CSC309: Individual Project Alpha Release

Name: ContainyJS

Description and Use cases: My library makes containers and cards objects that can be placed on the screen. The cards are stored in big containers which can be expanded or collapsed based on some user action. The card's front view gives brief description about of the subject. On clicking the card flips into detailed summary and also contains links related to the subject. I got inspired for this idea through the folders functionality in mobile phones where apps are combined into folder.
I feel there are a large number of use cases for this library. Some of them would be Contact Directories, Product Libraries, flashcards on Education websites, Photo walls, Navigation Window, etc.

Features implemented:

- Add card object to the big container object and display on screen. Container is adjusted automatically to account for new card.
- Add Links to any card object in container object
- Turn on Collapse feature. If user wants, the container object is modified to allow for expand and collapse on the screen on some user action
- If collapsed, the state is preserved for the current session
- Get Requested card to search a specific card
- Get all cards
- Delete a card object from container object and update the container object's view
- Different functions for changing colours of collapsed container, expanded container, card theme for a specific card

Link: https://containyjs.herokuapp.com/example.html

- On my webpage, first you can see some demo of some of my features. In another section, I've implemented 2 use cases using my library. I have also described some potential features.
- All the visual you see on my example page is built using my library apart from some text and a form used to describe the features.
- The big containers, the cards, collapse features, basically most of it is created using my library.

Explanation of structure:

I have two Javascript objects in my library:

1. **Card:**

   In my Card object, I have the following:

   - **`this.div`** stores the card itself
   - **`this.frontView`** stores the frontview of the card(before flipping)
   - **`this.backView`** stores the backside of the card (shown after flipping)
   - **`this.aboutSection`** stores the about section which is there in backview of the card
   - similarly, **`linkSection`** stores the link section which is hidden initially.
   - **`this.themeColor`** stores the color of the individual card that is used as background theme of the card in various places.

```
class card {
    constructor(id,themeColor){
        // Initialize main card container div
        this.div = document.createElement('div')
        this.div.id = id

        // Initialise and store front and back view of card
        this.frontView = document.createElement('div')
        this.backView = document.createElement('div')
        this.aboutSection = document.createElement('div')
        this.linksSection = document.createElement('div')
        this.themeColor = themeColor
    }
}
```

2. **Container Object:**

   - **`this.mainDiv`** contains the big container object itself
   - **`this.expanded`** contains all the cards, and also additional elements like buttons, etc depending on the current state of the object. When a card is added, it is shown here and the size is also adjusted automatically.
   - **`this.collapsed`** contains the condensed view of the container where all the cards are hidden and are only visible once container is expanded. It changes the view to expanded on clicking on it.
   - **`this.themeColor`** contains the default theme of the container and the cards unless changed by the user.

- **`this.cards`** is an array of the cards present in current container. This can be used to support various other functions.
- **`this.collapse`** stores the current state of the container whether it is expanded or collapsed.
- **`this.containerColor`** stores the background theme of the container when expanded.

```
class cardsGenerator{
    constructor(selector, themeColor){
        this.mainDiv = document.getElementById(selector);
        this.expanded = document.createElement('div');
        this.collapsed = document.createElement('div');
        this.themeColor = themeColor;
        this.cards = []
        this.collapse = false
        this.containerColor = 'dimgrey'
    }
```

Note:
It was necessary to store and preserve these properties because of the way my library is implemented. It allows us to switch easily between different views and DOM manipulations on some use action.

API Functions:

- **addCard**(id, imgUrl, title, description): adds card to the container

- **addLink**(id, linkUrl, linkText, iconImg): adds links to the card with id. This link will be displayed in link section of the card.

- **turnOnCollapse**(name): when active, the container has now two views i.e., expanded and collapsed. Additionally, a button is added automatically which lets us switch between the views in the DOM.

- **changeContainerBackground**(newColor): This changes the background of the container that contains the card.

- **changeThemeColorOfCard**(id, newColor): changes the theme color of a specific card. This colour is used in multiple

views. Changing a specific card does not impact the state of other cards.

- **changeCollapsedCardColor**(newColor): If a collapsed state is set, it lets the user to switch the colour of collapse card.

- **getRequestedCard**(id): Finds the requested card in the container and returns it for the developer.

- **getAllCards**(): Gets all the card in the container for the developer.

- **removeCard**(id): Deletes the specific card from the Container and view and adjusts the DOM automatically.

Next Steps:
For the final set of features, I am thinking to allow users to change the size of the cards without impacting the structure which I feel could be a bit challenging. I am also going to put some more additional functions for the developer allowing them to make changes of information and cards directly through a function.

On other challenging set of functionalities, I am thinking to add a feature that allows developers to activate a search bar which can search and filter the cards based on names.
Another functionality would be to allow to set a view capacity on the container and allow the user to move to next screen to view more cards if the container is full. (like having 10 cards and user can only view 3 at a time, so they have to slide to see more)

Note to the TA and my peer reviewer: Looking at the time constraints as it is end of semester, I gave my best to build as many features as I can and spent around 30-40 hours on it. This is the first time I'm learning web development, and I have put a lot of effort and time to research and learn this. I enjoyed the learning process, and I hope you like my work☺ I look forward to a constructive feedback.