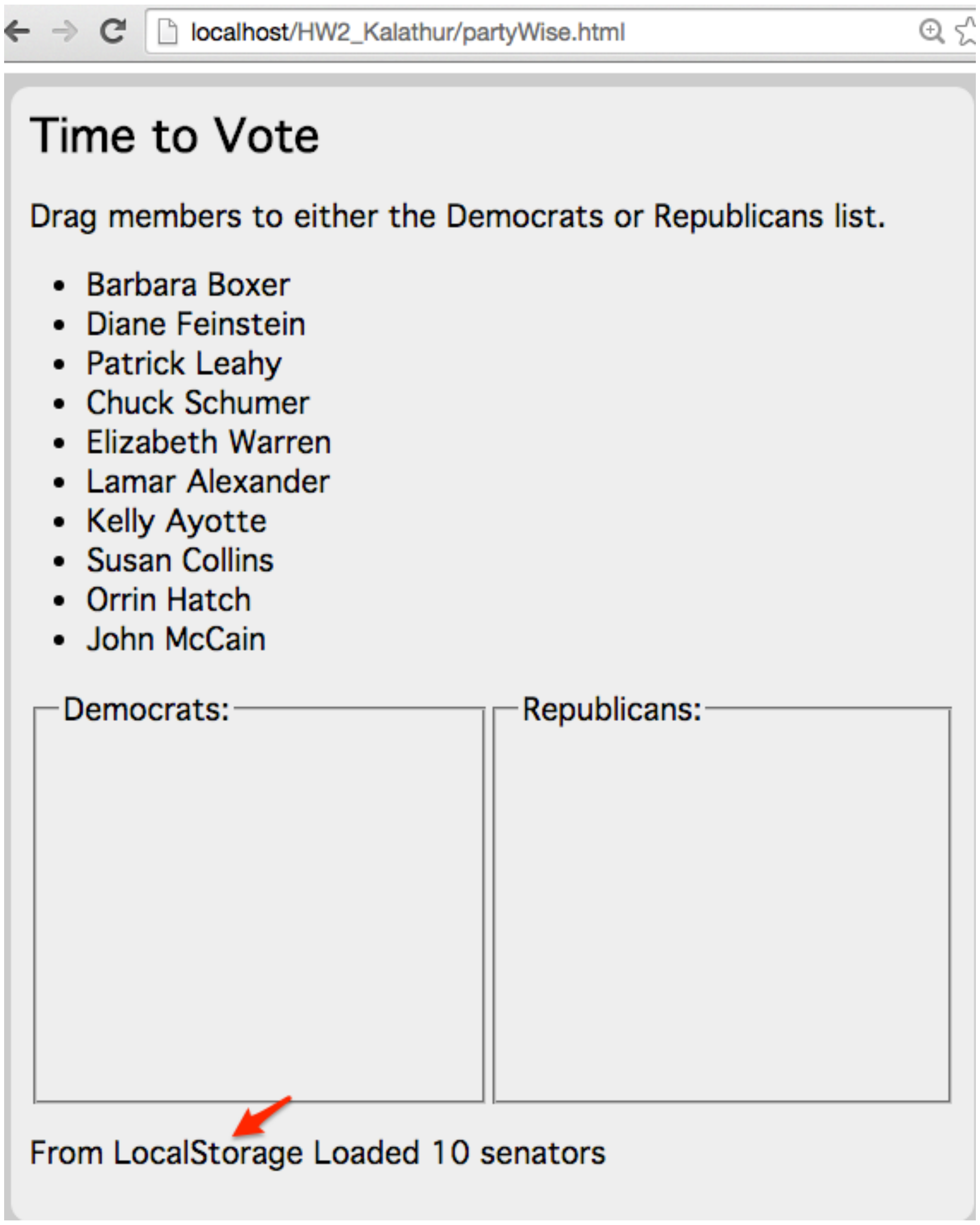
- Barbara Boxer
- Diane Feinstein
- Patrick Leahy
- Chuck Schumer
- Elizabeth Warren
- Lamar Alexander
- Kelly Ayotte
- Susan Collins
- Orrin Hatch
- John McCain

Democrats:

Republicans:

From AJAX Loaded 10 senators

If the application is refreshed, the data should be loaded from the local storage:



localhost/HW2_Kalathur/partyWise.html

Time to Vote

Drag members to either the Democrats or Republicans list.

