



420-311-VA Internet Programming

Assignment #5

 Vanier College 
Computer Science Technology Department
Teacher: Hanif Azhar

Reference: Sleiman Rabah

 **Due date: as scheduled on LEA** 

Contents

| | | |
|----------|--|----------|
| 1 | A Special Note | 2 |
| 2 | Learning Objectives | 2 |
| 3 | Online Resource and Documentation | 2 |
| 4 | Problem Statement | 3 |
| 5 | Implementation Details | 3 |
| 5.1 | Requirements and Constraints | 3 |
| 5.2 | Files and Folder to Create and Use | 4 |
| 5.3 | Steps to Follow | 4 |
| 5.4 | Constructors to Be Implemented | 4 |
| 5.5 | Objects to Be Created | 5 |
| 5.6 | Arrays to Be Created | 5 |
| 5.7 | Functions to Be Implemented | 5 |
| 5.8 | The Expected Web Interface | 8 |
| 6 | Evaluation criteria and Instructions | 8 |


7 What to Submit

8

1 A Special Note


The instructions presented in this assignment require special attention, so please take the time to read the present document multiple times. Again, you must read the entire document multiple times before you start writing code.

You should write down the steps to be performed (that is, what needs to be done) before you start coding. Note that subsequent assignments might be based on this one.

NOTE: remember that this is an individual assignment. You are asked to write your own `</>` (code). Any similarity detected in any assignment(s) submitted by another student(s) will result in serious repercussions. Again, you have to write your own code. You are not allowed to share code with, or write code for, or look  at code of, another student. If you do so, you will be seriously penalized. If you have any questions, ask me for help/clarifications.

2 Learning Objectives

The main focus of the work to be done in this assignment is the use of JavaScript **objects** and **arrays**. However, this assignment involves the use of, but not limited to, the following:

- Event-driven programming paradigm: using JavaScript events.
- Dynamic access to, and change the content of, HTML elements and their attributes.
- DOM (Document Object Model) API.
- Public JavaScript functions and objects.
- An open source CSS framework: ([Bootstrap \[1\]](#)).
- An open source vector-based icons framework:  [Font Awesome \[2\]](#).

3 Online Resource and Documentation

- * [Bootstrap Components](#) : where all Bootstrap's components are documented.
- * The list of all FA icons: [Font Awesome Icons](#) and usage examples: [Font Awesome Examples](#)
- * [Bootstrap Table](#) , [Bootstrap Cards](#) , [Bootstrap Buttons](#)
- * [Working with objects](#)
- * [Working with indexed collections](#)

✱ [Defining functions](#)

4 Problem Statement

In this assignment, you are asked to implement an online store for selling specific items of your choice. The desired web application must:

- Enable the user of the online store to browse the list of available items (that is, the catalog of items for sale).
- Allow the user to search for an item contained in the catalog of items.
- Provide the user with features to add/remove an item to/from his/her shopping cart as well as view the list of items that are stored in his/her shopping cart [🛒](#).
- Enable the user to see (that is, visualize) the details about a specific item (for example, by clicking on a *view details* button).

NOTE: Since this course focuses only on JavaScript and we are not using a database, or browser-enabled capabilities as local storage (cookies/session storage) the application must be implemented within a single webpage and information about the catalog's items and content of shopping cart must be stored using JavaScript objects and arrays. Therefore, bear in mind that any page reload/refresh will erase the content of your shopping cart.




5 Implementation Details

This section provides you with details about what needs to be implemented and the steps to be followed and performed.

5.1 Requirements and Constraints

Your implementation must **absolutely** respect the following requirements:

- ✱ Use HTML/HTML5, [Bootstrap \[1\]](#) , and [Font Awesome \[2\]](#) to design and implement the web interface.
- ✱ Use JavaScript only: unauthorized use of external code or libraries (such as jQuery) is strictly forbidden. You are required, and expected, to write your own code.
- ✱ You are not allowed to use the *document.write()* function in this assignment.
- ✱ Use only *getElementById()* function and *innerHTML* property to dynamically change/update the content or the attribute's value of any HTML element.
- ✱ **Proper error and exception handling.** Please refer to the following online resources [JavaScript error handling \[3\]](#) , and [JavaScript Control flow and error handling \[4\]](#) .

