Ejercicio Entregable 3

Javascript y las funciones de orden superior

El ejercicio de esta hoja es entregable y puede realizarse en el laboratorio. Los ejercicios entregables realizados a lo largo del curso computan el 10 % de la nota final, pero su entrega no es obligatoria para aprobar la asignatura. La entrega se realiza por parejas.

Fecha límite de entrega: 2 de noviembre de 2017.

La entrega se realizará a través del Campus Virtual.

En este ejercicio entregable implementaremos algunas operaciones en Javascript mediante el uso de las funciones de orden superior vistas en clase. Todas estas funciones recibirán un array de tareas, donde cada tarea se representa mediante un objeto con tres atributos: un atributo `text` que contiene el texto de la tarea, un atributo booleano `done`, que indica si la tarea ha sido finalizada o no, y un atributo `tags`, que es un array que contiene las etiquetas de la tarea. El atributo `done` es opcional y, si no se indica, se considera que tiene el valor `false`. Por ejemplo, aquí tenemos el array correspondiente a la lista de tareas del ejercicio entregable anterior:

```
let listaTareas = [
  { text: "Preparar práctica PDAP", tags: ["pdap", "practica"] },
  { text: "Mirar fechas congreso", done: true, tags: [] },
  { text: "Ir al supermercado", tags: ["personal"] },
  { text: "Mudanza", done: false, tags: ["personal"] },
];
```

Se pide implementar las siguientes funciones utilizando **exclusivamente** funciones de orden superior sobre arrays, excepto `forEach`. No es posible el uso de bucles para implementar estas funciones.

1. Escribe una función `getToDoTasks(tasks)` que devuelva un array con los textos de aquellas tareas de la lista de tareas `tasks` que no estén finalizadas. Por ejemplo:

   ```
   console.log(getToDoTasks(listaTareas))
   // Imprime:
   //  [ 'Preparar práctica PDAP', 'Ir al supermercado', 'Mudanza' ]
   ```

   Indicación: Utiliza `map` y `filter`.

2. Escribe una función `findByTag(tasks, tag)` que devuelve aquellas tareas del array `tasks` que contengan, en su lista de etiquetas, la etiqueta pasada como segundo parámetro. Por ejemplo:

   ```
   console.log(findByTag(listaTareas, "personal"))
   // Imprime:
   // [ { text: 'Ir al supermercado', tags: [ 'personal' ] },
   //   { text: 'Mudanza', done: false, tags: [ 'personal' ] } ]
   ```

Indicación: Utiliza `filter` e `indexOf`.

3. Implementa una función `findByTags(tasks, tags)` que devuelva aquellas tareas del array `tasks` que contengan al menos una etiqueta que coincida con una de las del array `tags` pasado como segundo parámetro.

   ```
   console.log(findByTags(listaTareas, ["personal", "pdap"]))
   // Imprime:
   // [ { text: 'Preparar práctica PDAP', tags: [ 'pdap', 'practica' ] },
   //   { text: 'Ir al supermercado', tags: [ 'personal' ] },
   //   { text: 'Mudanza', done: false, tags: [ 'personal' ] } ]
   ```

   Indicación: Utiliza `filter`, `some` e `indexOf`.

4. Implementa una función `countDone(tasks)` que devuelva el número de tareas completadas en el array de tareas `tasks` pasado como parámetro.

   ```
   console.log(countDone(listaTareas))
   // Imprime: 1
   ```

   Indicación: Utiliza `reduce`.

El siguiente ejercicio tiene como objetivo el uso de las funciones sobre expresiones regulares. Para ello es recomendable que consultes la documentación sobre el método `exec` de las expresiones regulares, y el método `replace` de las cadenas. En este último ejercicio puedes utilizar bucles, si lo deseas.

5. Implementa una función `createTask(texto)` que reciba un texto intercalado con etiquetas, cada una consistente en una serie de caracteres alfanuméricos precedidos por el signo `@`. Esta función debe devolver un objeto tarea con su array de etiquetas extraídas de la cadena `texto`. El texto de la tarea resultante no debe contener las etiquetas de la cadena de entrada.

   Por ejemplo:

   ```
   console.log(createTask("Esto es una cadena @de @texto"));
   // Imprime:
   // { text: 'Esto es una cadena', tags: [ 'de', 'texto' ] }

   console.log(createTask("Y por aquí va otra @personal"));
   // Imprime:
   // { text: 'Y por aquí va otra', tags: [ 'personal' ] }
   ```

   Indicación: Puedes suponer que todas las etiquetas están al final de la cadena de entrada. Una vez hayas extraído el texto de la cadena, utiliza la función `trim()` de las cadenas para eliminar los espacios de ambos lados.