

Nombre: Michael Steven Jiménez Basante

NRC: 23407

Materia: PROGRAMACIÓN INTEGRAL COMPONENTE WEB

Docente: Vilmer David Criollo Chanchicocha

Tema: Estructuras de Control en JavaScript

Taller 3

1. Descripción de la tarea:

Desarrollar una interfaz web que permita ejecutar 3 diferentes funciones programadas en un archivo de tipo .javascript aparte.

2. Desarrollo:

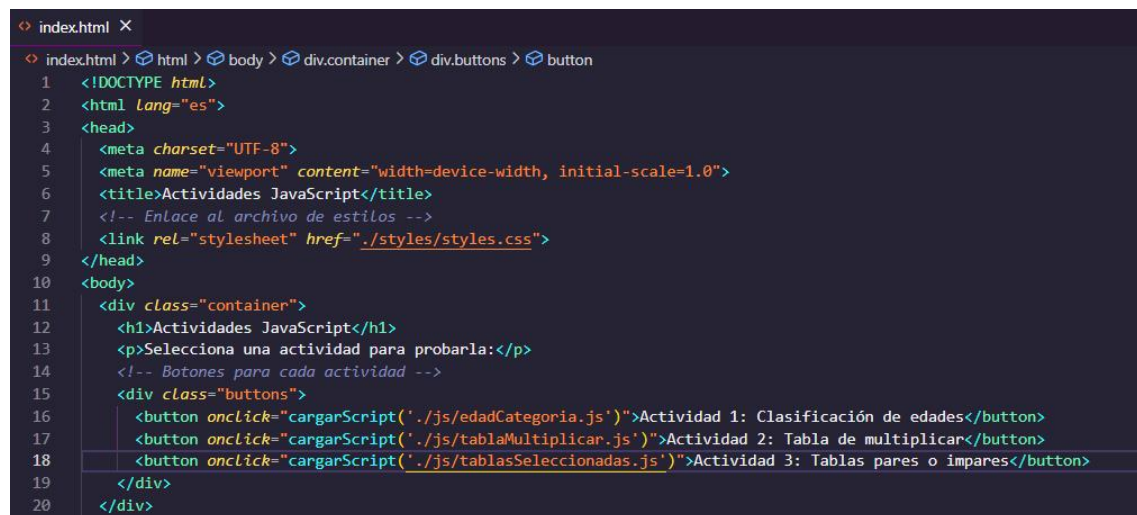
2.1 Enlace al GitHub:

<https://github.com/maik151/ActividadEstructurasControl.git>

3. Estructura del proyecto:

3.1 Archivo *index.html*:

Para esta actividad se creó una estructura de botones, mediante la etiqueta `<button>` se inserta un objeto de tipo botón a la estructura de la página. Dentro de la etiqueta creamos un atributo `"onclick()"` el cual funciona como un evento de tipo "click". el cual al presionar el botón definido, llama a la función `"cargarScript()"`.



```
index.html X
index.html > html > body > div.container > div.buttons > button
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Actividades JavaScript</title>
7   <!-- Enlace al archivo de estilos -->
8   <link rel="stylesheet" href="./styles/styles.css">
9 </head>
10 <body>
11   <div class="container">
12     <h1>Actividades JavaScript</h1>
13     <p>Selecciona una actividad para probarla:</p>
14     <!-- Botones para cada actividad -->
15     <div class="buttons">
16       <button onclick="cargarScript('./js/edadCategoria.js')">Actividad 1: Clasificación de edades</button>
17       <button onclick="cargarScript('./js/tablaMultiplicar.js')">Actividad 2: Tabla de multiplicar</button>
18       <button onclick="cargarScript('./js/tablasSeleccionadas.js')">Actividad 3: Tablas pares o impares</button>
19     </div>
20   </div>
```

La función `"cargarScript()"` se define dentro de una etiqueta `<script>` (la cual nos permite ejecutar código de tipo javascript dentro de HTML). Dentro de la función de `"cargarScript()"` definimos una función que toma como parámetro un objeto de tipo archivo.js; luego se declara una constante la cual toma el valor de algún script de un archivo anterior que ya se haya ejecutado (toma el valor usando un selector por id). Luego se declara un condicional `if` donde si existe un script anterior, se elimine usando el método de `"remove()"`.