- * Use the [Camel Case](#) [3] naming convention in your implementation.
- * Choose consistent variable names. For instance, naming an array of numbers *myArray* will result in a grade deduction.
- * You must document your code (this include the documentation of each and every function) see [JSDoc](#) [5] for more details.
- * Your script should alter the content of an HTML document in order to display different kinds of content (the list of items, result of user searches, shopping cart's content, information about an item). However, *console.log()* can be also used for debugging purposes.
- * As a web browser, use  (Google Chrome) or  (Mozzila Firefox). Please do not use  (Microsoft IE/Edge).
- * You can use global variables in this assignment.

5.2 Files and Folder to Create and Use

The following are the main files to be created and used:

- * Create an HTML file named *index.html* and link both **Bootstrap** and **Fontawesome** to it.
- * Create a JavaScript file named *cart.js* and placed in a folder named *js*. You must write your functions in this file and link it to *index.html*
- * Create a folder named *images*.

5.3 Steps to Follow

The following are the steps to be done in order (one after the other):

1. Download the required Fontawesome files (or copy them from Assignment #4)
2. Create the required files and folder as detailed in Section 5.2.
3. Implement the required constructor functions (see Section 5.4).
4. Create the required arrays and fill them with objects as instructed (see Section 5.5 and Section 5.6).
5. Implement the required functions.
6. Design and implement the web interface.

5.4 Constructors to Be Implemented

- * **Category:** a category object represents a specific category of items. For instance, laptops, desktop PC, computer components, tables, etc.

This object must have the following properties: *category ID*, *category name*, and *description*.

- ✱ **Item:** an item object must be created using a constructor function. An item object must have the following properties:
 - item ID: an integer of your choice (must be unique when added to the array of items).
 - item title: short description of the item in question.
 - description: long description of the item.
 - brand: the name(s) of the author(s) (can be one or more).
 - unit price: the unit price of the item [*float*].
 - quantity in stock: how many items are available in stock [*integer*].
 - make: the name of the manufacturer.
 - thumbnail image: (a string that stores only the path or the file name of the image)
 - category: an object that represents the category to which the current item belongs to. A “*has a*” relationship (between an item and a category) must be implemented in the **Item** constructor.

5.5 Objects to Be Created

1. You must create at least 8 different category objects.
2. You must create at least different 20 items belonging to 8 different categories.

5.6 Arrays to Be Created

You must create (in the global scope) and use the following global arrays:

1. **categories:** an array of *category* objects. This array represents the list of all the available categories of items.
2. **catalog:** an array of *item* objects. This array represents the list of all the available items in the store.
3. **cart:** an array of *item* objects. This array represents the shopping cart. It contains the items that might be added by the user of the online store.

5.7 Functions to Be Implemented

The following functions are to be implemented. However, you are encouraged to implement other function(s) if you judge necessary.

1. ***initializeCategories()***: a parameterless function that creates at least 8 category objects and stores the global array named *categories*.
2. ***initializeItems()***: a parameterless function that creates at least 20 items belonging to different categories and stores them in a array named *catalog* (see Section 5.6).
3. ***setUpCart()***: a parameterless function that must be called on page load (*hint*: use the *onLoad* body event as explained in class). This function must do the following (respect the order of function calls):
 - a) It calls *initializeCategories()*, as detailed above.
 - b) It calls *initializeItems()*, as detailed above.
 - c) It calls *showListOfItems()* in order to display the list of items the first time the user open the web page. At this point, you can write the *showListOfItems()* function and call it just for testing purposes. Later you can change what *showListOfItems()* does.
4. ***showStatusMessage(cssClass, messageToDisplay)***: when it is called, it shows a [Bootstrap Alerts](#) status message with the specified *cssClass* and the *message to display* as parameters.
NOTE: The message should be shown for 5 seconds only.
5. ***hideStatusMessage()***: it hides the [Bootstrap Alerts](#) status message from the HTML document. You should use the CSS display property to hide the message as explained during our lab meetings.
6. ***showListOfItems()***. this function does not receive any parameters. It processes the *catalog* created in Section 5.6 This function must be called only when the HTML document is loaded. The purpose of this function is to show the list of all the available items with the possibility to add an item to the shopping cart. The items must be displayed using **Bootstrap cards**.
Hint: when you are listing the items, you should also add a button named *add* to the Bootstrap Card. Upon clicking on the *add* button, the selected item must be added to the cart and the content of the cart must be dynamically updated.
7. ***viewCart()***: a parameterless function that displays the content of the shopping cart (that is, the list of items that have been added to the cart) using a bootstrap table (see Section 3). It should be called upon clicking on the view cart button.
8. ***findItemById(itemId)***: This function receives a item ID as a parameter and performs a search by item ID in the *catalog array* in order to find out if there possibly is a matching item. It returns the matched item object if found; otherwise, it returns undefined. Note that the search operation should be case insensitive.

