Assignment - Applying strings in your e-commerce project

Exercise

Task description

Your task is to program a set of small and useful functions in JavaScript that will be executed by clicking buttons on the pages of an e-commerce project.

Purpose

The purpose of this task is to apply the key concepts related to working with strings and objects in JavaScript through practical functions used in an e-commerce environment.

Tasks

1. Improve the previous task

In the previous task, you used a `RAW_COUPON` variable that contained only one valid coupon (for example, "`SAVE10`"). Now expand the load so that there are multiple valid coupons stored in a string:

- Instead of a `RAW_COUPON` variable, define a `VALID_COUPONS` string that contains at least three string values, for example, `"SAVE10", "SAVE15", and "FREESHIP".`

Note: All values in the string must be capitalised, without spaces, to match the value returned by the normalizeCoupon function.

2. Create a feature that checks if a coupon exists in the string:

```
function isValidCoupon(code) { // ... }
```

- The code parameter represents the already normalised coupon (the result of the `normaliseCoupon` function).
- The function must check if the code value is found in the `VALID_COUPONS` string.
- If found, the function should return `true`. If not, return false.

Note: It is allowed to use a loop (e.g., 'for' or 'for... of') or the includes() method to traverse the string.

3. Change the `validateAndNotify()` function, which next needs to:

- Read the value from the `#promo-input` field.
- call `normalizeCoupon`(code) for this value;
- Check if the coupon is valid by calling the `isValidCoupon(code)` function.
- Display a message that corresponds to the user through an alert.

4. Make different messages appear for different coupons, for example:

- `"SAVE10" → "Your` coupon offers a `10% discount."`
- `"SAVE15" → "Your` coupon offers a `15% discount."`
- `"FREESHIP" → "Your` Coupon" `offers free shipping."`

5. Create an allProducts associative string with:

- the name, price and quantity of all the products in your e-commerce store (enough to have up to 10 products).

6. Write a function that calculates the total value of all products in stock:

- For each product, calculate `price *` qty.
- Add all the values into a `totalValue` variable.
- display in the cantilever, for example, "Total stock value: $5420."

7. Create a new lowStock string that:

- contains only products for which `qty < 10`;
- Once you've formed the `lowStock` string, display it in the console.

8. Search product by name – write a function:

```
function findProductByName(list, searchName) { // ... }
```

- The search must be case-insensitive.
- The function returns:
  - the product item found, if any;
  - or null if there is no product with the given name.
- Test the feature in the console.

Tips for successfully performing the task

1. Check that you have removed the old variable `RA and BE` in `CAPITAL LETTERS`, without spaces, because the `normalizeCoupon` function does just that with user input. Example for console self-checking:

```
console.log(VALID_COUPONS);
```

2. `isValidCoupon(code)` function – use a `FOR` loop to go through all coupons.

Or use the `includes()` function, which can search in the string.

3. Modifying the `validatedAndNotify()` function: the most common mistakes are forgetting to normalise the coupon call or forgetting to pass the normalisation result to `isValidCoupon`.

4. Don't write 3 completely separate `'if'` scattered through the code.

The most common variants are the `if /else if/else` blocks.

5. Always check first if the coupon is valid (before looking for the concrete message).

6. `AllProducts` associative string: check if all elements are between [ ], if there is a comma between objects, if each object has the SAME fields `(name, price, qty)`, and not to have quantity sometimes and sometimes `qty`.

7. When calculating the total stock value, if you get `NaN`, it means that somewhere the `price` or `quantity` is not treated as a number (or is not spelt correctly).

8. Tip for `findProductByName(list, searchName)`: normalise both the product name and the search term in the same way (e.g., `toLowerCase().trim()`) to have a case-insensitive search.

Evaluation process

The course coordinator will check your paper and provide feedback within 24 hours. If you receive comments and suggestions, you'll only have the opportunity to edit once and resubmit your work.

Evaluation criteria

- Working with coupons and the coupon string

Is the string `VALID_COUPONS` defined correctly (minimum 3 coupons, capital letters, no spaces)?

Does the `isValidCoupon` function correctly check if the coupon exists in the string?

- Functions and logic

Do the functions `normalizeCoupon`, `validateAndNotify`, `isValidCoupon` and `findProductByName` work according to specifications?

Does it return expected values and execute the expected actions?

- Validation and branching

Does the `validateAndNotify` function correctly normalise the input, check the coupon, and display the corresponding alert messages?

Are branches used correctly `(if / else if / else)`?

- Working with associative strings and data processing

Is the `allProducts` string formed correctly?

Does the stock value calculation function work correctly?

Is the `lowStock` string formed correctly, containing the products with `qty < 10`?

- Product search

Does the `findProductByName` function perform `case-insensitive` searches correctly?

Does it return the corresponding result (`object` or `null`)?

- Code quality and readability

Is JavaScript code clean and easy to read?

Are comments used to explain functions?

Sending the task

Once you finish the task, upload the work in zip format on the same page where the task description is located.

Save the task file as: prenume_nume_assignment02.zip

The maximum file size that can be uploaded in this way is 50 MB. If you exceed this size, use the WeTransfer website.

How to use WeTransfer

1. Open the wetransfer.com site.
2. Click on "Add your files" and select the files you want to send – the zip file is mandatory.
3. Add your email address.

4. At Title, enter the same name as the file with the task.
5. In the message, mention the name and surname and the name of the course.
6. The deadline for downloading the file is 3 days.
7. Click on the three dots and choose the "Create link" option.
8. Then tap on "Get a link"".
9. Enter the verification code you received at your email address.
10. You send the generated link to the course coordinator on the assignment submission page.