FCyT FACULTAD DE CIENCIA Y TECNOLOGÍA UADER
SEDE CONCEPCIÓN DEL URUGUAY

Licenciatura en Sistemas de Información

Programación Avanzada

Manejo de callbacks

Los callbacks son funciones que se ejecutan cuando sucede algo.
En el ejemplo podemos observar la siguiente función.

otra forma de codear una función es con la nomenclatura ES6, conocida como las Arrow Function, o función de flecha.

El siguiente ejemplo muestra un callback que nos regresa un resultado, analice las funciones:

```
let getUsuarioById = (id, callback) => {
    let usuario = { // suponemos que traemos de una BBDD
      nombre: "Ernesto",
      id
    }

    if (id === 20) {
      callback(`El usuario con id ${id}, no existe en la Base de datos`);
    }else {
      callback(null, usuario);
    }

}

getUsuarioById(20, (err, usuario)=> { // llamamos a

  if (err) {
    return console.log(err);
  }

   console.log('Usuario de base de datos', usuario);
});

```

1- Ejecute con nodemon las funciones, que sucede cuando modificamos el id?
2- En otro archivo ***callback2.js***, creamos una función para obtener un empleado en base al modelo anterior con el callback:

```
// Simulemos una base de datos -->
//BBDD empleados
let empleados = [{
  id:1,
  nombre: 'Ernesto'
},{
  id:2,
  nombre: "Marcelo"
},{
  id:3,
  nombre:'Pedro'
}];

//BBDD Salarios
let salarios = [{
  id:1,
  salario: 1000
},{
  id:2,
  salario:2000
}];

// Creamos una funcion para obtener un empleado por id //
```

```
// Creamos una funcion para obtener un empleado por id //

let getEmpleado = (id, callback) => {

  let empleadoDB = empleados.find(empleado => empleado.id === id) // barre toda la funcion sobre los id

    if (!empleadoDB) {
      callback(`No existe un empleado con el id ${id} `)

    }else {
      callback(null,empleadoDB);
    }

}

getEmpleado(4,(err, empleado)=>{

  if (err) {
    return console.log(err);
  }
  console.log(empleado);
});
```

3- Agregue al código de nuestro programa para obtener los salarios de ese empleado:

```
//Obtenemos el salario //

let getSalario = (id, callback) => {

  let salarioDB = salarios.find(salario => salario.id === id)
    if (!salarioDB) {
```

Recuerde de manejar el error con un callback, en el caso de que no exista un salario con el id (tanto).

Promesas:

Nos permiten ejecutar un trabajo, ya sea asíncrono o síncrono y después de ejecutar la tarea, realizar algún trabajo en particular.
Veamos el siguiente ejemplo:

```
let empleados = [{
  id:1,
  nombre: 'Tito',
},{
  id:2,
  nombre: 'Pedro'
},{
  id:3,
  nombre: 'Juan'
}];

let salarios = [{
  id: 1,
  salario: 3000
},{
  id: 2,
  salario:4000
}];

let getEmpleado = (id) => {

  return new Promise((resolve, reject) =>{ // LA promesa es una funcion, que tiene dos callback, resolve y reject
    //El "resolve", se llama si la promesaa es exitosa, si logra encontrar un empleado
    // El "reject" se llama si no es exitosa no existe un empleado en la BBDD
    let empleadoDB = empleados.find(empleado => empleado.id === id)

    if (!empleadoDB) {
      reject(`No existe  el empleado con el ID ${ id }`)
    } else {
      resolve(empleadoDB)
    }
```

1- Pruebe el siguiente código en Node.
2- Genere una nueva Promise para obtener un getSalario() del empleado

Async-Away

Otra forma de manejar los callback, y de manera mas optima es con Async-Away,

Copie le siguiente código siguiendo la estructura del modelo, puede leer mas sobre Async-Away:
https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Sentencias/funcion_asincrona

https://javascript.info/async-await

```
let empleados = [{
  id:1,
  nombre: 'Tito',
},{
  id:2,
  nombre: 'Pedro'
},{
  id:3,
  nombre: 'Juan'
}];

let salarios = [{
  id: 1,
  salario: 3000
},{
  id: 2,
  salario:4000
}];

let getEmpleado = async (id) => {

    let empleadoDB = empleados.find(empleado => empleado.id === id)

    if (!empleadoDB) {
      throw new Error(`No existe  el empleado con el ID ${ id }`)
    } else {
      return empleadoDB;
    }

}
```

1- Genere el manejo del async para que me devuelva un salario ingresado por id, también deberá manejar el error, en caso de que no se encuentre el id dentor de la base de datos local.