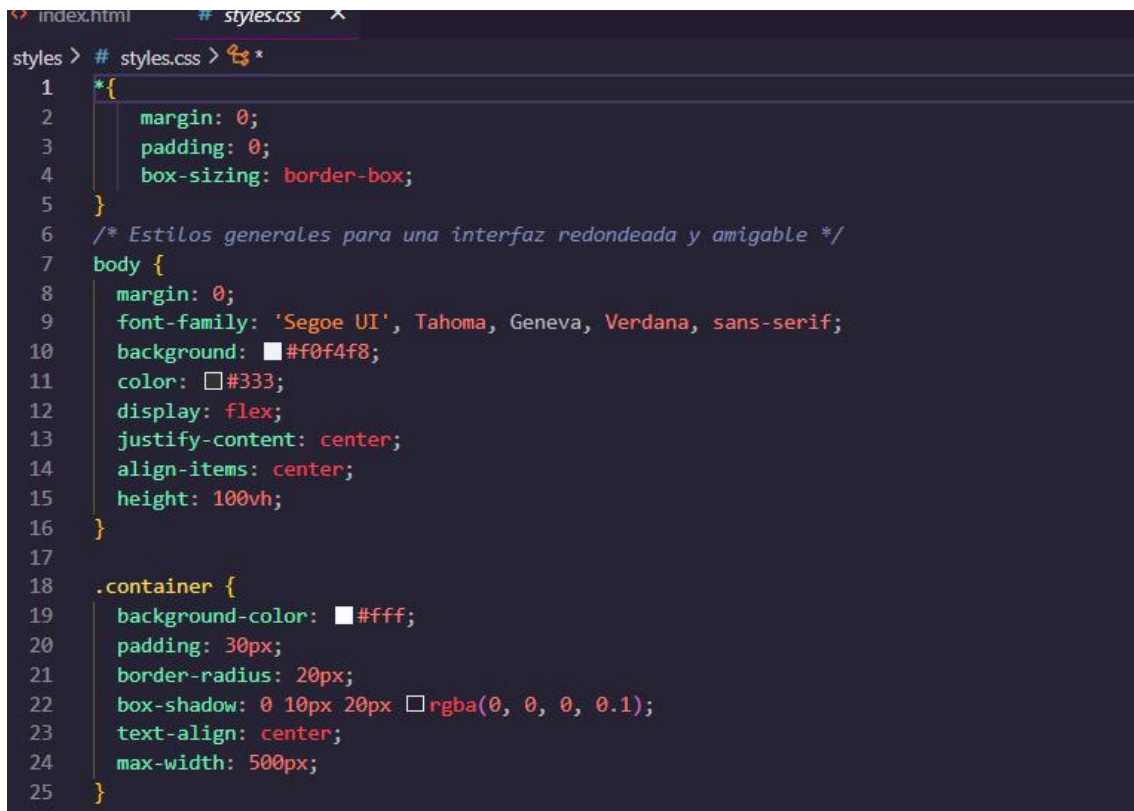
Luego creamos una constante donde creamos un elemento de tipo script, a dicho script le adjuntamos el archivo que se pasa como parametro en la funcion, y se le asigna el id de "scriptEjercicio". Luego usando el metodo de "appendChild(script)" agregamos la nueva etiqueta script al ultimo de nuestra estructura HTML.

```
<!-- Script para cargar dinámicamente los archivos JS -->
<script>
  function cargarScript(archivo) {
    // Elimina cualquier script cargado anteriormente
    const anterior = document.getElementById("scriptEjercicio");
    if (anterior) anterior.remove();

    // Crea y agrega el nuevo script
    const script = document.createElement("script");
    script.src = archivo;
    script.id = "scriptEjercicio";
    document.body.appendChild(script);
  }
</script>
```

3.2 Archivo styles.css:

Dentro de este archivo definimos los estilos en base a las clases definidas en el HTML.



```
1 *{
2   margin: 0;
3   padding: 0;
4   box-sizing: border-box;
5 }
6 /* Estilos generales para una interfaz redondeada y amigable */
7 body {
8   margin: 0;
9   font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
10  background-color: #f0f4f8;
11  color: #333;
12  display: flex;
13  justify-content: center;
14  align-items: center;
15  height: 100vh;
16 }
17
18 .container {
19   background-color: #fff;
20   padding: 30px;
21   border-radius: 20px;
22   box-shadow: 0 10px 20px rgba(0, 0, 0, 0.1);
23   text-align: center;
24   max-width: 500px;
25 }
```

3.3 Archivo *edadCategoria.js*:

Dentro de este archivo, definimos una variable con un scope global tipo **var** donde primero pasamos a un valor de entero (int) el valor obtenido por medio de un `prompt()`, ese valor lo guardamos en una variable **edad**.

