COMP 266: Unit 6

JQuery Proposal

Richard Sullivan Andison

2025, Feb 22h

JQuery Overview and Implementation Proposal

JQuery is a very popular JavaScript library (currently in it's fourth version) that aims to simplify vanilla JavaScript code by abstracting and combining common functionalities into logical blocks that adhere to a specific syntax and methodology for web-content manipulation (a dollar sign is used to indicate a JQuery function, "$()") [1]. Aside from simplifying common functionalities, JQuery also supports a variety of plugins that can provide additional features (such as platform-specific functionalities, animation packs, and event or AJAX handling) [2], Now that all the major functionalities are implemented in vanilla JavaScript, the next task will be finding and replacing elements within the exisiting scripts that can utilize the core JQuery library to improve program readability. Furthermore, a system for implementing JQuery code must be created to assure that any potential errors that occur during the transition can be tracked and fixed; for example, early changes should be implemented on elements that have a small scope and few dependencies (such as the function for centering the screen, or listening for DOM content to be fully loaded).

JQuery implementations will begin with transitioning DOM element variables (for example, "document.getElementById("elementId")") with JQuery objects element selections ("$("#elementId")") to simplify code. One effect of this change is that JQuery syntax differs from vanilla Javascript for object operations; all logic that uses these variables must be updated accordingly (for example, "innerHTML" can now be

updated using the “.html()” method instead of “element.innerHTML = something“ instruction sequence . Most updates for functionalities will simply involve finding the equivalent JQuery function that replaces the vanilla code methods via searching JQuery’s official API documentation. After element acquisition and manipulation are replaced, other functions will be converted where possible; for example, AJAX functions (suchs as when acquiring the text for literautre), and event handling could be simplified; for example, JQuery object lists can have event listeners to all items with a single function instead of requiring a loop to iterate through and add listeners to each element [3], [4].

Additional Implementations

While not entirely related to JQuery, another change is planned for the catalogue data. Currently, catalogue items are stored in a CSV file, but this could be converted to a JSON (JavaScript Object Notation) file to be more compatible with JavaScript code [5]. Furthermore, reading and parsing the CSV file is fairly complicated and relies on a regex expression to match exactly three object values per line; updating and implementing this system elsewhere would require additional functions and regex expressions; instead, JavaScript’s built in handling for JSON files will be utilized. Finally, the websites final CSS configuration will be refactored to provide a base configuration file for both desktop and DS systems with additional theme files that can be added across both systems; this would improve the ability to standardize the overall

structure across themes, remove CSS repetition, and reduce code need to configure the site for different systems (currently, the site's CSS is updated via JQuery methods when a desktop browser is detected; instead, the website's base CSS file will simply be switched). Furthermore, having a separate CSS base for modern browser will allow for more modern CSS frameworks to be used (such as vertical heights, flexboxes, and grids).

Resources Used

[1] Fireship, "The Legend of jQuery in 100 Seconds," *YouTube*, Sep. 14, 2020. https://www.youtube.com/watch?v=UU-GebNqdbg (accessed Feb. 20, 2026).

This resource provided a brief interview of the history and purpose of JQuery.

[2] "The jQuery Plugin Registry," *Jquery.com*, 2024. https://plugins.jquery.com/

This resource provided a means for exploring official JQuery plugins.

[3] JS Foundation - js.foundation, "jQuery API Documentation," *Jquery.com*, 2019. https://api.jquery.com/

Visited extensions include:

- jQuery.get/#jQuery-get-url-data-success-dataType – Used for modifying and simplifying getText function in common.js.

- filter/#filter-elements – Used filtering font and theme buttons for specific clicking and keydown functions in common.js. (No longer implemented, but was used earlier).
- category/selectors/ - Used for getting an understanding for selecting elements by class, id, and tag.
- category/css/ - Used to understand syntax for changing a JQuery elements CSS properties.
- /Jquery.ajax/ - Used when updating getText function to more polymorphic “get” function that handles HTML and JSON formats. Furthermore, this resource was used again for understanding response data types (JSON, text, HTML).

[4] *Duckai.ai*, 2026. https://duckai.ai/ (accessed Feb. 22, 2026).

I included DuckAI as a resource because my process for implementing JQuery code typically followed a pattern of searching for logic blocks to substitute with JQuery via DuckDuckGo (for example, “how to add event listeners to multiple elements with JQueary”). Typically, an example would be shown via the search AI assist, with sources being from the official JQuery site documentation API or StackOverflow.

[5] “JSON vs CSV: what is the difference and what should I use?,” Sep. 13, 2023. https://jsoneditoronline.org/indepth/compare/json-vs-csv/

I used this as an initial resource for understanding JSON syntax.