FULL STACK DEVELOPER

Módulo 2 – ECMAScript 2015

Jorge Borrego Marqués

Prueba Evaluable

1. Crea una estructura de proyecto HTML y JavaScript.

Creamos un proyecto para la práctica en la carpeta que uso al respecto en el path:

C:\Users\jorge\Desktop\Master FullStack\proyectos\M2_ECMAScript2015

- Consola y comienzo del repositorio en local con el comando “git init”.
- Abro la carpeta en Visual Studio Code y la raíz de la carpeta archivo index.html.

Index.html

Html:5, código resultante cambiado :

```
<html lang="en">
<title>Ejercicio de Evaluación</title>
```

Creamos carpeta js donde creamos una archivo main.js que linkaremos posteriormente en el index con la etiqueta script y atributo src.

```
<script src="./js/main.js"></script>
```

**Código del index.html:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible"
content="IE=edge" />
    <meta name="viewport" content="width=device-
width, initial-scale=1.0" />
    <title>Ejercicio de Evaluación</title>
  </head>
  <body>
    <script src="./js/main.js"></script>
  </body>
</html>
```

2. En el archivo index.html, usa el siguiente código para crear un sencillo formulario.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible"
content="IE=edge" />
    <meta name="viewport" content="width=device-
width, initial-scale=1.0" />
    <title>Ejercicio de Evaluación</title>
  </head>
  <body>
    <input type="text" id="name" placeholder="Nombre"
/>
```

```
    <input type="text" id="surname"
placeholder="Apellidos" />
    <input type="number" id="points"
placeholder="Puntuación entre 0 y 12" />
    <button onclick="showUserResult()">Ver listado
por consola</button>
    <script src="./js/main.js"></script>
  </body>
</html>
```

3. Desarrolla una clase para alumnos que contenga las propiedades de nombre, apellidos y puntos.

Desarrollo de la clase Alumno(nombre, apellidos, puntos)

```
class Alumno {
    nombre;
    apellidos;
    puntos;

    constructor (nombre, apellidos, puntos){
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.puntos = puntos;
    }
}
```

En la clase Alumno declaramos las variables requeridas de: nombre, apellidos y puntos.

Seguidamente método constructor que recibe como parámetros nombre, apellidos y puntos.

En el constructor iniciamos las variables de clase indicando con la palabra reservada “this” que las variables toman el valor de los parámetros que recibe el métoso constructor.

4. En la clase anterior, añade un método “set” para establecer el valor de los puntos y otro método “get” que devuelva el string “Apto” si los puntos son iguales o superiores a 5 y “No apto” en caso contrario.

En la clase Alumno añadimos los métodos setPuntos() y getPuntos(), que nos permiten darle valor a la variable puntos y recibir su valor.

```
setPuntos(puntos) {
        this.puntos = puntos;
    }

    getPuntos(puntos) {
        if (puntos >= 5)
            return `Apto`;
        return `No apto`;
    }
```

En el método get recibimos como parámetro la variable del constructor para pasar por un bloque if-else que nos retorno un string indicando si es apto o no dependiendo de la condición >= 5.

5. . Declara una función que, pasados 2 segundos, devuelva una promesa que instancie a partir de los datos del formulario un objeto “alumn” y devuelva un mensaje con su nombre, sus apellidos y si ha resultado apto o no. Para obtener los valores del formulario, usa las siguientes líneas.

Función Promesa:

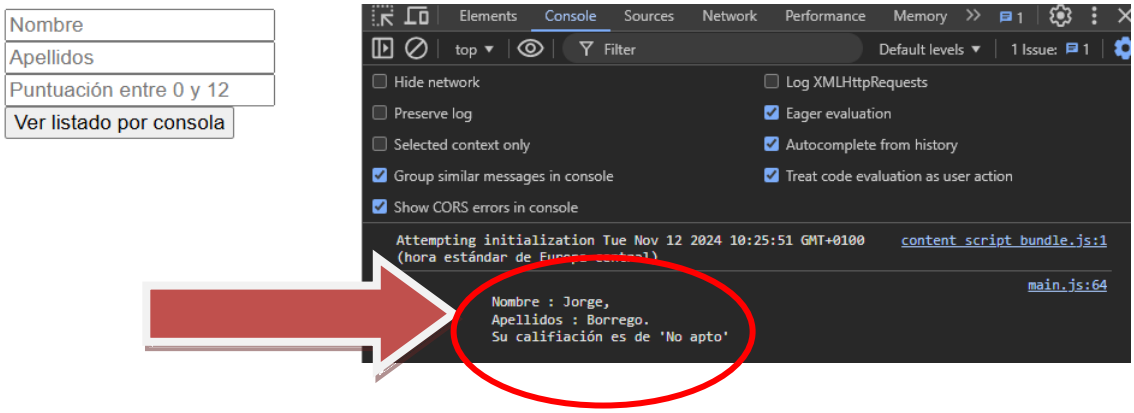
```
function getAlumno (name,surname,points) {
    return new Promise((resolve, reject) => {

        setTimeout(() => {
            resolve(new Alumno(name,surname,points))
        },2000)
    })
}
```

En la función promesa usamos unos de los métodos de la misma “resolve” que nos permite resolver con la instancia de un nuevo objeto de la clase Alumno después de un retardo de 2 segundos, que aplicamos dentro de un método “setInterval()”.

