Fundamentos de JavaScript II

Tareas a realizar

Lectura

Lee atentamente los siguientes artículos y sus correspondientes subsecciones en caso de que las tengan:

- https://es.javascript.info/array
- https://es.javascript.info/array-methods
- https://es.javascript.info/iterable
- https://es.javascript.info/keys-values-entries
- https://es.javascript.info/destructuring-assignment
- https://es.javascript.info/date
- https://es.javascript.info/json
- https://es.javascript.info/advanced-functions (Bastante denso, pero muy interesante para conocer los entresijos del lenguaje. Las secciones "Decoradores y redirecciones, call/apply" y "Función bind: vinculación de funciones" tienen cierta complejidad: no hace falta que se comprenda todo, simplemente tener una idea muy general)

Fichero de la aplicación

Utilizaremos el fichero llamado `gestionPresupuesto.js` en la carpeta `js` del repositorio. A no ser que se indique lo contrario, todo el código que se cree deberá guardarse en este fichero.

Variables globales

Añade las siguientes variables globales:

- `gastos` - Almacenará el listado de gastos que vaya introduciendo el usuario. Inicialmente contendrá un array vacío.
- `idGasto` - Se utilizará para almacenar el **identificador actual** de cada gasto que se vaya añadiendo. Su **valor inicial** será **0**. Se irá incrementando con cada gasto que se añada.

```
// TODO: Variable global
let presupuesto = 0;
let gastos = [];
let idGasto = 0;
```

Modificación de `export`

Añade las
funciones `listarGastos`, `anyadirGasto`, `borrarGasto`, `calcularTotalGastos` y `calcular Balance` al objeto `export` del final del fichero.
Define las funciones vacías (sin parámetros y sin cuerpo) en el
fichero `gestionPresupuesto.js` para que los tests no den error de sintaxis y se puedan ir comprobando conforme se vaya avanzando en la práctica.

```
export {
    mostrarPresupuesto,
    actualizarPresupuesto,
    CrearGasto,
    listarGastos,
    anyadirGasto,
    borrarGasto,
    calcularTotalGastos,
    calcularBalance,
}
```

Funciones

Función `listarGastos`
Función **sin parámetros** que devolverá la variable global `gastos`.

```
//Función listarGastos
function listarGastos(){
    return gastos;
}
```

```
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git add .\gestionPresupuesto.js
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git commit -m "Función listar gastos"
[master a72b9d6] Función listar gastos
 1 file changed, 13 insertions(+), 1 deletion(-)
```

Función `CrearGasto`

Actualiza la función constructora para que incluya la **fecha y las etiquetas** (ver apartado de Objeto `gasto`). Los parámetros adicionales de la función deben ir a continuación de los existentes.

- Si no se indican los parámetros de `etiquetas`, se almacenará en la propiedad `etiquetas` un **array vacío**.

```
this.etiquetas = [];
```

- Si no se indica el parámetro `fecha`, se almacenará en la propiedad `fecha` la **fecha actual**.
- El parámetro `fecha` deberá ser un `string` con formato válido que pueda entender la función `Date.parse`. Si la fecha no es válida (no sigue el formato indicado), se deberá almacenar la **fecha actual** en su lugar.
- Tal como se indica en la sección de objeto `gasto`, la fecha se almacenará en formato `timestamp`.

```
if (fecha){
    fecha = Date.parse(fecha);
}
else {
    fecha = Date.now();
}
this.fecha=fecha;
```

- Las etiquetas se pasarán como una **lista de parámetros de número indeterminado**.

```
function CrearGasto( descripcion, valor, fecha, ...etiquetas )
    //Función constructora
```

- Para añadir las etiquetas se utilizará el método `anyadirEtiquetas` explicado en la sección de objeto `gasto`.

```
//Método anyadirEtiquetas
this.anyadirEtiquetas = function (...etiquetas){
    let pos = -1;

    for (let e of etiquetas){

        pos = this.etiquetas.indexOf(e);

        if (pos != -1){
            this.etiquetas.push (pos, 1);
        }
    }
}
this.anyadirEtiquetas (...etiquetas);
```