Dicha variable la pasamos por condicionales tipo IF para poder definir los rangos de las edades, según eso, dependiendo de los intervalos, se muestra en consola a que grupo pertenece, si a niño, adolescente, adulto o adulto mayor.

```
js > JS edadCategoria.js > ...
1  // edadCategoria.js
2  // Actividad 1: Clasificación de edades usando if anidados
3
4  // Pedimos la edad al usuario
5  var edad = parseInt(prompt("Por favor, ingresa tu edad:"));
6
7  // Verificamos la categoría de edad
8  if (edad < 0) {
9      console.log("Edad no válida");
10 } else if (edad <= 12) {
11     console.log("Niño");
12 } else if (edad <= 17) {
13     console.log("Adolescente");
14 } else if (edad <= 59) {
15     console.log("Adulto");
16 } else {
17     console.log("Adulto mayor");
18 }
19
20 // Todas las salidas se muestran en la consola del navegador
```

3.4 Archivo *tablaMultiplicar.js*:

Dentro de este archivo definimos igual una **var** de nombre **numero**, donde primero validamos que el numero sea valido usando el metodo **isNaN()**. Si la variable no es un numero entonces ejecuta en consola el mensaje de que se digite un numero valido, y si no, entonces se ejecuta un ciclo **FOR**, donde empezamos en 1, avanzamos hasta menor igual a 12 y avanzamos de 1 en 1. El ciclo ejecuta un console.log donde usando una nomenclatura de backticks, creamos una plantilla usando la variable numero para por ejecutar las operaciones en la misma linea de console.log (esto nos permite ahorrar lineas de codigo en tener que definir otra variable resultado, ejecutar la operacion y luego mostrarla, etc).

```
js > JS tablaMultiplicar.js > ...
1 // tablaMultiplicar.js
2 // Actividad 2: Tabla de multiplicar usando for
3
4 // Pedimos al usuario el número deseado
5 var numero = parseInt(prompt("Ingresa un número para mostrar su tabla de multiplicar (1 al 12):"));
6
7 // Verificamos que sea un número válido
8 if (isNaN(numero)) {
9   console.log("Por favor, ingresa un número válido.");
10 } else {
11   console.log("Tabla de multiplicar del " + numero + ":");
12   // Bucle para mostrar la tabla del 1 al 12
13   for (let i = 1; i <= 12; i++) {
14     console.log(`${numero} x ${i} = ${numero * i}`);
15   }
16 }
17
```

3.5 Archivo *tablasSeleccionadas.js*:

De la misma forma declaramos una variable de scope global tipo **var**, de nombre **tipo**, donde esta va a tener el valor dado de un prompt, el cual se pasa a minúsculas en el caso de que se haya escrito en mayúsculas. Luego declaramos un if, donde usamos el operador de igualdad estricta (===), que compara tanto el valor como el tipo de dato, para verificar si la variable tipo es igual a "pares" o "impares". Esta doble condición se representa con el operador lógico || (OR).

Si se cumple alguna de estas dos condiciones, se ejecuta un ciclo for que comienza con un iterador i en 1, avanza de uno en uno (i++) hasta llegar a 10.

Dentro de ese mismo for, se declara otro if que evalúa si:

tipo es "pares" y $i \% 2 === 0$ (es decir, si el número actual es par), o

tipo es "impares" y $i \% 2 !== 0$ (es decir, si el número actual es impar).

Si la condición se cumple, se imprime en consola la tabla de multiplicar correspondiente al número actual.

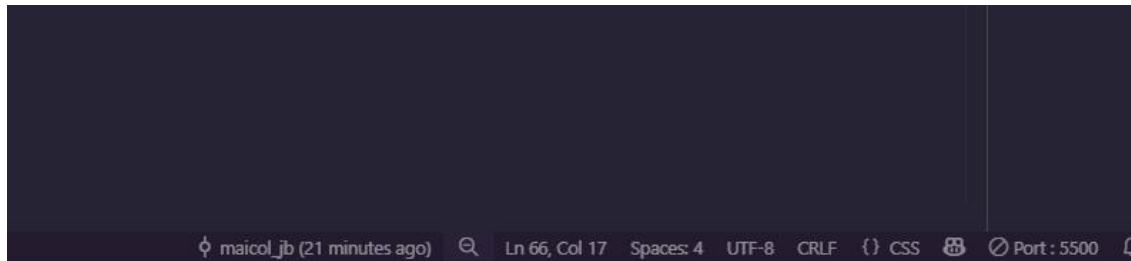
Para ello, se usa un segundo bucle for anidado que va desde 1 hasta 10, multiplicando el número actual (i) por cada uno de los valores del iterador j, e imprimiendo el resultado con formato:

