

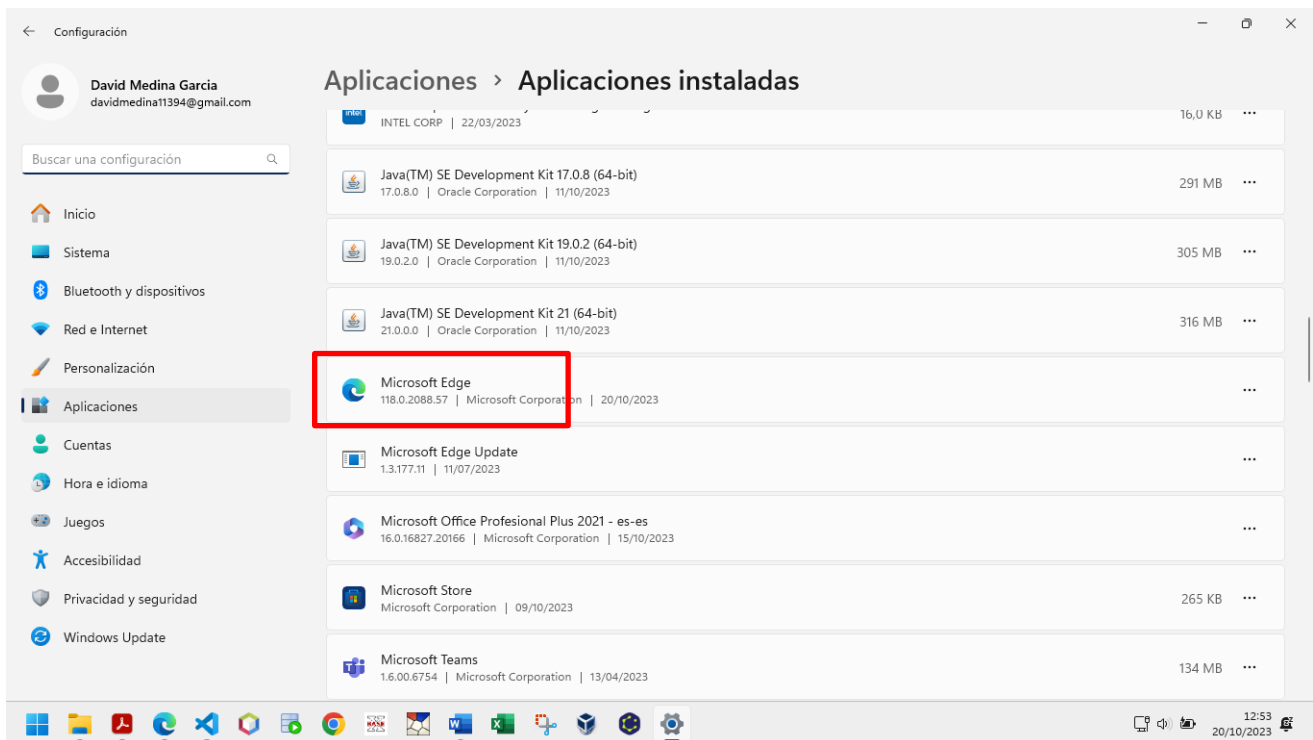
**CICLO: DESARROLLO APLICACIONES WEB-DISTANCIA**  
**MÓDULO: DESARROLLO WEB EN ENTORNO CLIENTE**  
**TAREA: DWEC-01**  
**ALUMNO: DAVID MEDINA GARCIA**

