Práctica 3

Javascript (ES6)

Objetivos:

- Entender:
  - Objetos JS con funciones
  - Clases JS

Referencias:

- Herencia y la cadena de prototipos
- POO, Herencia, constructores y prototipos:
  - Parte 1
  - Parte 2
  - Parte 3
- Objeto this en javascript

Ambiente de programación:

1. Pueden usar visual code, editar el archivo main.js y correrlo en el browser.
2. Pueden usar https://repl.it/languages/babel .Si se hacen una cuenta pueden guardar el progreso.
3. También pueden utilizar https://jsfiddle.net/ y abrir el debugger del navegador.

En esta práctica veremos cómo se implementan los diferentes conceptos del paradigma de objetos en JS.

Solo utilizaremos dos de las posibles formas de programar con objetos en JS: Prototipos y Funciones (que aplican tanto a ES5, como a ES6), y la notación de clases de ES6.

A lo largo de la práctica iremos extendiendo nuestro modelo.

Si bien las relaciones y el protocolo es bastante evidente a partir del enunciado cuando se pide que diseñe considere que uds. agregar protocolo o relaciones de conocimiento entre objetos que sea necesarias para su solución.

Ejercicio 1:

En una red social los usuarios hacen posts. En este primer ejercicio nos interesa modelar solo eso. De los usuarios se conoce:

- Nombre y apellido
- Nickname
- Fecha de nacimiento (puede utilizar objetos Date, ya definidos en JS).

Cada usuario puede hacer posts, y los conoce, es decir, a un usuario le podemos pedir todos sus posts. Es razonable también pensar que a un post se le puede preguntar el autor del mismo.

De un post se conoce:
- Fecha y hora (idem anterior)
- Contenido: un string

El protocolo para esta operación podría ser algo como:

```
aUser.post('Que lindo es JS');
```

donde `aUser` es el usuario que postea, y el objeto post se crea a partir del string pasado como parámetro

Tareas
1) Diseñe (realice un diagrama de clases UML)
2) Implemente utilizando prototipos y funciones (Common JS)
3) Implemente utilizando clases ES6.

Ejercicio 2:

En la misma red social ahora se agrega la posibilidad de tener amigos. Usuario 0 o muchos amigos. Un usuario debe poder agregar o sacar usuarios a su lista de amigos.

El hecho de tener amigos implica que cuando un usuario hace un post todos sus amigos se enteran, es decir reciben un mensaje que les informa que alguien hizo un post.

Un posible protocolo sería:

```
user.postDoneBy(aUser, aPost)
```

Donde: `user` es el usuario que se está enterando del post. `aPost` es lo que alguien posteó, `aUser` es quien ha posteado.

Cuando un usuario postea algo debe mostrarse un log en la consola: quien lo hizo, qué posteó y cuando.
Cuando un usuario se entera que un amigo posteó algo, debe mostrarse en la consola que recibió esa notificación, cuál fue el post y quien lo hizo.

Tareas

4) Defina los cambios necesarios, defina el protocolo necesario para los objetos y diseñe (realice un diagram de clases UML)
5) Implemente utilizando prototipos y funciones (Common JS)
6) Implemente utilizando clases ES6.

Ejercicio 3:

Nuestra red social sigue creciendo y ahora debe ser posible comentar los posts.
Los comentarios son como posts tienen un texto, pero con la limitación de solo poder tener 100 caracteres.
Los comentarios conocen el post al cual refieren, también conocen la fecha, hora y autor del comentario.
Un post puede tener muchos comentarios.
Los comentarios no pueden tener comentarios ni respuestas.

Cuando un usuario comenta un post de alguien, el "alguien" debe ser notificado. Un protocolo sugerido para esto sería:

```
aUser.postHasBeenCommentedBy(post, comment)
```

Donde aUser es el autor del post original, quien está siendo notificado. Comment es el comentario que se está notificando. Note que al comentario le podemos preguntar el autor para saber quien comentó.

Cuando un usuario hace un comentario debe loggearse a la consola.
De la misma manera, cuando un usuario se entera que un post suyo ha sido comentado, debe loggearse.