Desarrollo de Aplicaciones Telemáticas, examen práctico, 22/05/2025

Grado en Ingeniería en Tecnologías de la Telecomunicación
Grado en Ingeniería en Sistemas de la Telecomunicación
Universidad Rey Juan Carlos

Instrucciones

1. Emplea lo visto en clase, y solo lo visto en clase (que puedes consultar en las transparencias). No será válido ni JavaScript más antiguo ni más avanzado.

Ejercicio 1. 4.0 puntos

La práctica de auto-pedido que has hecho era, intencionadamente, muy simple. Una de las primeras cosas que habría que mejorar es que la gestión de los productos esté basada en identificadores. Por ejemplo algo como esto[1]

```
let productos = [
  ["0001", "Café con leche", "Coffee with milk", "Café au lait", "Kaffee
     mit Milch"],
  ["0002", "Café solo", "Black coffee", "Café noir", "Schwarzer Kaffee"],
  ["0003", "Té", "Tea", "Thé", "Tee"],
  ["0004", "Vino de Rioja", "Rioja wine", "Vin de Rioja", "Rioja-Wein"],
  ["0005", "Vino de Ribera del Duero", "Ribera del Duero wine",
    "Vin de Ribera del Duero", "Ribera del Duero-Wein"],
];
```

Ahora harás un programa muy parecido a tu práctica *duplicados.js*, pero con esta estructura. Puedes aprovechar el código que escribiste, con tal de que sea JavaScript como el visto en clase, ni más antiguo ni más avanzado. Escribe un programa en JavaScript contemporáneo en el fichero `pedidos.js`. Tendrá una función que recibirá:

1. Una lista de sublistas. Todos los elementos de las sublistas serán cadenas. El primer elemento de cada sublista será un código de producto. El segundo, tercero, cuarto y quinto contendrá, respectivamente, el nombre del producto en español, inglés, francés y alemán. Como el ejemplo anterior.

2. Un número entero que representa el idioma que debe usarse en la salida. 1 para español, 2 inglés, 3 francés, 4 alemán.

3. Una lista no vacía de elementos, donde cada elemento será una clave de producto pedido por el cliente. Ejemplo:

   ```
   ["0002", "0004", "0001", "0004", "0001", "0004"]
   ```

La función devolverá una lista de sublistas, donde

- Habrá una sublista por cada elemento del argumento 1, teniendo en cuenta que si hay elementos repetidos, solo se generará una sublista.

- El primer elemento de cada sublista devuelta será un número sin decimales, que indicará cuántas veces aparecía el elemento en la lista de entrada.

[1] Sería más adecuado un plain object, pero usaremos una lista de listas, que es una estructura que has usado más en las prácticas.

- El segundo elemento de cada sublista devuelta será el nombre de producto, en el idioma especificado.

Ejemplo de lista devuelta:

```
[
  [1, "Black coffee"],
  [3, "Rioja wine"],
  [2, "Coffee with milk"]
]
```

Observaciones:

- La función llamará a otras funciones, si procede para estructurar razonablemene el código.
- El primer argumento de tu función, la lista de productos, no es necesario que lo revises. Puedes suponer que ya ha sido validado y que todo está bien.
- Lo que sí tienes que comprobar es que la función que escribes realmente recibe tres argumentos, que el segundo es un entero entre 1 y 4 y que el tercero es una lista de cadenas. Si no es el caso, el programa levantará una excepción.

