CPSC 2030 Assignment 5: ARIA and Accessible Menu

- I have provided HTML and CSS code for this assignment. Do not modify this code unless specifically directed to in the instructions!
- I recommend using Chrome to test your assignment since it has keyboard accessibility set up by default.

Task:

- I have provided the HTML (including ARIA) and CSS for a menu and menu button. Your task is to use jQuery to make this menu functional according to W3C ARIA best practices, and according to the demo video I provided.

- Save your jQuery code in a file called **aria-menu.js** and link it to your document.

- **Guidelines:**
    - The functionality will be based on that recommended by W3C ARIA specifications:

    Menu Button Pattern: https://www.w3.org/TR/wai-aria-practices/#menubutton
    Menu or Menubar Pattern: https://www.w3.org/TR/wai-aria-practices/#menu
    Menu Button Example:
    https://www.w3.org/TR/wai-aria-practices/examples/menu-button/menu-button-links.html

    However, for this assignment, **you only need to implement the following functionality**:
        - "Menu" button:
            - Should expand the menu when clicked, or with the following keys: down arrow, enter, or space. Should also control the "aria-expanded" attribute on the menu.
        - Menu items (potatoes, swiss chard, etc…)
            - Navigate the list with up and down arrows (not with the tab key); when you reach the top or bottom of the list, focus should loop back to the bottom/top. Press enter to visit a link (default behavior). Press tab to collapse the menu and move focus to the next item on the page. (the "aria-expanded" attribute should be suitably updated.)

    - The code should be generalizable to any menu on the page. No new JS or CSS code should have to be written to add additional menus. Do not change the HTML markup for the menu. Remember the advanced CSS selectors and jQuery traversal methods.

  - Try to minimize the number of times you create jQuery objects; store jQuery objects in variables to avoid additional queries. In addition to being more efficient, this also improves readability of code.
    - You can use the above example, provided as part of the specification document, for guidance. Note that for mouse users, we will set up the menu to open when the button is **clicked** instead of hovered upon.
    - Remember that **an ARIA role is a promise**. Specifying certain roles means we must implement certain functionality.
  - Use jQuery traversal methods to move the focus throughout the menu and apply the necessary CSS classes.
    - Some methods that may be helpful: **children, find, first, parent, closest, next, prev**. These are methods that I used in my solution, but there is more than one way to solve this, and you may find other methods helpful as well.
  - Use the CSS that I've written to style the menu. (if you want to make adjustments, check with me.)
    - Particularly, notice that I've written styles for a class called "**expanded**", and figure out how to apply this class using jQuery.
  - Make sure the menu properly handles application of the **aria-expanded** attribute. **Read the specification** and study the example to see how this should be implemented.

- **Hints:**
  - Use the **on** method with the **keydown** event to listen for key presses. Use **event.key** to find out which key was pressed. Key values: https://developer.mozilla.org/en-US/docs/Web/API/KeyboardEvent/key/Key_Values
  - Remember that you can add listeners for more than one event to an element by *chaining* them! $("something").on(…).on(…) …
  - Remember that you may need to prevent the default behavior for certain keys.
  - Remember that if you find yourself copying and pasting code, you can write a function and call it instead. (Try to prevent duplication and repetition of code.)
  - To check if you're at the start/end of the list, you can try to get the next/prev item and see if the set that is returned is empty (length == 0).

Hand in

- Rename your folder to **a5_firstname_lastname,** zip it, and hand it in to the Assignment 5 folder.

Checklist

- [6 marks] keyboard functionality properly implemented
- [2 marks] aria-expanded properly handled
- [2 marks] mouse functionality properly implemented

Total: 10
As usual, the marker may deduct up to -30% for improper naming, hand in, poor indentation/coding style, etc.

Acknowledgements and References:

- Jordan Millier, Langara College
- jQuery API http://api.jquery.com

Additional Resources

- General jQuery info and API: http://try.jquery.com/