COMP 266: Unit 4

Javascript Proposals

Richard Sullivan Andison

2026, Feb 6th

Javascript Usage Proposals

This document serves as a proposal for future changes to my online library's Javascript functionalities. I must admit that I have already applied a majority of what I intended to implement in my site design to my website before acquiring approval; this is a catalogue for each feature that is currently implemented and a proposal for further functionalities. The current site Javascript can be found and viewed on this GitHub repository, with a current working beta of the site available here (furthermore, there is an attached file with all the scripts at this time of writing). Additionally, a video is supplied to demonstrate elements of Javascript that are only implemented on 3DS systems; the video shows what special problems arise when using the 3DS browser and how particular functions address those problems. Furthermore, several sources of original code will be discussed (this includes 3DBrew.com and Simba's and Wolfyxon's 3DS GitHub repositories). Finally, a series of tasks for future changes or research will be drafted as a means of building and refactoring the currently implemented features.

There are several implementations of JavaScript that handle various aspects of the site's functionality. The general structure for Javascript files for this site is a common file that holds functionalities used across multiple pages and page-specific files that handle any unique considerations for each page (such as loading literature, updating the default menu controls, or changing a theme).

Common.js

The first file, “common.js,” provides basic data and functionality that is used across the entire site. The contents of common.js include constant values and functions used for screen centering, cookie checking and setting, theme changing, device checking, and input control. To start are the constant declarations for numeric keycodes that correspond to their respective 3DS D-Pad and “A” buttons: “LEFT,” “RIGHT,” “UP,” “DOWN,” and “A.” Next are the constant declarations for the keyboard presses “BACKSPACE,” “F5,” and “ENTER.” The final constants are the values for which the 3DS screen will center to: “centerX” and “centerY.”

Functions

The first function is the “center” function. The center function simply selects the window of the Document Object Model (DOM) and scrolls it to the x and y parameters of the centerX and centerY constants; this function is set to run intervallically after the DOM is fully loaded. The center function is called within a “setInterval” function that itself is called when the DOM’s content is fully loaded and will continuously call the function every set amount of milliseconds; currently, there is no interval set, so the function reverts to its default interval value. This function was originally copy-pasted from Wolfyxon’s GitHub repo but was modified to use constants that have different values for centering due to different means of structuring the sites’ HTML and CSS (my site has set width and height for the body element to ensure the background remains the same on both screens of the 3DS) [1].

The next set of functions are all related to checking and updating site cookies within the site, such as checking the current book, book page, and theme. The first two functions, “setCookie” and “getCookie,” are general-purpose functions for setting and acquiring the value of a cookie via their name, value, and expiration times [2]. These two functions are used to build additional functions for checking and setting the site’s three cookie values. There are two functions for checking the “bookname” and “theme” cookie values and another function for updating the theme value.

The next set of functions determines what device the site was visited from. The “includes” function is a function for determining if a string belongs within a window container, and the “is3DS” function uses the “includes” function to determine if the windows userAgent container includes the string “Nintendo 3DS”; finally, the “registerNon3DSlink” function sets a warning event listener function on clicking to any links that have an attribute indicating they are not 3DS compatible; all three of these functions originated from Wolfyxon’s GitHub repository [1].

The final common function is the “preventKey” function that limits user input to the backspace, F5, Enter, and character keys; this function allows the user of the 3DS system to use the D-Pad without screen juttering and allows desktop users to select elements with the arrow keys and Enter key [1].

Beneath the function declarations is an immediately invoked function, which itself contains an event listener that runs a block of code when the DOM is fully loaded. When the DOM is fully loaded, the theme is checked; if a theme does not exist or is not found, then the site's default theme is set. Next, a conditional is run that determines if the browser is a 3DS browser. If the conditional evaluates to true, an event listener for errors is added to display the error as an alert so 3DS users can view errors when they occur. The next block of code sets the center function to run at a certain interval. Finally, all links are stored in a variable and looped through; if a link has an "nc" attribute, the "registerNon3DSlink" function is called on that link. At the very end is the block of code that runs if the system is not a 3DS system; the document body margins are set from 50px to 10px.

"common.js" provides all the necessary and shared functionalities of each page. Future considerations may involve adding functions that are currently declared in other files if it makes sense to do such (such as adding a tab index to any elements that should be selected); this could simplify or potentially remove some files and contain code in logical chunks.

Menu.js, Settings.js, Catalogue.js, Gallery.js, and Read.js

The menu, settings, catalogue, and gallery scripts are intended to handle the control and display elements of their respective pages; the main menu (index.html) has a scrollable div of anchor links that navigate to different pages via the 3DS's D-Pad and A button; the settings script has a series of buttons with tab indexes that are selected in a similar, but not exactly the same, manner as the main menu; the catalogue script is almost the same as the menu but has additional logic for reading from a CSV file to populate anchors that update the sites "bookname" and "pagenum" cookie values before redirecting the user to the "read.html" page. The general logic for all these scripts is for setting functions of the D-Pad and A button for each menu and determining the logic that would run when a selectable element is interacted with (focus, blur, keydown/up, clicked, etc.). Much of the current logic for selecting and deselecting elements originated from Simba's (Simon Basset's) "main.js" file in their "n3dsite" GitHub repo but underwent significant modifications so that focused items update a header and subtitle value on the upper screen instead of displaying a photo (except for within the gallery page) [3]. Furthermore, selectable items are focused via a "tabIndex" attribute instead of the 3DS's default focusing system, which causes jittering if no elements are selectable in a particular direction. For handling key down events, code was copied and modified, again, from Wolfyxon's "3DSWeb-Stuff" public repository; major changes include updating the event keycode logic to a switch statement and utilizing an index system for selecting elements (to enable a looping system of selectable elements without jittering).