9. ***showItemDetails(itemId)***: This function receives a item ID as a parameter and display its information. You can use a Bootstrap table to show all the information about the selected item. You need to use the *findItemById* function in order to find the item object.
10. ***addToCart(itemId)***: This function receives a item ID as a parameter, and adds the selected item (the object representing the selected item) to the cart.
 - This function must call *findItemById(itemId)* in order to find the object that belongs to the selected item and add it to the ***cart*** array.
 - If the selected item has been successfully added to the cart, your application must show a notification/confirmation message on the document (see *showStatusMessage* function).
11. ***removeFromCart(itemId)***: This function receives an item ID as a parameter, and removes the selected item from the cart. The content of the shopping cart must be redisplayed after removing a item.

NOTE: If the selected item has been successfully removed from the cart, your application must show a notification/confirmation message on the document (see *showStatusMessage* function). The *cssClass* that is to be passed *showStatusMessage()* should be: 1) Bootstrap-related alert CSS class; 2) and should be different than the one used in *addToCart()*.

12. ***searchByKeyword(searchKeyword)***: that receives the search keyword as parameter. This function must be called upon clicking the *search* button in the search form. Your application should call this function and search in the list of items by category name, item description, item title, or any other item property of your choice.

The list of the matching items (the result of the search) must be dynamically displayed in an HTML table. The table must list the item id, item title, item description, category name, and two buttons: 1) view details; and 2) add to cart.

NOTE: If your search does not yield any results, your application must show a notification message on the document (see *showStatusMessage* function). The *cssClass* that is to be passed *showStatusMessage()* should be: 1) Bootstrap-related alert CSS class; 2) and should be different than the ones used in *addToCart()* and *removeFromCart()*.

5.8 The Expected Web Interface

The Web interface must be designed and implemented using [Bootstrap \[1\]](#) and [Font Awesome \[2\]](#). It consists of a single HTML document whose main content will be dynamically updated according to the user's actions.

All what you have to do is:

1. Add the following Bootstrap components combined with [Font Awesome Icons](#) :
 - a) *view items* button: to show the list of all items.
 - b) *view cart* button: to show the content of the shopping cart.
 - c) *search box*: an HTML *input* element for entering a search keyword.
 - d) *search* button: to search in the catalog by the entered search keyword.

6 Evaluation criteria and Instructions

Please read carefully, failing to respect the provided instructions will result in a major grade deduction.

- * The use of file names and functions exactly as provided.
- * Relevance and accuracy of the source code documentation as instructed in [5.1](#).
- * Completeness, correctness, and overall functionality of the implementation.
- * Compliance of the implementation with the stated problem as well as simplicity and appropriateness of your implementation.
- * Programming style, code readability, naming convention, and clarity see [5.1](#).
- * The user-friendliness of the web interface (presentation of the output).
- * Relevance of the test scenarios.
- * Proper validation and error handling.
- * Correctness of data validation.

7 What to Submit

- Compress all your files (.html, .js, .css) and upload them to LÉA before the submission deadline.

References

- [1] Bootstrap Project, "Bootstrap CSS Framework." <http://getbootstrap.com/>, 2020. [Online; accessed 1-Oct-2020].
- [2] Font Awesome Project, "The iconic font and CSS toolkit." <https://fontawesome.com/>, 2020. [Online; accessed 18-Oct-2020].

- [3] W3 Schools, “JavaScript Errors Handling.” https://www.w3schools.com/js/js_errors.asp, 2020. [Online; accessed 18-Oct-2020].
- [4] Mozilla Web Docs, “Control Flow and Error Handling.” https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Control_flow_and_error_handling, 2020. [Online; accessed 03-Sep-2020].
- [5] The JSDocs Documentation Project, “JSDoc.” <http://usejsdoc.org/>, 2020. [Online; accessed 03-Sep-2020].