jQuery Exercises

Works on the *index.html* file

3. jQuery Basics

3.8.1. Selections

1. Selects all *div* elements that have the module class. Once selected, it displays the number of items that have been selected in the console. Using property . *lenght*
2. Select all descendants of #myList and print with . *text*() the third element.
3. Selects the *label* element of the *input* element using an attribute selector. Once done print it in the console using . *text*()
4. Selects hidden attributes and prints the number of hidden items in the console. (Help *type*=" *hidden").*
5. Prints per console the number of images that have the *alt* attribute.
6. Prints in the console . *text*() the odd elements of the body of the table (*tr* odd).

3.8.2. Traversing

1. Select all of the image elements on the page; log each image's alt attribute.
2. Select the search input text box, then traverse up to the form and add the *current* class to the form. This way, the *Go* button will be red.
3. Select the list item inside #myList that has a class of "current" and remove that class from it; add a class of "current" to the next list item.
4. Select the *select* element within #specials to make that when you press the *input* type *submit* that appears a few lines below whose *value* is *Go*, the previous selection is eliminated. Does it work? Why does it make a *submit*? goes back with the browser when it doesn't find the page and verifies if it erased or not the *select*. See documentation on the $. *fn.preventDefault*() function.
5. Select the first item in the list in the #slideshow element; add the *current* class to it and then add the *disabled* class to the sibling elements.

3.8.3. Manipulating

1. Add five new list items to the end of the unordered list #myList . Hint: use a loop.
2. Remove odd items from the #myList
3. Add another *h2* element and another paragraph to the last *div.module*
4. Add another option to the *select* element; give the added option the value *Wednesday*.
5. Add a new *div.module* to the page after the last one; then add a copy of one of the existing images inside the new *div*. (This last point you must comment when you do the exercise 5.7.2 paragraph 3 as it interferes with their behavior.)

5. Events

5.7.1 Create an Input Hint

Your task is to use the text of the label for the search input to create "hint" text for the search input. The steps are as follows:

1. Modify the placeholder of the input of the search box so that it is equal to the value of the label element.
2. Add a class of "hint" to the search input.
3. Remove the label element.
4. Bind a focus event to the search input that removes the hint text and the "hint" class.
5. Bind a blur event to the search input that restores the hint text and "hint" class if no search text was entered.

5.7.2. Add Tabbed Navigation

The task is to create a tabbed navigation for the two *div.module* elements. The steps to follow are as follows:

1. Hide all of the div.modules.

1. Create an unordered list element before the first module.
2. Iterate over the modules using `$.fn.each`. For each module, use the text of the h2 element as the text for a list item that you add to the unordered list element.
3. Bind a click event to the list item that:
    - Shows the related module, and hides any other modules
    - Adds a class of "current" to the clicked list item
    - Removes the class "current" from the other list item
4. Finally, show the first tab.

6. Effects

6.5.1. Reveal hidden text

Your task is to add some interactivity to the blog section of the page. The spec for the feature is as follows:

- Clicking on a headline in the #blog div should slide down the excerpt paragraph
- Clicking on another headline should slide down its excerpt paragraph, and slide up any other currently showing excerpt paragraphs.

Hint: don't forget about the `:visible` selector!

6.5.2. Create a drop-down menu

Your task is to add dropdowns to the main navigation at the top of the page.

- Hovering over an item in the main menu should show that item's submenu items, if any.
- Exiting an item should hide any submenu items.

To accomplish this, use the `$.fn.hover` method to add and remove a class from the submenu items to control whether they're visible or hidden. (The file at `/exercises/css/styles.css` includes the "hover" class for this purpose.)

6.5.3. Create a *slideshow*

Your task is to take a plain semantic HTML page and enhance it with JavaScript by adding a slideshow.

1. Move the #slideshow element to the top of the body.
2. Write code to cycle through the list items inside the element; fade one in, display it for a few seconds, then fade it out and fade in the next one.
3. When you get to the end of the list, start again at the beginning.

For an extra challenge, create a navigation area under the slideshow that shows how many images there are and which image you're currently viewing. (Hint: $.fn.prevAll will come in handy for this.)