**Assignment 1 – Webmail Inbox of Games**

Using plain Javascript, this assignment requires you to use template literals, access/change the DOM, event handling and object/arrays methods. Your mark counts for 15% of your final grade.

Due Date:  **Friday,Oct. 6 @ 11:59 pm**

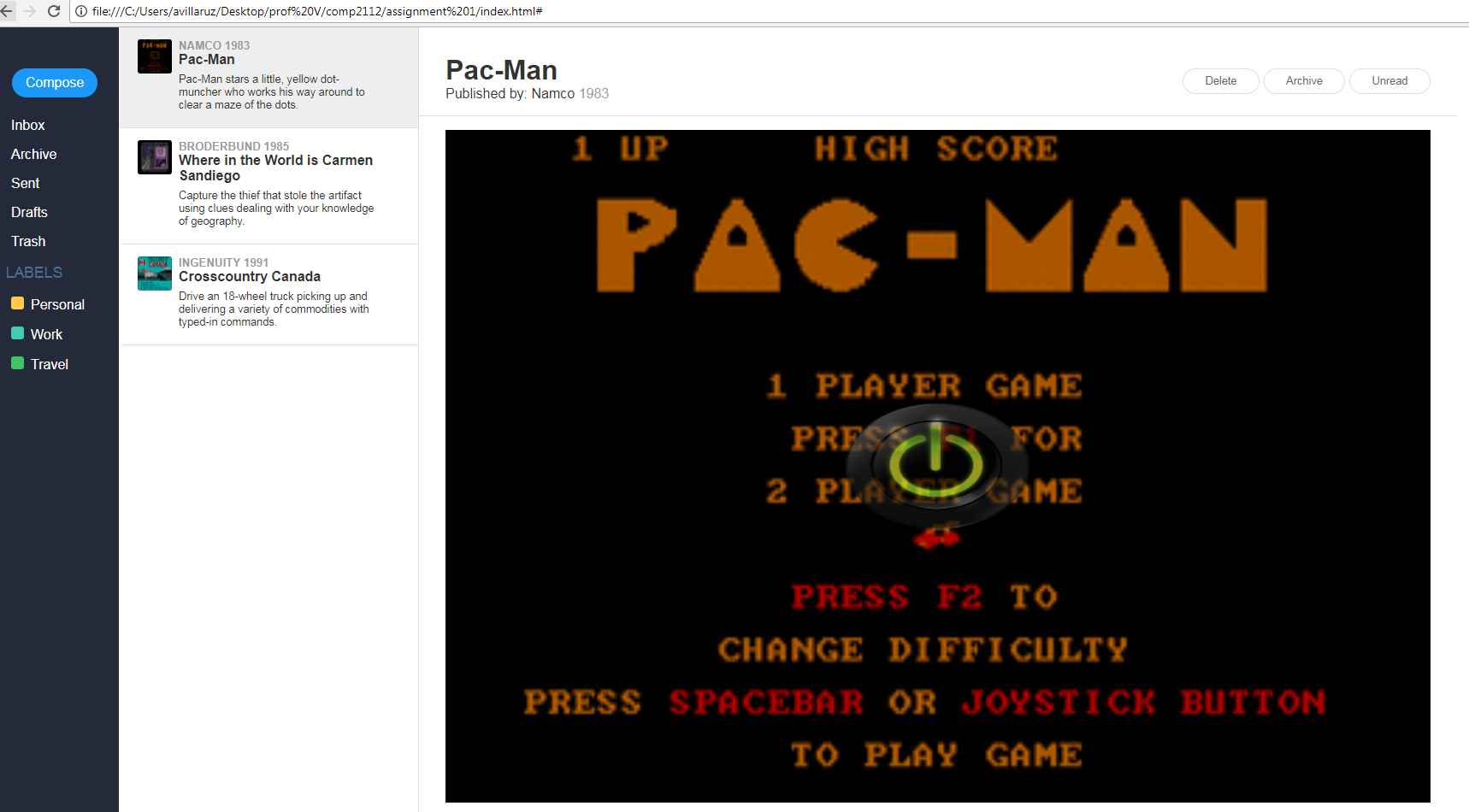
**Submission Requirements via Blackboard:**

* a single .zip file containing all your pages
* a link to your github repository where this assignment is also stored

All work must be your own. Failure to submit an independent assignment will result in a grade of zero.

# Purpose:

In this assignment you will take an existing, static email-like HTML template, and make it dynamic via Javascript.



# What you’ll need:

* HTML/CSS framework for creating webmail <https://purecss.io/layouts/email/>
* An array of objects taken from <https://archive.org/details/softwarelibrary_msdos_games>

let games = [

{'publisher' : 'Namco', 'avatar' : 'https://archive.org/services/img/msdos\_Pac-Man\_1983', 'subject' : 'Pac-Man', 'body' : 'Pac-Man stars a little, yellow dot-muncher who works his way around to clear a maze of the dots.', 'date' : '1983', 'ifrmSrc' : 'https://archive.org/embed/msdos\_Pac-Man\_1983'},

{'publisher' : 'Broderbund', 'avatar' : 'https://archive.org/services/img/msdos\_Where\_in\_the\_World\_is\_Carmen\_Sandiego\_1985', 'subject' : 'Where in the World is Carmen Sandiego', 'body' : 'Capture the thief that stole the artifact using clues dealing with your knowledge of geography.', 'date' : '1985', 'ifrmSrc' : 'https://archive.org/embed/msdos\_Where\_in\_the\_World\_is\_Carmen\_Sandiego\_1985'},