i x j = resultado.

```
js > JS tablasSeleccionadas.js > ...
1 // tablasSeleccionadas.js
2 // Actividad 3: Tablas de multiplicar de pares o impares usando if + for
3
4 // Solicitamos al usuario si desea ver tablas pares o impares
5 var tipo = prompt("¿Quieres ver las tablas de multiplicar de números 'pares' o 'impares' entre 1 y 10?").toLowerCase();
6
7 // Verificamos la opción ingresada
8 if (tipo === "pares" || tipo === "impares") {
9   console.log("Mostrando tablas de multiplicar de números ${tipo}:");
10
11   for (let i = 1; i <= 10; i++) {
12     // Condición para filtrar pares o impares
13     if ((tipo === "pares" && i \% 2 === 0) || (tipo === "impares" && i \% 2 !== 0)) {
14       console.log(`\nTabla del ${i}:`);
15       for (let j = 1; j <= 10; j++) {
16         console.log(`${i} x ${j} = ${i * j}`);
17       }
18     }
19   }
20 } else {
21   console.log("Opción no válida. Escribe 'pares' o 'impares'.");
22 }
23
24
```

4. Ejecucion del Proyecto

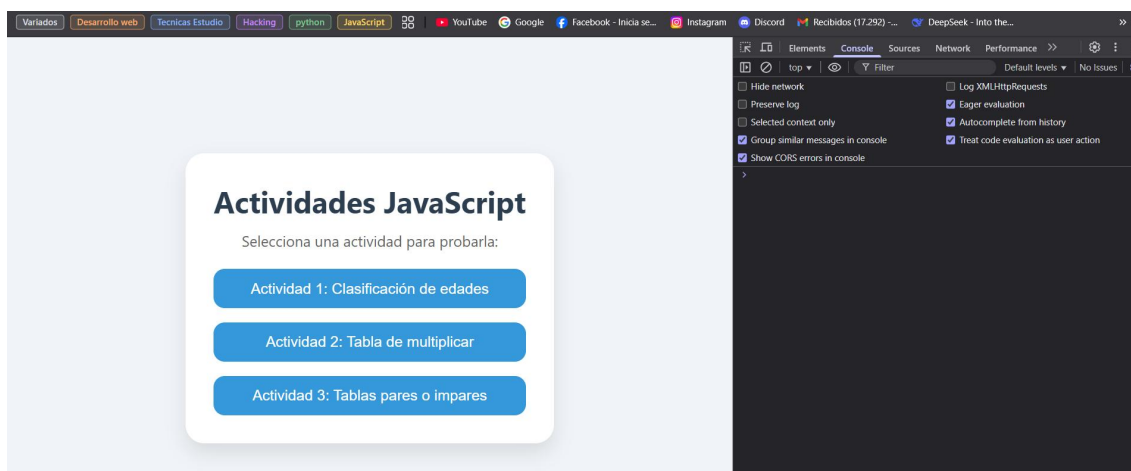
Para realizar la ejecucion del proyecto usamos la herramienta de *liveServer* la cual nos permite ejecutar el proyecto en un miniservidor que recepta las solicitudes http, estas solicitudes se hacen bajo el localhost y el puerto 5500.