Para que podamos comprobar el correcto funcionamiento de la Promesa la invocamos con valores de ejemplo:

```
getAlumno('Jorge','Borrego',4) // Los parámetros
usados en la function son de Ejemplo
    .then( data => {
        console.log(`
            Nombre : ${data.nombre},
            Apellidos : ${data.apellidos}.
            Su califiación es de '${data.puntos}'
            `);
    })
    .catch(error =>
        console.error(error)
    );
```

Recibimos por Consola el siguiente resultado:



Por consola se muestran los valores de las propiedades del Objeto que pasamos como parámetros de Ejemplo a la hora de invocar a la función "getAlumno()".

En este caso:

Name: Jorge

Surname: Borrego Marqués

Points: 7

6. Declara la función que es invocada desde el botón index.html con el nombre “showUserResult()” de forma que implemente el patrón “async-await” e invoque la función anterior para recibir el mensaje de la promesa e imprimirlo por consola.

Lo primero recibimos los datos del formulario por medio de :

```
let name = document.getElementById("name").value;
let surname =
document.getElementById("surname").value;
let points = document.getElementById("points").value;
```

Aplicamos el async-await para “capturar” los datos recibidos del formulario y pasarlo como argumentos a la función Promise que se encarga de instanciar el nuevo Objeto de la clase Alumno.

```
async function showUserResult() {
        let alumn = await
getAlumno(name,surname,puntos);
        console.log(`
                Nombre : ${alumn.nombre},
                Apellidos : ${alumn.apellidos}.
                Su califiación es de
'${alumn.puntos}'
               `);
}
```

En el cuerpo de la función damos instrucciones síncronas de llamada a la función Promise anteriormente implantada.

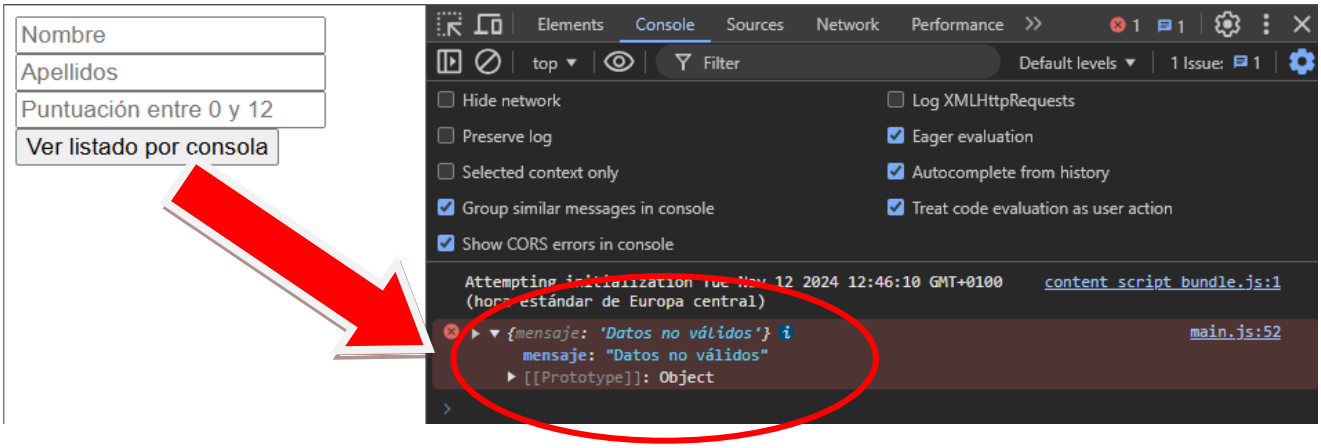
7. Implementa en la función que devuelve la promesa la gestión de errores para que, en caso de que el nombre, los apellidos o los puntos recibidos desde HTML sean strings vacíos, devuelva un mensaje de error con el valor "Datos no válidos".

```
function getAlumno (name,surname,points) {
    return new Promise((resolve, reject) => {
        if (
            (name === '') || (surname === '') ||
(points === '')
        ) {
            reject({mensaje: `Datos no válidos`})
        }
        setTimeout(() => {
            resolve(new Alumno(name,surname,points))
        },2000)
    })
}
```