Algunos ejemplos de llamadas de función `CrearGasto` podrían ser:

```
let gasto1 = new CrearGasto("Gasto 1");
let gasto2 = new CrearGasto("Gasto 2", 23.55);
let gasto3 = new CrearGasto("Gasto 3", 23.55, "2021-10-06T13:10" );
let gasto4 = new CrearGasto("Gasto 4", 23.55, "2021-10-06T13:10", "casa" );
let gasto5 = new CrearGasto("Gasto 5", 23.55, "2021-10-06T13:10", "casa",
"supermercado" );
let gasto6 = new CrearGasto("Gasto 6", 23.55, "2021-10-06T13:10", "casa",
"supermercado", "comida" );
```

Función `anyadirGasto`

Función de **1 parámetro** que realizará tres tareas:

- Añadir al objeto gasto pasado como parámetro una propiedad `id` cuyo valor será el valor actual de la variable global `idGasto`.
- **Incrementar** el valor de la variable global `idGasto`.
- **Añadir** el objeto gasto pasado como parámetro a la variable global `gastos`. El gasto se debe añadir **al final del array**.

```
//Función anyadirGasto
function anyadirGasto (gasto){
    gasto.id = idGasto;

    idGasto++

    gastos.push(gasto);
}
```

```
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git add .\ges
tionPresupuesto.js
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git commit -m
 "Función añadir gasto"
[master 4b587f1] Función añadir gasto
 1 file changed, 10 insertions(+), 1 deletion(-)
```

Función `borrarGasto`

Función de **1 parámetro** que **eliminará** de la variable global `gastos` el objeto gasto cuyo `id` haya sido pasado como parámetro. Si no existe un gasto con el `id` proporcionado, no hará nada.

```
//Función borrarGasto
function borrarGasto(id){
    let pos = gastos.findIndex(gasto => gasto.id === id);

    if(pos !=-1){
        gastos.splice(pos,1);
    }
}
```

```
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git add .\gestionPresupuesto.js
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git commit -m "Función borrar gasto"
[master 65d43d2] Función borrar gasto
 1 file changed, 9 insertions(+)
```

Función `calcularTotalGastos`

Función **sin parámetros** que devuelva la suma de todos los gastos creados en la variable global `gastos`. De momento no los agruparemos por período temporal (lo haremos en sucesivas prácticas).

```
//Función calcularTotalGastos
function calcularTotalGastos (){
    let total = 0;

    for (let g of gastos){
        total += g.valor;
    }
    return total;
}
```

```
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git add .\gestionPresupuesto.js
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git commit -m "Función calcular el total de los gastos"
[master 839119e] Función calcular el total de los gastos
 1 file changed, 10 insertions(+)
```

Función `calcularBalance`

Función **sin paràmetros** que devuelva el balance (presupuesto - gastos totales) disponible. De momento no lo obtendremos por período temporal (lo haremos en sucesivas prácticas). Puede utilizar a su vez la función `calcularTotalGastos`.

```
//Función calcularBalance
function calcularBalance (){
    return presupuesto - calcularTotalGastos();
}
```

```
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git add .\gestionPresupuesto.js
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git commit -m "Función calcular balance"
[master 668ec6b] Función calcular balance
 1 file changed, 5 insertions(+)
```

Objeto `gasto`

Propiedades

Añade las siguientes propiedades al objeto `gasto`:

- `fecha` - Almacenará la fecha en que se crea el gasto en forma de **timestamp** (ver https://es.javascript.info/date#creacion).

```
if (fecha){
    fecha = Date.parse(fecha);
}
else {
    fecha = Date.now();
}
this.fecha=fecha;
```

- `etiquetas` - Almacenará en un array el listado de etiquetas (categorías) asociadas al gasto.

```
this.etiquetas=[];
```

Métodos

Añade o modifica los siguientes métodos del objeto `gasto`:

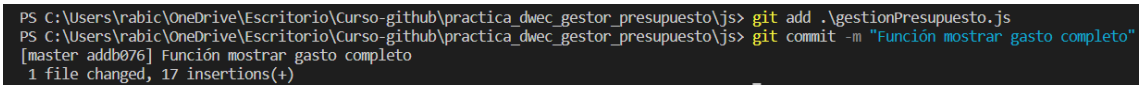
- `mostrarGastoCompleto` - Función sin parámetros que **devuelva** el texto multilínea siguiente (ejemplo para un gasto con tres etiquetas):
- `Gasto correspondiente a DESCRIPCION con valor VALOR €.`
- `Fecha: FECHA_EN_FORMATO_LOCALIZADO`
- `Etiquetas:`
- `- ETIQUETA 1`
- `- ETIQUETA 2`
- `- ETIQUETA 3`

```
//Método mostrarGastoCompleto
this.mostrarGastoCompleto = function (){
    let txt = "";
    let fecha = new Date (this.fecha)
    let fechaLocal = fecha.toLocaleString (fecha);

    txt += `Gasto correspondiente a ${this.descripcion} con valor ${valor} €\n
    Fecha: ${fechaLocal}\n
    Etiquetas:\n`;

    for (let e of this.etiquetas){
        txt += (`- ${e}\n`);
    }

    return txt;
}
```



```
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git add .\gestionPresupuesto.js
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git commit -m "Función mostrar gasto completo"
[master addb076] Función mostrar gasto completo
 1 file changed, 17 insertions(+)
```

Para mostrar la fecha en formato localizado puedes utilizar el método `toLocaleString()` (referencia de toLocaleString()).

- `actualizarFecha` - Función de **1 parámetro** que actualizará la propiedad `fecha` del objeto. Deberá recibir la fecha en formato `string` que sea

entendible por la función `Date.parse`. Si la fecha no es válida, **se dejará sin modificar**.

```
//Método actualizarFecha
this.actualizarFecha = function (fecha){
    fecha = Date.parse (fecha);

    if (fecha){
        this.fecha = fecha;
    }
```

```
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git add .\gestionPresupuesto.js
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git commit -m "Método actualizar fecha"
[master 93e7c1a] Método actualizar fecha
 1 file changed, 9 insertions(+)
```

- `anyadirEtiquetas` - Función de un **número indeterminado de parámetros** que añadirá las etiquetas pasadas como parámetro a la propiedad `etiquetas` del objeto. **Deberá comprobar que no se creen duplicados**.

```
//Método anyadirEtiquetas
this.anyadirEtiquetas = function (...etiquetas){
    let pos = -1;

    for (let e of etiquetas){

        pos = this.etiquetas.indexOf(e);

        if (pos != -1){
            this.etiquetas.push (pos, 1);
        }
    }
}
this.anyadirEtiquetas (...etiquetas);
```

- `borrarEtiquetas` - Función de **un número indeterminado de parámetros** que recibirá uno o varios nombres de etiquetas y procederá a eliminarlas (si existen) de la propiedad `etiquetas` del objeto.

```
//Método borrarEtiquetas
this.borrarEtiquetas = function (...etiquetas){
    let pos = -1;

    for (let e of etiquetas){

        pos = this.etiquetas.indexOf(e);

        if (pos != -1){
            this.etiquetas.splice (pos, 1);
        }
    }
}
```

```
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git add .\ges
tionPresupuesto.js
PS C:\Users\rabic\OneDrive\Escritorio\Curso-github\practica_dwec_gestor_presupuesto\js> git commit -m
 "Método borrar etiquetas"
[master 7a36dc4] Método borrar etiquetas
 1 file changed, 14 insertions(+)
```

```
> mocha test/02_testIntroJSII.js


  Función listarGastos e inicialización de la variable global gastos
    ✓ Devuelve un array vacío si no hay gastos (valor por defecto)

  Función CrearGasto y funcionamiento de objeto gasto
    1) CrearGasto: comprobación con fecha y etiquetas
    2) Método 'mostrarGastoCompleto' del objeto gasto
    ✓ Método 'actualizarFecha' del objeto gasto
    3) Método 'anyadirEtiquetas' del objeto gasto
    4) Método 'borrarEtiquetas' del objeto gasto

  Función anyadirGasto
    ✓ Añade el gasto que se pasa como parámetro a la variable global 'gastos'

  Función borrarGasto
    ✓ Elimina de la variable global 'gastos' el gasto cuyo id se pasa como parámetro

  Función calcularTotalGastos
    ✓ Calcula la suma de todos los gastos presentes en la variable global 'gastos'

  Función calcularBalance
    ✓ Calcula el balance (presupuesto - gastos totales) a partir de los gastos almacenados en la variable global 'gastos'


  6 passing (48ms)
  4 failing
```