FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020

fondo sociale europeo

TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Web e Mobile App Developer

Unità Formativa: Programmazione web - Javascript

Docente: Shadi Lahham

Titolo argomento: Progetto Expiry List

in collaborazione con:

MINISTERO DELL'ISTRUZIONE, DELL'UNIVERSITÀ E DELLA RICERCA

UNIONE EUROPEA

REGIONE PIEMONTE

per una crescita intelligente, sostenibile ed inclusiva

www.regione.piemonte.it/europa2020

INIZIATIVA CO-FINANZIATA CON FSE

Aim

Write a javascript program that outputs a list of supermarket goods filtered by expiry date

Requirements

Supermarket rules

1. Expired items should be removed
2. Items that have been on the shelf for more than N weeks should be removed
3. Each week M new products arrive.
4. The program should start from the current date plus K days and run for X weeks
5. Each weekly list should be printed after a duration of R seconds
6. N,M,K,X,R should be configurable by the supermarket manager and should have *meaningful* variable names

Item states

1. new - the item has arrived this week
2. valid - the item is not expired and has been on the shelf for LESS than N weeks
3. old - the item is not expired but has been on the shelf for MORE than N weeks
4. expired - the item has expired (the date is older than the current week date)

Item structure

Each item should have:

1. a unique ID
2. a name
3. an expiration date
4. any other properties that you think are needed

Guidelines

- The ID should be unique!
- The ID should be a zero padded number e.g. 01 or 001 (should be configurable)
- The names should be randomly generated from a given set
- The expiry dates should be randomly generated
- The expiry dates should be between the start and finish date of the weekly runs
- All the names and states should be padded to have the same length
- The padding character MUST be configurable
- The date should be exactly formatted e.g 03-JUN-2021
- In general the printout to the console should always be aligned properly

Important

- Variables should have meaningful names
- The code should be well documented
- The code should be well indented
- The data structures should be effective and the code should be efficient
- You need to be able to explain each line of your code
- Your output should match the sample outputs provided

Bonus (extra points)

- Bonus 1: make the duration R a random number between MIN and MAX which are configurable settings in the config object
- Bonus 2: use colors in the console log output
- Bonus 3: accept a date format in the config object (user defined) and format the dates accordingly. Don't use *moment.js*

Help

Plan your data structures and functions before starting

Use arrays, objects and any other structures you think are necessary

Use a config object to store all the manager settings

IIFEs

Use IIFEs to make your code in multiple javascript files scope-safe (requirement)

Immediately invoked function expression

I Love My IIFE

Immediately-Invoked Function Expression (IIFE)

The many ways to write an IIFE

Suggestions

You will probably need to write functions such as these:

- unique

- padNum

- padString

- generateName

- generateExpiry

- addDays - adds n days to a date

- formatDate

- printProduct or printProducts

- updateState

- checkProduct

- add any other function that you need

Compatibility

The project should be tested and work properly on: **Chrome, Firefox, IE11**

Compatibility with other browsers is a nice **bonus**

Libraries, frameworks and language features

Do not use any external libraries or frameworks.

You have to write all the code yourself.

If you use any language features not seen in class, they have to be justifiable, correct and tested.

Documentation and validation

Comments and code documentation

- All HTML, CSS and JS files should contain comments and be well documented
- JSDoc header documentation for every file
- JSDoc documentation for every function
- CSS files (if used) should have a header and contain comments where needed
- HTML files should contain comments to indicate important sections
- Follow all the comments and documentation requirements in Appendix 01

Validation

HTML files should be validated https://validator.w3.org/

CSS files should be validated https://jigsaw.w3.org/css-validator/

JSDoc documentation (required)

- Generate a JSDoc documentation for your code and put it in a folder called /JSDoc

Readme

Include a readme.md file that includes at least the following sections

- Introduction / Project description
- Usage (how to set up, run and use the application)
- Configuration and technical characteristics
- Files and project structure
- Features delivered
    - feature 1: description of feature 1
    - feature 2: description of feature 2

- feature 3: missing

- Bonuses delivered

    - Bonus 1: description of bonus 1

    - Bonus 2: missing

- Browser compatibility

    - IE11: tested and fully compatible

    - Chrome v##.##.##: tested and partially compatible - feature x not working

    - Opera v##.##.##: not tested or not functional

- External resources (Links and description of external resources such as JSON files, APIs, DBs, etc)

- License and contact information

- Authors: names, roles and team composition

- Changelog and version history

- Any other information that you think is important

Output samples

Your application should be able to output the attached output samples and other similar outputs

Each has a different set of initial configuration settings