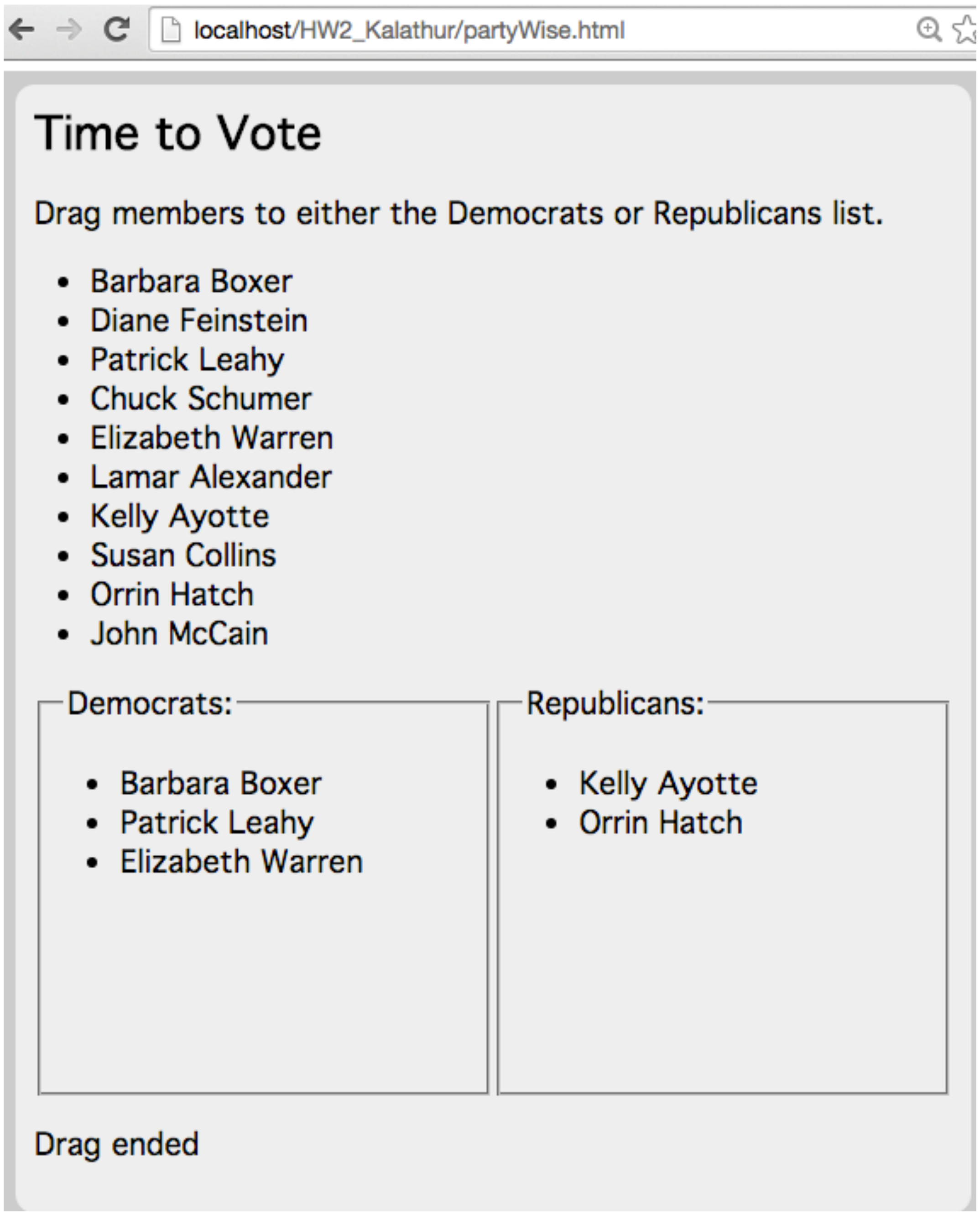
- Barbara Boxer
- Diane Feinstein
- Patrick Leahy
- Chuck Schumer
- Elizabeth Warren
- Lamar Alexander
- Kelly Ayotte
- Susan Collins
- Orrin Hatch
- John McCain

Democrats:

Republicans:

From LocalStorage Loaded 10 senators

After a few drag and drops, the application looks like this:


localhost/HW2_Kalathur/partyWise.html

Time to Vote

Drag members to either the Democrats or Republicans list.

- Barbara Boxer
- Diane Feinstein
- Patrick Leahy
- Chuck Schumer
- Elizabeth Warren
- Lamar Alexander
- Kelly Ayotte
- Susan Collins
- Orrin Hatch
- John McCain

Democrats:

- Barbara Boxer
- Patrick Leahy
- Elizabeth Warren

Republicans:

- Kelly Ayotte
- Orrin Hatch

Drag ended

Note that the democrats can only be dropped into the Democrats area. Similarly for the Republicans. You cannot drag and drop the member already voted.

If the application is quit and then reloaded, the application should resume from the last snapshot.

Part3 – Web Workers and Local Storage (30 Points)

Your web application starts 5 web workers (computeWorker.js) and sends the messages to them to compute the sum of all the integers from the specified start value to the specified end value. The two values are send as a JSON object having *start* and *end* properties. The web worker computes the sum of all the integers from the specified start value to the specified end value and sends back the result. The result is sent as a JSON object having *start*, *end*, and *result* properties. The main web application stores all the results it receives in local storage.

a) Show the JavaScript code for computeWorker.js.
b) Show the code for the main web application starting the workers and sending them the messages. The values send for the five web workers can be (1, 1000), (1001, 2000), (2001, 3000), (3001, 4000), and (4001, 5000), respectively.
c) Show how you will use local storage to record the results as they are being received. Use a single key in the local storage and store the results received into an array. Update the user interface as the results arrive from the workers.

Submission: Export your HW2_*lastName* folder as a zip file, with the appropriate index.html for the above files, and upload the zip file to the Assignment section.