Jim Nichols

Front-End Web Development I (INF651G)

December 11, 2023

Instructor: Ravindra Ramsey Muvva

GitHub Link:

YouTube presentation of User Interface: <https://www.youtube.com/watch?v=Celgj79Heuc>

Final Project Write-up: Family Chore Chart

The approach I took to develop the Family Chore Chart web application was first to create the pages that were mocked up in the planning document. I spent a significant amount of time coding the HTML and CSS for each major page that I had envisioned. Working first on the common elements, I created a template with a common header, layout and footer. Then I used that template as a starting point for each of the pages, adding to the external CSS style sheet as the unique features of each page required. I took time and care to make sure each page was responsive, with a few exceptions that were overlooked. The images used were open source and credit was given in the footer.

After the HTML and CSS were mostly complete, I turned my attention to page 1 – index.htm. The main javaScript concern here was to create a login mechanism. In order to log in, a user first needs to create an account, so they were directed to the account creation page – account.htm. I used javaScript to collect the form data from the account creation form. This included the choice of an avatar icon. It was unexpectedly difficult to use javaScript to toggle a thick border around each of the avatar choices. I ended up creating a short function for each icon with an eventListener to increase the borderWidth style. I simultaneously used a for loop to turn off the border highlight on all other icons at the same time. It’s not elegant but effective. The icon choice also generated a value that is saved into a global variable that is collected at account creation.

For account management, I created a function to collect the form. On submit, the data is passed to an Account constructor to create a userAccount object. Each account object is then pushed to an array for safe keeping and to enable for loop access in future functions. With the creation of each account, a listener also invokes a function to save the userAccounts array to localStorage for persistence among multiple pages. Finally, a confirmation box pops up after account creation with an option to be forwarded to the login page. Users have the option to cancel the location reassignment if they want to stay on the account creation page to generate more accounts, like a parent might choose to do for their children when setting up a family chore group.

When the user does finally choose to move to the login page, the saved user account data is loaded from localStorage on page load. When they login, the username and password presented are collected from the login form and checked against each user account object in the userAccount array. If no match is found, the login function populates a paragraph on screen with a message to try again. If successful, they are greeted by name on screen, and their name and avatar icon show up in the login/logoff widget on the upper right corner of every page. The Login link changes to a Logout link. This is done by using the login function to grab the elements by their IDs and replace content and toggle the display styles between “none” and “block”. A global loggedIn variable is set to the logged-in user’s username and saved to localStorage for persistence among pages.

From here, a user can navigate to the chore creation page. The loggedIn data and user account information are loaded via functions invoked on page load. This allows for the user to remain logged in from page to page. Chorecreate.htm is a simple page, but one feature I enjoyed coding is a function that reads names from the userAccounts array and uses them to create and append option elements to the select list whose ID is specified as a parameter. On the chore creation page, there are two select list widgets, exemptions and assignment, that had their options populated with this function. Similar to the process used for userAccounts, when the chore creation form is submitted, it runs a function that collects the form data and uses it to create a chore object from a Chore constructor. It then pushes the object into a choreList array. Finally, the array is saved to localStorage to remain persistent among pages.

Finally, I created the chore scheduling page. This turned out to be a lot more difficult than I bargained for. The idea of the page is to assign chores to certain days on a week-long schedule and assign each to a user. I started with the date. There is a form input of type Date midway down the page. The user is instructed to choose a Sunday as the starting day of a chore week. If they choose a non-Sunday, an alert instructs them to try again. To make this work, I had to collect the form data from the date input and use it to create a date object. A peculiar thing is that the Date() constructor assumes the input to be UTC but generates output in local time, meaning it was several hours off. I had to concatenate a modifier to the input parameter to compensate. I then used the Date object together with .getMonth(), .getDate() and .getFullYear() methods to produce a correct date and assign it to an empty heading in the Sunday schedule box. I then incremented the date and repeated for the rest of the days of the week. I should have formulated a loop to handle the repetition but did it manually instead.

For the manual chore assignment widget, I once again used functions to populate select lists. Reading the respective arrays, I did this for chores, users and days of the week. Upon submit, another function collects the form data and uses it to generate a schedule object from a Schedule constructor. It then pushes the object into an array and saves the array to localStorage. I wrote an additional function to display each item on the visual weekly schedule by getting the appropriate daily div and appending a new paragraph to the div with the name of the chore in it. It also assigns the class of the paragraph to correspond to the assigned user listed in the legend so that the color coding matches.

I was having challenges with the chore scheduler page automatically reloading localStorage scheduling data and adding duplicate items to the schedule that was already populated. To deal with this, I removed the display function from being invoked at page load and instead created a button at the top of the schedule to display the last-saved schedule. When displayed, this view hides the scheduling controls so that a user doesn’t accidentally assign duplicate chores. There is also a reset button that wipes scheduling data from local storage completely and reloads, giving the user a fresh beginning if they chose to start from scratch. It does reserve user and chore data.

Some of the pages I had planned were abandoned in the end. I hope to revisit the project at some point but the clock ran out this time. Despite my best effort and many, many hours dedicated to the project, I was unable to finish the dashboard pages for individual users and for the administrator. In addition to a progress meter, each of these dashboards were meant to provide access to detail pages regarding the chores and the users. I envisioned the user-level chore detail page as a place where workers would not only learn details about the chore assignments, but also check off completed chores. This action would inform both progress bars. The HTML and CSS draft pages can be seen in indchoresched.htm, choretemp.htm, and admin.htm, with abandoned javaScript code regarding indchoresched.htm at the end of the myScript.js file.

On a positive final note, I learned a great deal about developing with javaScript and object-oriented programming that I had never, or in some cases rarely, used before. Although briefly acquainted with OOP classes and constructors, this was the first original javaScript script that I’ve created that was centrally focused on them. I recognize that many of my functions would be better off as methods embedded within those classes. I remain curious about how to know when each is more appropriate. This is my very first outing with localStorage. It did take some hours of experimentation to get the hang of it. I found it most annoying that my loading function loops would retrieve stored items that did not match my naming pattern, and then save them as null, gumming up the works. I eventually figured out how to filter them out with a conditional. Although I enjoyed using localStorage and see it as a great advantage in maintaining persistence among pages for things like logged-in status, I can also see how it proposes a security concern. In the future, I think I’m more likely to deal with the more sensitive data from the server side.

This project gave me plenty of practice reading data from form inputs which is always handy. I used a good many for loops including for loops within for loops. On one occasion, I even used a for loop inside a for loop inside a for loop to manage users, chores and schedules. I interacted with the DOM quite a bit, changing styles and adding elements at will. Overall, I found the learning experience to be very valuable and I hope to build and expand on these skills going forward.