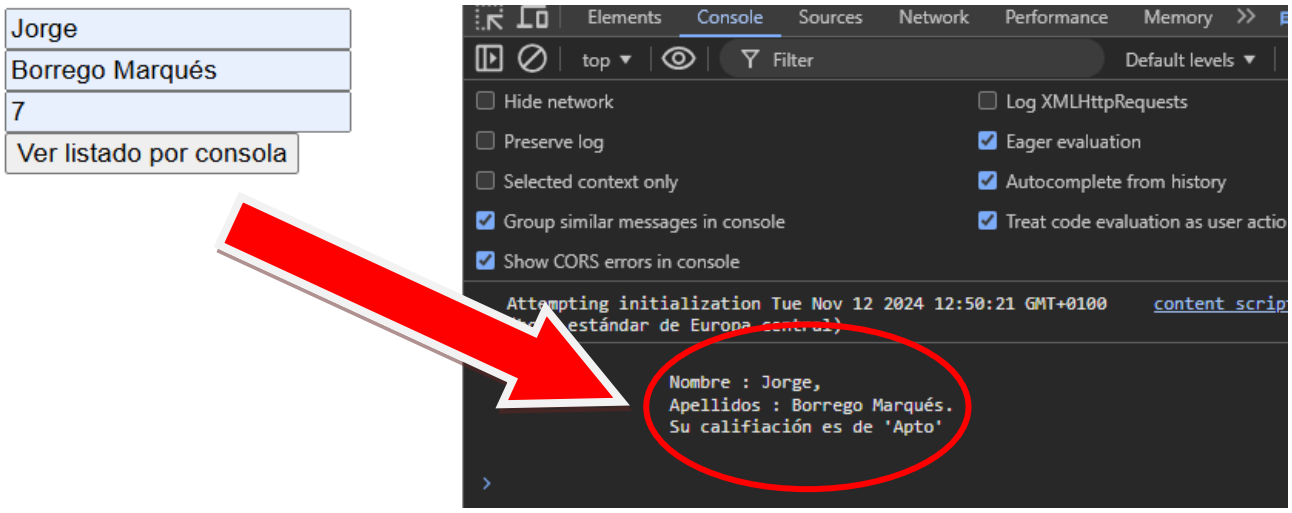
Usamos el método "reject" de la Promesa para validar los datos recibidos la función async-await por medio de un bloque if-else dónde resuelve si los campos no son vacíos y en caso contrario devuelve por consola un mensaje de "Datos no válidos".

Respuesta por Consola:

- **Campos vacíos**



- **Campos rellenos OK.**



8. Usa el patrón “async-await” con un “try-catch” para imprimir por consola el error del paso anterior cuando se produzca.

Dentro de la función async-await recogemos los posibles de errores por medio de un bloque “try-catch”, dónde mostraremos por consola dichos errores.

```
async function showUserResult() {
    try {
        let alumn = await getAlumno(
            document.getElementById("name").value,
            document.getElementById("surname").value,
            document.getElementById("points").value

        );
        console.log(`
                Nombre : ${alumn.nombre},
                Apellidos : ${alumn.apellidos}.
                Su califiación es de
'${alumn.puntos}'
               `);
    } catch (error) {
        console.error(error);
    }
}
```

9. Código completo de la tarea.

```
// ** Desarrollo de la práctica Evaluable en
JavaScript **

// * Clase Alumno *
class Alumno {
    nombre;
    apellidos;
    puntos;

    constructor (nombre, apellidos, puntos){
        this.nombre = nombre;
        this.apellidos = apellidos;
        this.puntos = this.getPuntos(puntos);
    }

    setPuntos(puntos) {
       this.puntos = puntos;
    }

    getPuntos(puntos) {
        if (puntos >= 5)
            return `Apto`;
        return `No apto`;
    }
}

function getAlumno (name,surname,points) {
    return new Promise((resolve, reject) => {
        if (
            (name === '') || (surname === '') ||
(points === '')
        ) {
```

```
            reject({mensaje: `Datos no válidos`})
        }
        setTimeout(() => {
            resolve(new Alumno(name,surname,points))
        },2000)
    })
}

async function showUserResult() {
    try {
        let alumn = await getAlumno(
            document.getElementById("name").value,
            document.getElementById("surname").value,
            document.getElementById("points").value
        );
        console.log(`
                Nombre : ${alumn.nombre},
                Apellidos : ${alumn.apellidos}.
                Su califiación es de
'${alumn.puntos}'
               `);
    } catch (error) {
        console.error(error);
    }
}
```