Solución

```
'use strict'

let productos = [
  ["0001", "Café con leche", "Coffee with milk", "Café au lait",
    "Kaffee mit Milch"],
  ["0002", "Café solo", "Black coffee", "Café noir", "Schwarzer Kaffee"],
  ["0003", "Té", "Tea", "Thé", "Tee"],
  ["0004", "Vino de Rioja", "Rioja wine", "Vin de Rioja", "Rioja-Wein"],
  ["0005", "Vino de Ribera del Duero", "Ribera del Duero wine",
    "Vin de Ribera del Duero", "Ribera del Duero-Wein"],
  ["0006", "Cerveza", "Beer", "Bière", "Bier"],
  ["0007", "Agua mineral", "Mineral water", "Eau minérale", "Mineralwasser"
     ],
  ["0008", "Zumo de naranja", "Orange juice", "Jus d'orange", "Orangensaft"
     ],
  ["0009", "Tarta de queso", "Cheesecake", "Gâteau au fromage", "Käsekuchen
     "],
  ["0010", "Tarta de chocolate", "Chocolate cake", "Gâteau au chocolat",
    "Schokoladenkuchen"]
];


function cuenta_elementos(elemento,lista){
    // Cuenta cuántas veces aparece el elemento en la lista
    let contador = 0;
    for (let l of lista){
        if (l === elemento)
            contador += 1;
    }
    return contador;
}
```

```
function elimina_repes(lista){
    // Devuelve copia de la lista, sin repetidos
    let rval = [] ;  // Lista a devolver
    for (let l of lista){
        if (cuenta_elementos(l,rval) == 0) // Si el elemento no está
            rval.push(l);                  // lo añade
        }
    return rval;
}


function devuelve_valor(productos, idioma, clave){
    // Devuelve el valor de la clave, en el idioma pedido
     let rval = null;  // Valor a devolver
     for (let producto of productos){
         if (clave === producto[0])
             rval = producto[idioma];
             //console.log("encontrado ",producto[idioma]);
     }
     return(rval);
}


function comprueba_lista_cadenas(lista_cadenas) {
    if (!Array.isArray(lista_cadenas)) {
        throw new TypeError("El argumento debería ser una lista.");
    }
    for (let l of lista_cadenas) {
        if (typeof l !== "string") {
            throw new TypeError("Todos los elementos de la lista deberían
               ser cadenas.");
        }
    }
}


function comprueba_idioma(idioma){
    if (typeof(idioma) !== 'number')
        throw new TypeError(
            'El segundo argumento, idioma, debería ser un número.');

    if (idioma <1 || idioma >4)
        throw new RangeError(
            'El idioma debería ser un número entre 1 y 4');
}


function genera_lista_salida(productos, idioma, pedidos){
    if (arguments.length !== 3){
        throw new SyntaxError(
            'La función genera_lista_salida debería recibir 3 argumetos.');
    }

    comprueba_idioma(idioma);
```

```
    comprueba_lista_cadenas(pedidos)

    let claves = elimina_repes(pedidos);
    let rval = []; // valor a devolver
    for (let clave of claves){
        let unidades = cuenta_elementos(clave, pedidos);
        let nombre_producto = devuelve_valor(productos, idioma, clave);
        let sublista = [ unidades, nombre_producto];
        rval.push(sublista);

    }
    return rval;
}


let pedido = ["0002", "0004", "0001", "0004", "0001", "0004"];
let idioma = 2; // Inglés

console.log(genera_lista_salida(productos, idioma, pedido))
```

Ejercicio 2. (6.0 puntos)

Modifica tu aplicación de auto-pedido según la siguiente especificación:

- La aplicación ya no será de auto-pedido, sino para que un camarero tome nota. Por tanto, será necesario añadir el concepto *número de mesa*.
- En la aplicación aparecerá una nueva tabla, la *tabla de mesas*. Habrá una fila por cada mesa, de forma que el camarero pueda indicar a qué mesa corresponde el pedido. El número de mesas vendrá determinado por una variable global al comienzo de tu programa.
- Es imprescindible que el camarero empiece por indicar la mesa. Por tanto, inicialmente no se verán las tablas de productos ni de pedidos. Solo serán visibles cuando el camarero indique mesa. En ese momento, en alguna parte del interface se verá cláramente a qué mesa se corresponde el pedido, y ya no se verá la tabla de mesas.
- En algún lugar bien visible habrá un botón *Finalizar pedido*. Solo se verá cuando el camarero esté introduciendo el pedido. Cuando lo pulse, el sistema volverá al estado inicial: se verá la tabla de mesas y no se verán las tablas de productos y pedidos.