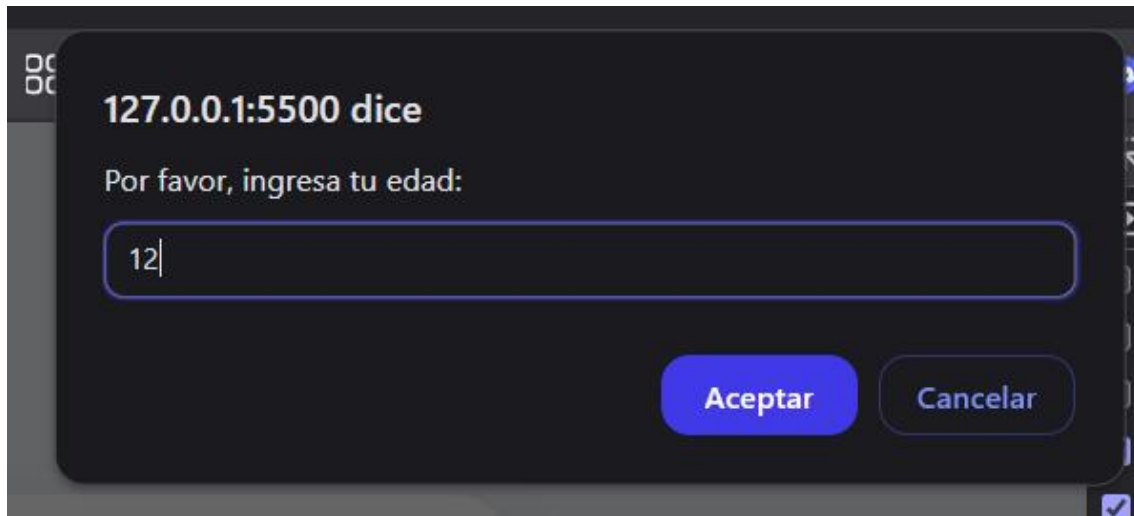
Al momento de ejecutar la herramienta tenemos que:



Podemos presion F12 o usar click derecho>inspeccionar para poder abrir la consola del navegador (consola donde corre codigo js)



Ahora segun hagamos click en cada uno de los botones se ejecuta, la funcion definida. Para el caso de la primera funcion tenemos que:

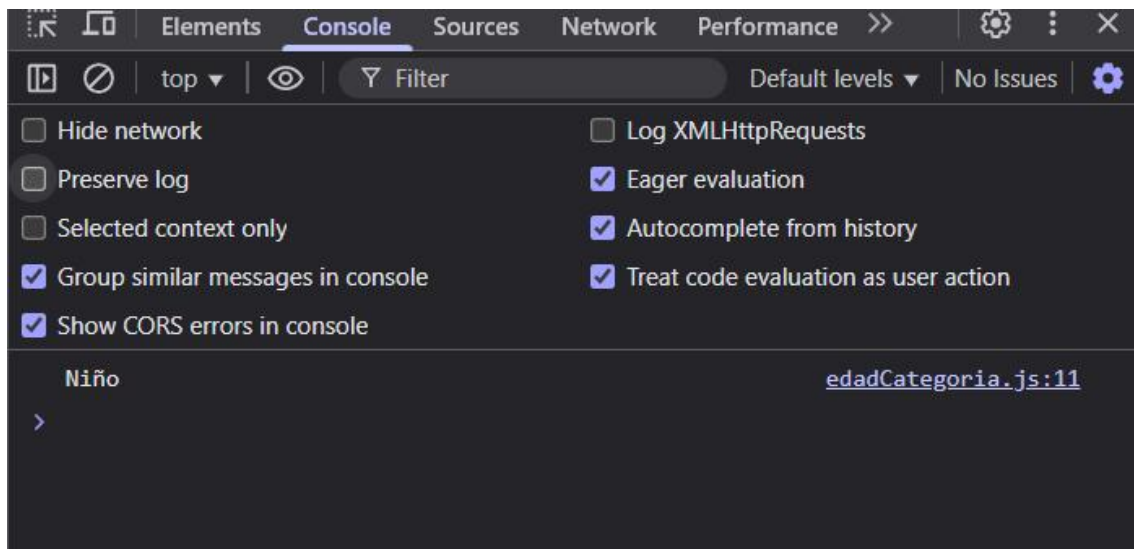


127.0.0.1:5500 dice

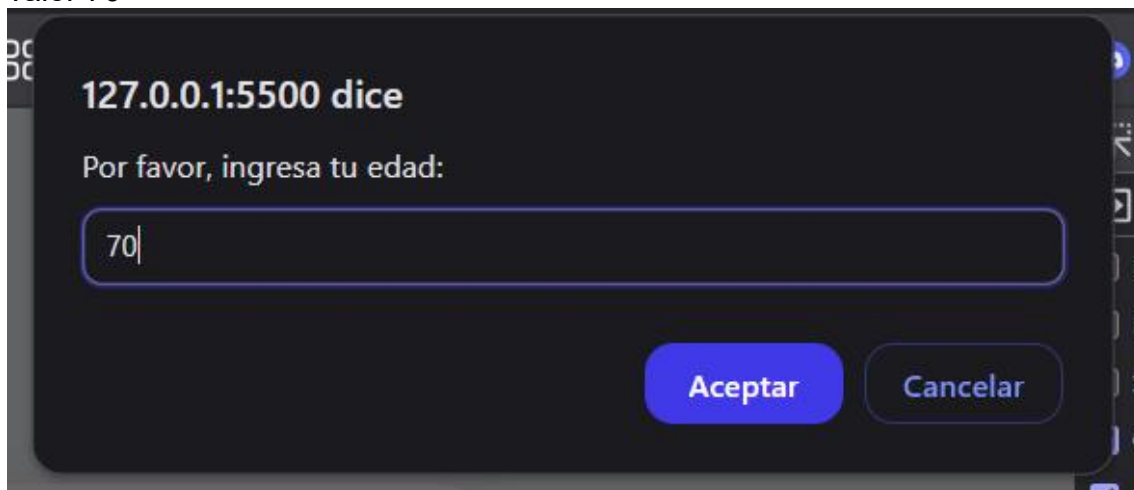
Por favor, ingresa tu edad:

Aceptar Cancelar

Se ejecuta el *prompt()*, donde introducimos un valor 12, y en la consola, nos arroja que pertenece a la categoria de niño:



Si introducimos otro valor la variable se sobrescribe, en este caso usamos el valor 70

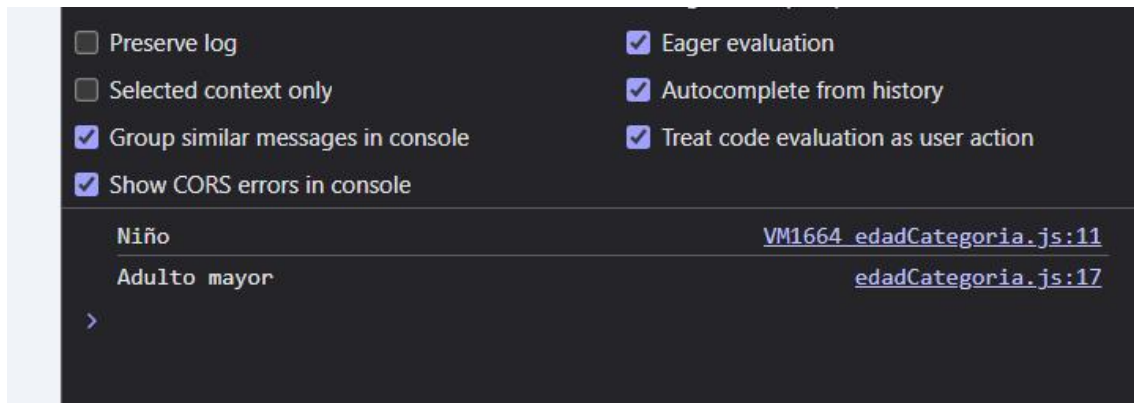


127.0.0.1:5500 dice

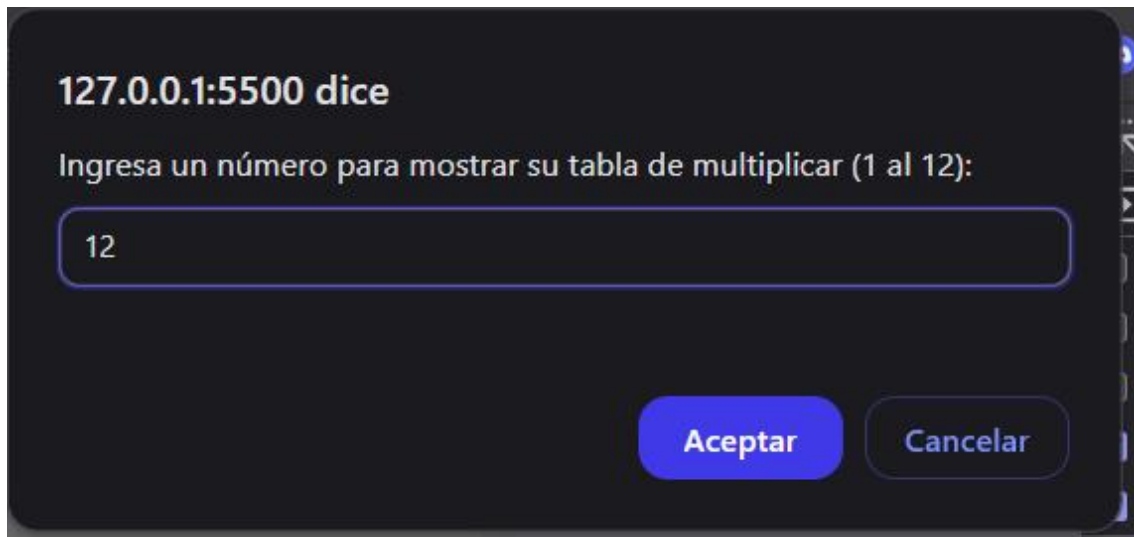
Por favor, ingresa tu edad:

Aceptar Cancelar

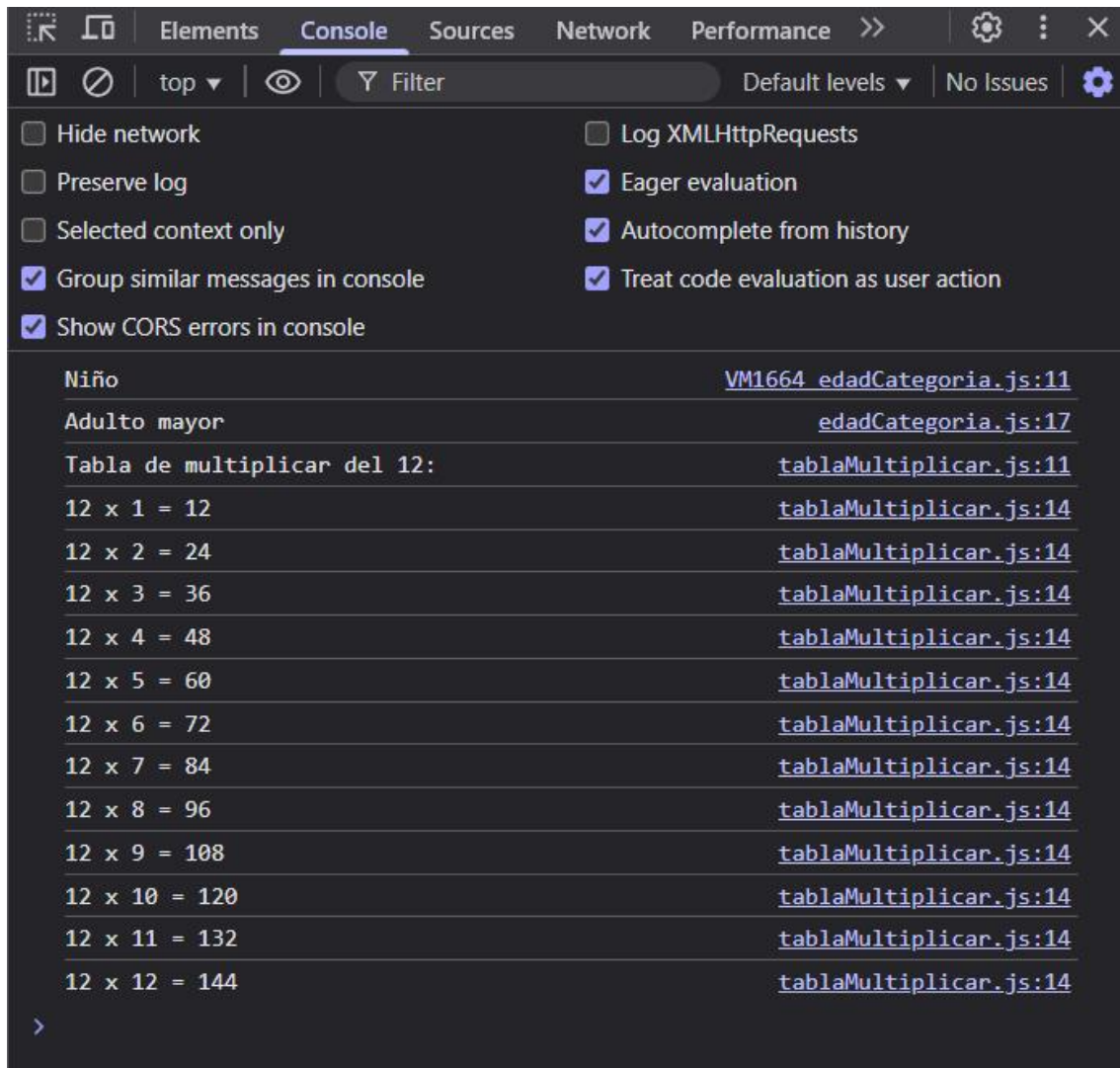
Nos arroja que pertenece a la categoría de adulto mayor:



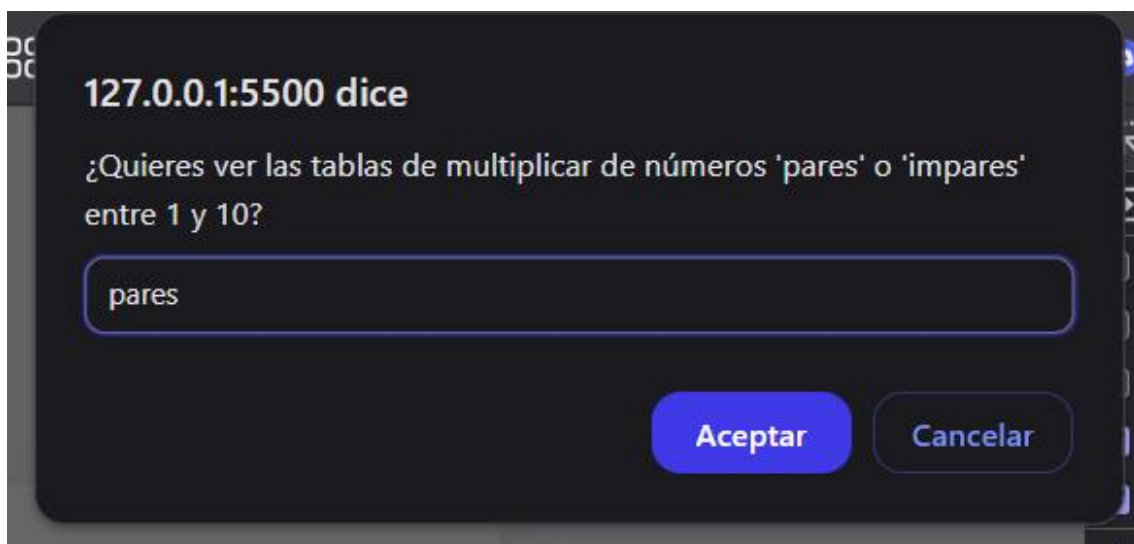
Para la segunda función se ejecuta otro **prompt()** donde tenemos que ingresar un número que nos muestre la tabla de multiplicar de ese número, este caso insertamos 12:



Donde la consola nos arroja la tabla de multiplicar basada en ese numero:



Para el tercer metodo de la misma manera obtenemos los datos desde un **prompt()**. En este caso, debemos ingresar si queremos par o impar, tecleamos la palabra "pares".



Dentro de la consola nos muestra las tablas de multiplicar de cada uno de los numero pares:

```
☒ Show CORS errors in console

Tabla del 2:
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20

Tabla del 4:
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
```

$$6 \times 10 = 60$$

[tablasSeleccionadas.js:16](#)

[tablasSeleccionadas.js:14](#)

Tabla del 8:

$$8 \times 1 = 8$$

[tablasSeleccionadas.js:16](#)

$$8 \times 2 = 16$$

[tablasSeleccionadas.js:16](#)

$$8 \times 3 = 24$$

[tablasSeleccionadas.js:16](#)

$$8 \times 4 = 32$$

[tablasSeleccionadas.js:16](#)

$$8 \times 5 = 40$$

[tablasSeleccionadas.js:16](#)

$$8 \times 6 = 48$$

[tablasSeleccionadas.js:16](#)

$$8 \times 7 = 56$$

[tablasSeleccionadas.js:16](#)

$$8 \times 8 = 64$$

[tablasSeleccionadas.js:16](#)

$$8 \times 9 = 72$$

[tablasSeleccionadas.js:16](#)

$$8 \times 10 = 80$$

[tablasSeleccionadas.js:16](#)

[tablasSeleccionadas.js:14](#)

Tabla del 10:

$$10 \times 1 = 10$$

[tablasSeleccionadas.js:16](#)

$$10 \times 2 = 20$$

[tablasSeleccionadas.js:16](#)

$$10 \times 3 = 30$$

[tablasSeleccionadas.js:16](#)

$$10 \times 4 = 40$$

[tablasSeleccionadas.js:16](#)

$$10 \times 5 = 50$$

[tablasSeleccionadas.js:16](#)

$$10 \times 6 = 60$$

[tablasSeleccionadas.js:16](#)