{'publisher' : 'Ingenuity', 'avatar' : 'https://archive.org/services/img/msdos\_Crosscountry\_Canada\_1991', 'subject' : 'Crosscountry Canada', 'body' : 'Drive an 18-wheel truck picking up and delivering a variety of commodities with typed-in commands.', 'date' : '1991', 'ifrmSrc' : 'https://archive.org/embed/msdos\_Crosscountry\_Canada\_1991'},

];

(see <https://github.com/avcoder/comp2112-assignment1/blob/master/README.md> )

# Application Requirements:

1. Display a set of games in the style of your student webmail but instead of emails, using game information--game title, producer, avatar picture of game, iframe of url
   1. Use of template literals to eventually output HTML code and each game object’s info for each entry
   2. Use of map (or other looping method) on template literal to display all entries
   3. Putting your HTML code in the appropriate place in the DOM via innerHTML (or similar)
2. Clicking a game in the inbox will add some kind of visual indicator to make it look selected and also, display the game in the main section
   1. addEventListener added for each entry to listen for click, when click happens, adding appropriate CSS class to make it look selected
   2. the upper section of main should show the game title, publisher, year published. In the main section, show the iframe game.
   3. avoid any ‘undefined’ from showing on publisher, year, body, and game title by testing if those fields/keys exist on the object. If it doesn’t, then gracefully output an empty string.
3. Clicking on the Compose button will open a form where you to enter data for a new game where upon submit, will add a new entry to the inbox
   1. Form shows all relevant fields
   2. After data is entered, clicking the Save button will create a new game object whose key/values match the form data, then will be added to your existing array of game objects. View is updated to reflect current state.
4. Clicking delete button will remove entry from inbox. Clicking on the trash link will allow you to view all items that have been deleted. Clicking the inbox link again will return you back to the inbox view.
   1. addEventListeners implemented for delete button, trash link, inbox link which updates its current view
   2. Deleting an entry removes it from inbox to the trash. Deleting an already trashed entry deletes it permanently.
   3. Deleting an entry changes the text on the Delete button to “Deleted”. While in trash view, clicking the “Deleted” button will change text to “Delete” and move the entry back to inbox.
5. Whether you click inbox, or trash, it always displays an accurate, current state of the array of objects.
6. Use local storage such that it stores the current state of objects (assume you’ve deleted/added entries) so if the browser refreshes, it does not return back to the view of original state of objects, but rather will return you to the same view you had before refreshing.
   1. Use localstorage to set information of the current state of objects whenever a change happens
   2. Use localstorage to get information so that a browser refresh remembers current state of object and does not default back to the original view.
7. Document each section of your script with comments. You do not need to document every single line.

**Evaluation Method**

Your work will be evaluated based on how your script performs when I load and try your web page on my local Chrome/Firefox browser.

# Conclusion

In the end, it should function like this: <https://youtu.be/re3ZkZQY2bg>

# Evaluation Criteria (max 15 marks + 2 possible bonus marks)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria** | **0** | **1** | **2** | **3** | **Marks** |
| **Display Inbox of games** | - no game entries shown in inbox | - Some template literal code correct | - Most code correct | - All game entries showing.  - Complete code correct  - Check absent key/values for ‘undefined’ before showing ‘ ‘ | **3** |
| **Selecting a game adds a CSS class to entry and shows the game in main** | - No visual indicator | - CSS class added to visually differentiate it | - game shows in main | - Smaller details like game title, date, publisher also show in upper portion of main section  - if an object has missing info, gracefully test it and if so, output an empty string instead of seeing ‘undefined’ | **3** |
| Compose button opens form to submit new game | - No form opens when compose button clicked | - Form shows | - Able to fetch values from form then add new object to existing array which then updates view | n/a | **2** |
| Functional delete button, trash link and inbox link | - No code implemented for delete button, trash link, inbox link | addEventListeners implemented for delete button, trash link, inbox link | - Deleting moves entry to the trash. | - Clicking the delete button on an inbox entry will change text from “Delete” to “Deleted” and vice versa | **3** |
| **Clicking inbox or trash displays current state of objects for that view** | - Current state of objects not stored  - state is only reflected in DOM | - Current state of objects is accurate when inbox or trash is clicked | n/a | n/a | **1** |
| **Local storage used to remember state of objects** | - No local storage used | - used local storage to set information | - working functionality of local storage to set and get information | n/a | **2** |
| **Comments** | No comments | Comments present |  | n/a | **1** |
| **\*\*\* WILD-CARD BONUS \*\*\*** | - Up to 2 bonus marks for: | Make the compose form more efficient by using string methods to extrapolate from the iframe URL -- the title, the year, and the avatar URL such that the only field forms needed are iframe URL, Publisher, and Description. | Implementing ‘Unread’ button which shows a visual indicator of the unread email.  Refer to your own student webmail to see when ‘Unread’ styling disappears  Also, show accurate number of unread emails beside ‘Inbox’ or ‘Trash’. For example: while viewing the trash, try clicking ‘Unread’ on one of the items in the trash. | | **2** |