Programación de software.
Centro de Comercio y Turismo.
Regional Quindío. 2020
Instructor: Germán Alberto Angarita Henao

TALLER TYPESCRIPT - SECCIÓN 3

Promesas:

- Cree y ejecute una promesa que siempre se resuelva, y que lo haga con un string “Somos programadores, hacemos mover el mundo”, de tal manera que en su .then, se imprima tal string mediante console.log
- Cree y ejecute una promesa que siempre se rechace, y que lo haga con un string “Ha ocurrido un error desconocido.”, de tal manera que en su .catch, se imprima tal string mediante console.log
- La entrega de subsidios en una institución pública depende de si el estrato de la persona es menor o igual a 2. Cree un programa que implemente una variable que guarde el estrato del usuario, y mediante la evaluación de tal variable, muestre si el usuario tiene derecho o no a un subsidio usando promesas, de tal manera que si la variable contiene un valor numérico entre 1 y 6, la promesa se resuelva con un string que indique si el usuario tiene o no derecho al subsidio, tal string deberá ser imprimido en el .then de la promesa con console.log. En caso de que la variable contenga un valor nó válido la promesa se debe rechazar lanzando el siguiente error: new Error("Estrato no válido"), tal error debe ser mostrado en el .catch de la promesa de la siguiente manera: console.log('Ha ocurrido un error: ', err.message), tenga en cuenta que err, es la información del error(motivo de rechazo) que se pasa desde el reject al catch como parámetro de nombre err.
- Cree una promesa que implemente 4 métodos .then en cadena de tal manera que la promesa se resuelva inicialmente con el valor 2 <resolve(2)> y los siguientes métodos .then eleven al cuadrado el valor pasado, imprimiendo al final el mensaje “el valor final es: X” donde x es el resultado final del encadenamiento
- Cree tres promesas promesa1, promesa2 y promesa3, de tal manera que la promesa1 siempre se resuelva con la cadena “Somos ADSI”, que la promesa2 tenga dos opciones, resolverse o rechazarse; de tal forma que si se resuelva lo haga con la cadena “ Somos programadores” , pero si se rechaza lo haga lanzando un error “Promesa 2 no cumplida” (use una variable y un condicional doble para controlar la resolución o rechazo de la promesa, p.e. una variable estado, velocidad, edad etc). Por último, la promesa3 siempre se debe resolver con la cadena “, Hacemos mover el mundo”. Encadene las promesas de tal manera que cuando la promesa1 se cumpla, imprima desde su .then el mensaje con el que se resolvió y retorne la promesa2 y cuando ésta última se cumpla, imprima desde su .then el mensaje con el que se resolvió y retorne la promesa3, y cuando la promesa 3 se cumpla imprima desde su .then el mensaje con el que se resolvió. Implemente .catch para manejar el error que puede ocurrir en la promesa2, en caso de que ocurra el rechazo, imprima el mensaje del error mediante err.message.

Contacto: gangaritah@gmail.com

- Cree una promesa cuyo resolve y reject dependan cada uno de una llamada asíncrona usando setTimeout y así, su rechazo o resolución dependerá del proceso asíncrono que termine primero. Implemente métodos .then y .catch. Si la promesa se rechaza, lance el error con new Error('info error') en el reject e imprima el stack del error en .catch, si se cumple, resuelva la promesa con la cadena "promesa resuelta" e imprímalo en el .then
- Cree cuatro promesas donde cada una para resolverse dependa de un setTimeout, de tal manera que cada promesa se resuelva en tiempos diferentes. Ejecute las cuatro promesas de forma paralela, e imprima los cuatro resultados de resolución(recuerde que estos resultados van en un arreglo, se sugiere use .foreach). No olvide .catch, qué pasa si una de las promesas falla al estar las cuatro en paralelo ?. Haga una implementación de Promise.race con dos de las promesas anteriores.