	I.E.S. HERMANOS MACHADO **2º CFGS DAW** **Desarrollo Web en Entorno Cliente** **Unidad 4: Boletín 1**	**DEPARTAMENTO DE INFORMÁTICA**

En la carpeta compartida de ejemplos de esta unidad tenéis disponible las plantillas de todos los ejercicios del boletín.

1. Convierte la siguiente función tradicional a una Arrow Function. Simplificar al máximo (elimina llaves y return si es posible).

```
function saludar(nombre) {
    return "Hola " + nombre;
}
```

2. Crea una función flecha llamada `ToogleButton` que reciba un texto y retorne un objeto cualquiera creado por ti simulando un elemento HTML.

3. Dadas dos variables marca y modelo, crea una frase que se devuelva por consola que diga "El coche es un [marca] [modelo] del año [año actual]".

4. Crea una función que reciba un objeto usuario (con nombre e imagen) y devuelva por consola un string que simule una tarjeta HTML usando Template Literals. Ejemplo de tarjeta HTML:

```
<div class="card">
      <img src="pepeGomez.jpg" alt="Pepe Gómez" />
      <h2>Pepe Gómez</h2>
</div>
```

5. A partir del objeto indicado más abajo, crea una función **`mostrarPerfil`** que lo reciba como parámetro. Usa desestructuración directamente en los argumentos de la función para extraer nombre, email y un valor por defecto para rol ("invitado").

```
const user = { nombre: "Pedro", email: "pedro@email.com" }
```

6. A partir del objeto indicado más abajo, crea un nuevo objeto **`nuevoEstado`** que sobrescriba **`loading`** a **`false`** y mantenga el resto de propiedades igual, sin modificar el objeto original.

```
const estado = { loading: true, error: null, data: [45,53,23] }
```

7. Crea una función **`sumar`** que pueda recibir cualquier cantidad de números como argumentos y devuelva la suma total. Para resolverlo tienes que usar rest parameter como argumento de la función y el método **`reduce`** del objeto predefinido Array.

8. En el fichero **`funciones.js`** crea y exporta las funciones suma y resta que reciben dos parámetros y devuelven cada una la operación correspondiente. En el fichero **`ej9.js`** debes importar ambas funciones y mostrar por consola la demostración de que funcionan correctamente.

9. En el fichero **`persona.js`** crea una clase Persona con tres atributos, nombre, apellidos y edad. Incluye los métodos consultores y modificadores de cada atributo y el método constructor de la misma. Expórtala por defecto. En el fichero **`ej8.js`** debes importar esa clase pero llamándola Usuario. Prueba a instanciar un objeto con este nuevo nombre y a visualizarlo por consola.

10. A partir del array de objetos indicado más abajo, genera un nuevo array de objetos cadena que contenga: "Producto: [nombre] - [precio]€". A continuación, a partir del array original genera un nuevo array solo con los productos de precio mayor a 30 y devuelve el objeto cuyo nombre sea "Ratón". Muestra por consola el resultado de estas tres operaciones.

```
const productos = [
    { id: 1, nombre: "Ratón", precio: 20 },
    { id: 2, nombre: "Teclado", precio: 50 }
];
```

11. A partir del documento HTML proporcionado para este ejercicio, implementar las funciones javascript necesarias para iniciar una promesa y atenderla si se pulsa el botón “Procesar Promesa” antes de 2 segundos. En el caso de no ser pulsado en ese plazo, la promesa no será atendida y se procesará un error. En el párrafo de salida se debe mostrar el mensaje que devuelva el objeto Promise, debiendo coger éste lo que el usuario haya indicado en el formulario de entrada para reflejar los mensajes de éxito y de error.

12. A partir del documento HTML proporcionado para este ejercicio, implementar las funciones javascript necesarias para iniciar una promesa y atenderla para que lea el texto incluido en el input y lo añada al párrafo de salida pasados dos segundos. Para este ejercicio no será necesario controlar el posible error del objeto Promise y se debe usar async/await.

13. A partir del documento HTML proporcionado para este ejercicio, implementar las funciones javascript necesarias para añadir a la capa de salida el contenido del

fichero de texto especificado en el formulario. El contenido debe añadirse sin producir ninguna recarga de página, para ello use fetch API.

14. A partir del documento HTML proporcionado para este ejercicio, implementar las funciones javascript necesarias para atacar la API Rest facilitada en el input del formulario y mostrar los objetos leídos por la consola.

15. A partir del documento HTML proporcionado para este ejercicio, implementar las funciones javascript necesarias para atacar la API Rest siguiente: https://picsum.photos/list . Ésta devuelve un objeto JSON con una lista de datos de las fotos que almacena. Construir una lista donde cada ítem tendrá un enlace a la URL de cada foto junto al nombre del autor de la misma.