Funciones

Crear un archivo HTML y vincular el mismo con un archivo **app.js**, dentro del mismo crear:

1. un objeto literal **estudiante** con las siguientes propiedades y sus valores:
    a. nombre (String)
    b. curso (String)
    c. dni (Number)
    d. email (String)
    e. Crear una función **fromObjectToArray** que reciba un parámetro, será un objeto literal, y que retorne los valores de cada una de las propiedades de ese objeto en un Array.
    f. Mostrar en consola el Array que retorna esa función.

2. Crear una función **cambiarColorDeFondoDelBody**. Que reciba como parámetro un String (nombre del color ó valor hexadecimal) y que cambie el color de fondo de la etiqueta `<body>`. El cambio sólo deberá realizarse, si el valor pasado como parámetro es diferente a *green* ó *#0f0* ó *#00ff00*. Si el cambio de color es posible, la función retornará `true`. De lo contrario retornará `false`.
    a. Ejecutar la función y pasarle como parámetros diferentes valores.
    b. Mostrar en consola si el cambio de color fue posible.

3. Dentro del documento HTML, crear al menos 10 párrafos (`<p>`) con texto aleatorio.
    a. Dentro de **app.js** capturar todos los párrafos del documento.
    b. Hacer una función que, tomando como parámetro a los párrafos capturados previamente. Recorra los mismos y:
        i. Les cambie el color de tipografía a **rojo**.
        ii. Genere que la tipografía esté en **negrita**.
        iii. Defina que el texto esté con alineación al **centro**.
    c. El proceso anterior deberá ejecutarse SOLAMENTE para los párrafos con número par.
    d. La función deberá retornar la cantidad de elementos `<p>` que no fueron afectados con estos cambios de estilo.
    e. Mostrar en consola: `Párrafos que no se vieron afectados: N`.

Métodos de Array

Dentro del archivo **app.js**:

1. Vamos a jugar al detective. Hemos recibido un código anónimo cuyo mensaje queremos descifrar. las únicas pistas que tenemos para tal fin son: `filter()` y `typeof`. El código es:

```
var enigma = ["l", 1, "a", 2, 2, 5, "p", 5, 7, 5, 3, "e", 6, "r", 7, 6, 5, 3, 2, 1, "s", 9, 9, 9, 6, "e", 2, "v", 5, "e", 3, "r", 2, "a", 1, 6, 4, 1, 2, "n", 2, "c", 3, 5, 5, 5, 7, "i", 4, "a", 5, 2, 1, 3, "e", 6, "s", 7, "l", 4, "a", 3, "c", 2, 3, 1, 5, 3, 2, "l", 3, "a", 4, "v", 5, "e", 6];
```

   a. Con toda esta información nos piden encontrar la altura de una calle. Para descifrar la misma nos dieron la siguiente pista: *"Si descifrar la altura quieres, sumar todos los números debes"*. ¿Qué método de Array podemos usar para ésto?.

   b. Cómo último paso, nos piden encontrar el nombre de la calle. Para ello tenemos la siguiente pista: *"Si el nombre de la calle necesitás, omitir todos los números deberás".*

   c. Finalmente debemos mostrar en consola el nombre de la calle junto con su altura.

D.O.M - 1

1. Crear un archivo **gastos.html** y vincularlo con archivo **gastos.js.** Importante, tener precaución de generar todo nuestro código JS dentro del `window.onload`.

2. Usando los métodos **prompt(), confirm()** y **alert()** e incluso cosas que hemos visto en clases pasadas. Vamos a generar un reporte de gastos diarios de una familia. Dicho reporte funciona de la siguiente manera: Pide la cantidad de integrantes de la familia, para cada uno de ellos pide nombre y luego la cifra que ésta persona gastó. Al final veremos cada nombre con su

correspondiente cifra, la persona que más gastó y la persona que menos gastó. Y el total de gastos de toda la familia.

a. Lo primero que haremos es preguntarle al visitante recién ingresa al HTML si quiere iniciar. Si la respuesta es negativa, deberá mostrarse una alerta que diga *"Gracias por haber venido"* y luego será direccionado al sitio web de Netflix.
b. Ahora, si la respuesta es positiva, vamos a iniciar nuestro proceso, lo primero que haremos es pedirle al visitante la cantidad de integrantes de la familia. Validar que el dato ingresado sea un número y que no sea inferior a 3, de NO ser un número deberá alertar que el valor necesario es un número y volverá a preguntar por la cantidad de integrantes. Para este proceso, puede ser de utilidad la siguientes función nativa de JS `isNaN()`. Puedes buscar qué hace la misma.
c. Una vez con la cantidad de integrantes listos, vamos a pedir para cada uno: nombre y gastos del día. Tener en cuenta de validar que:
    i. El nombre no puede estar vacío.
    ii. Los gastos no puede ser un texto ni estar vacío.

    En cualquiera de esos casos, alertar del error y volver a pedir el dato. Al final se deberá generar un Array de Objetos Literales cada uno con la propiedad **nombre** y **valor**. Ejemplo:

```
integrantes[
      {nombre: "Ada", gastos: 300},
      {nombre: "Tim", gastos: 570},
      {nombre: "Vincent", gastos: 80},
]
```

d. Después de esto haremos mostraremos en consola dicho array.

D.O.M - 2

Seguidamente y teniendo en cuenta toda la funcionalidad existente, queremos mostrar los resultados dentro del contenido del archivo **gastos.html** . **ATENCIÓN**: todo lo siguiente deberá ser generado desde javascript, nuestro archivo html no debe tener NADA dentro del `<body>` excepto la vinculación del `<script>`.

1. Generar un título con el texto *"Reporte de gastos familiares"*.

2. Generar una lista dentro de la cual se muestre:
    a. El nombre del integrante que más gastó junto con su gasto.

   b. El nombre del integrante que menos gastó junto con su gasto.
   c. Los gastos de toda la familia.
   d. El promedio de gastos del día.

   Todos los elementos de lista, deberán tener el atributo `title` con el mismo texto que muestra cada uno.

3. Generar un `<button>` con el texto "¿Nos pasamos del presupuesto?" el cual, al ser clicado, muestre un texto llamativo diciendo si el presupuesto diario fue superado, teniendo en cuenta que para toda la familia el presupuesto máximo diario es de **$1200**.

4. Agregar un mensaje de ayuda oculto que solamente se muestre al pasar por encima del botón generado en el ejercicio anterior. Al quitar el mouse del botón el mensaje de ayuda debe desaparecer

5. Agregar funcionalidad para que el mensaje de ayuda solo desaparezca 3 segundos posterior a que el mouse se aleje del botón

6. Generar otro `<button>` que al ser clicado, asigne si no existe y elimine si existe, la clase **dark-theme** al `<body>`. Crear un **css** que defina todo el set de estilos para dicha clase.

7. Agregar una funcionalidad "secreta" que al presionar la tecla "q" se dispare una alerta que diga "Oh, encontraste el secreto!"