Broadly speaking, each Javascript file only has slight differences in its function. The menu script sets the function of the up and down D-Pad functions and adds event listeners to anchor elements; the settings script does the same, but with button elements; the catalogue script uses anchors again but has the additional step of creating them through data collected from a CSV file (and additional settings for clicking on links, such as updating the "bookname" cookie value); finally, the gallery displays images but presents focusable image elements that are displayed as a 2x2 grid on the lower screen (this required creating additional logic for D-Pad controls). Aside from general navigation, there are functions for receiving textual information via an XMLHTTP request and parsing that text to user-end formats (displaying the book catalogue in this instance); these functions utilize string manipulation and regular expressions and asynchronous event handling [4], [5], [6], [7]. The last script, "read.js," provides similar logic to the Javascript for the catalogue page but parses an HTML file using paragraph tag indexes instead of regex expressions; furthermore, the D-Pad controls page updating and scrolling (this includes updating the page number).

Conclusion

The Javascript used throughout the site provides functionality for controller input handling, information retrieval, and updating page elements. Almost every page menu has a unique aspect for handling selections:

- The main menu page simply redirects when an anchor is clicked.
- The about page simply updates the header and subtitle but does not redirect.
- The catalogue redirects and updates the “bookname” cookie.
- Items on the settings page update the “theme” cookie.
- The gallery simply displays a larger photo to the top screen.
- The read pages are indexed and displayed via the left and right D-pad controls. The upper and lower D-pad controls allow for up and down scrolling.

Furthermore, sending and retrieving data included:

- Receiving literature in HTML-formatted text.
- Receiving a current literature catalog in CSV-formatted text.

Finally, other functionalities include

- Functions for formatting the screen.
- Functions for checking device type.
- Functions for handling element focus/blur/click events.

The current site functions and performs well, but several elements of the code could be refactored to improve readability, remove redundancy, and follow better Javascript practices. To start, a better understanding of Javascript functions could be developed in order to know the difference between each particular kind of function declaration used. For example, a practice of containing the majority of a page's code within an immediately invoked function to ensure the global scope for the page is not polluted was adopted from Simba's "n3dsite" repository but requires further exploration to understand all the nuances of this practice. Another example is the assigning of functions to variables, as in the cases for the several "active" and "inactive" event functions that are used for handling focused elements across several scripts (also adopted from Simba's repository) and whether they are necessary. Additionally, several elements within the current code scheme could be functionalized or implemented better; for example, creating functions for XMLHTTP requests, setting D-Pad inputs, and standardizing selectable elements (potentially using an element ID system to set the appropriate event functions). Finally, additional features that would implement Javascript could be adding additional settings for font family and size, adding additional paging functionalities for the reading page (such as displaying the current page and allowing the user to index pages via number), and creating different modes for displaying literature via the top, bottom, or both screens.

Resources Used

[1] Wolfyxon, “GitHub - Wolfyxon/3ds-web-stuff: A collection of browser games and other stuff for the Nintendo 3DS,” GitHub, 2025. https://github.com/Wolfyxon/3ds-web-stuff

[2] W3Schools, “JavaScript Cookies,” www.w3schools.com. https://www.w3schools.com/js/js_cookies.asp

[3] Simon Basset, “GitHub - simbas/n3dsite: optimized website for 3DS,” *GitHub*, 2025. https://github.com/simbas/n3dsite

[4] A. Max, “What is the difference between fetch and XMLHTTPRequest? ⚙️,” *DEV Community*, Feb. 17, 2025. https://dev.to/hmpljs/what-is-the-difference-between-fetch-and-xmlhttprequest-1g9m (accessed Jan. 21, 2026).

This resource was used to understand the difference between XMLHTTP requests and Fetch requests. In essence, Fetch requests are an updated API for obtaining data from XMLHTTP requests. XMLHTTP requests have less features, but are more backwards compatible.

[5] “XMLHttpRequest,” *MDN Web Docs*, May 10, 2019. https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

This resource provided context to using XMLHTTPrequest object instance in Javasript

[6] Firas Dib, “Regex101 - online regex editor and debugger,” *@regex101*, 2019. https://regex101.com/

While the initial regex expression was created via an AI prompt from DuckDuckGo, it was later replaced with simpler expression built using this editor and debugger. While the original regex worked, I did not understand it’s syntax unlike the updated version.

.[7] “JavaScript Callbacks,” *www.w3schools.com*. https://www.w3schools.com/js/js_callback.asp

This resource provided context for callback functions within functions and how they can be used as a form of sequence control for asynchronous events (such as loading an XMLHTTP request response text fully).