**1. Instalar navegadores adicionales.**

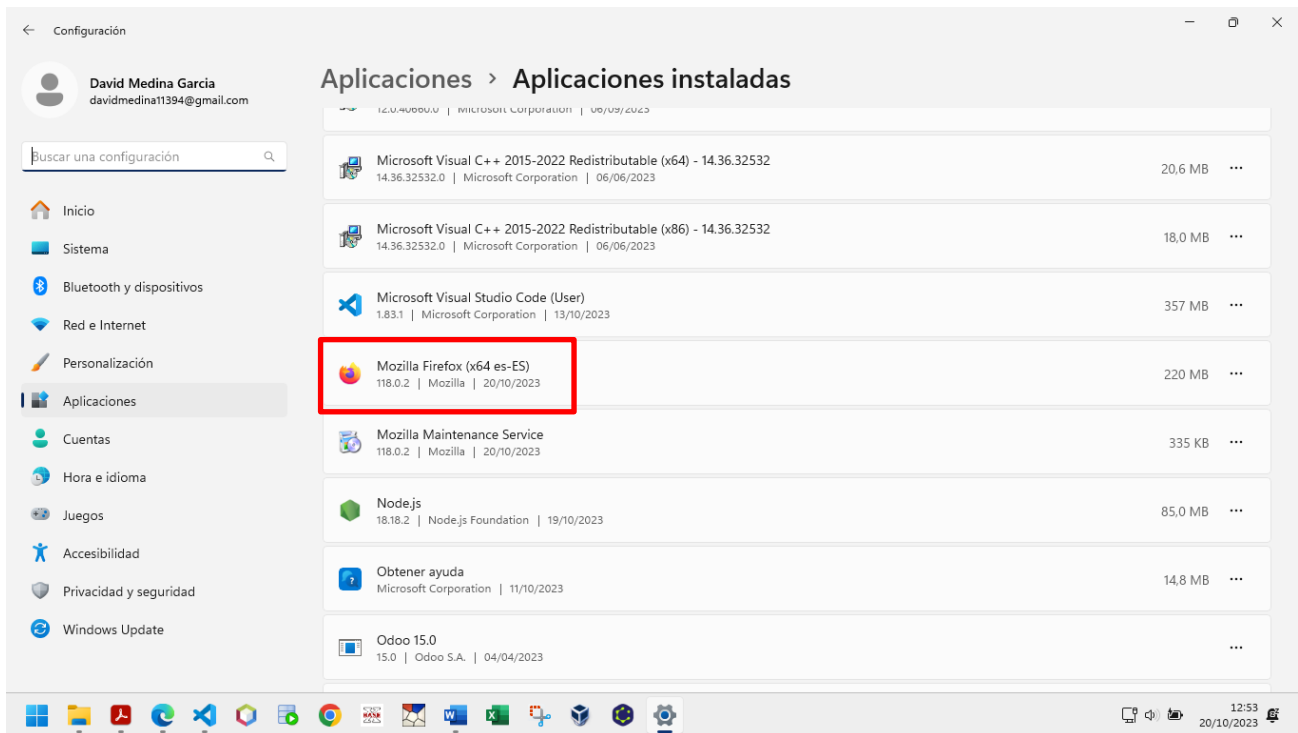
Navegadores adicionales instalados: Microsoft Edge y Mozilla Firefox. Chrome estaba previamente instalado. Desde google buscamos la pagina web de cada navegador, nos lo descargamos y lo instalamos con el asistente que se ejecuta, tal como se muestra en las siguientes imágenes.

**Navegadores instalados**

**Microsoft Edge**



## Mozilla Firefox



## 2. Instalación de editor web. Razones para elegir el editor web.

**Editor elegido:** Apache Netbeans

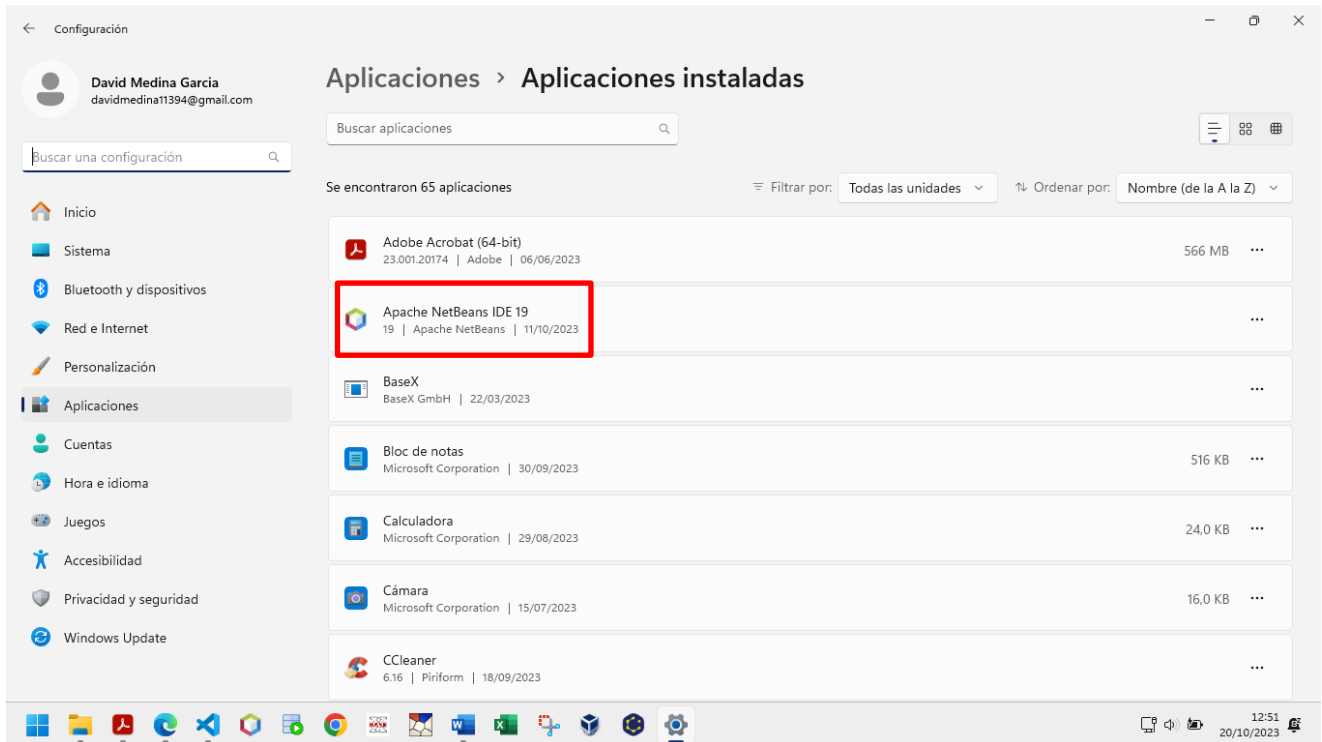
Después de buscar información de varios programas para comparar, me decidí por Apache Netbeans por las siguientes razones:

- Es gratuito:** Puedes acceder a todas sus funciones sin necesidad de realizar ningún pago
- Sencillez:** Es más sencillo de usar que otras alternativas.
- Multiplataforma:** Se puede emplear en distintos dispositivos, así como ejecutarse en diferentes sistemas operativos (Mac OS, Windows, Linux y Solaris).
- Multilenguaje:** Permite desarrollar aplicaciones multilenguaje. Es decir, no solo opera con Java; sino que Eclipse puede utilizar otros lenguajes como PHP, Python, C o Ruby, entre otros.

Una razón adicional para elegir este editor es que es utilizado en otras asignaturas del ciclo y así estaremos más familiarizados con él.

Para instalarlo nos vamos a su pagina web , nos lo descargamos y lo instalamos con el asistente que se ejecuta.

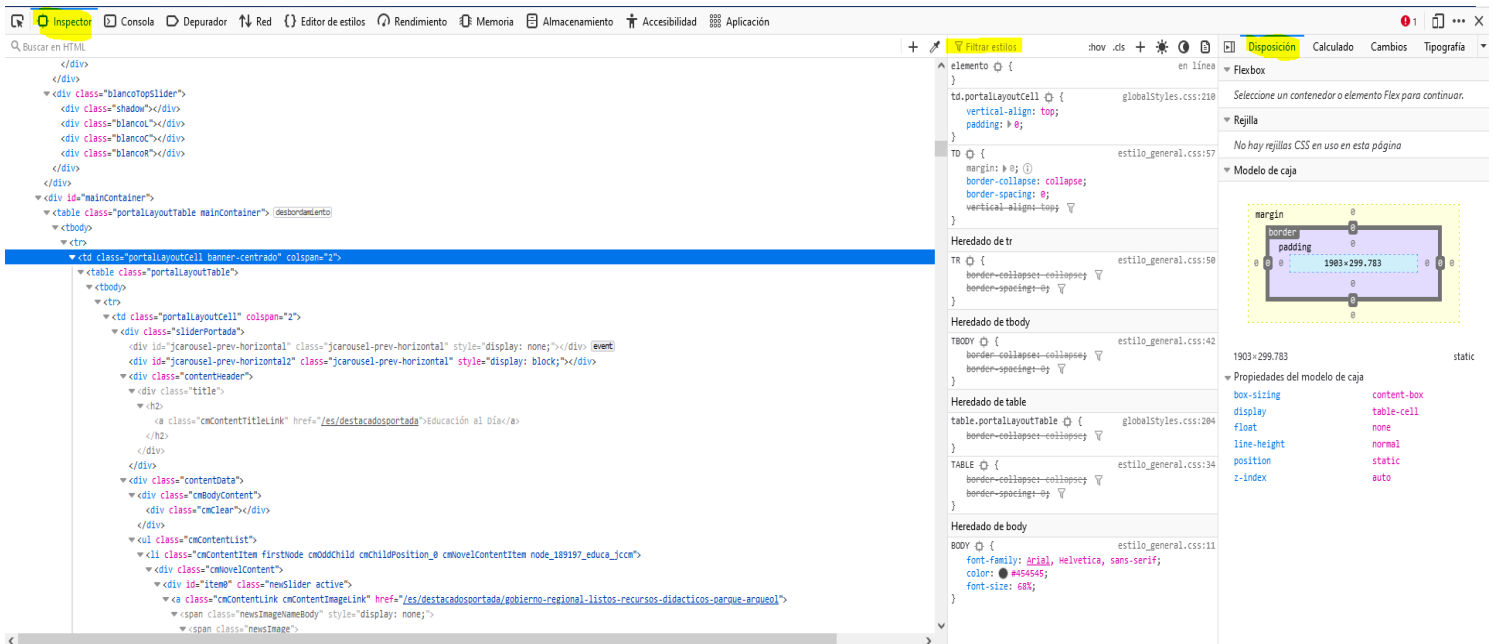
## Apache Netbeans



## Herramienta de desarrolladores:

- Firefox

Podemos abrir la herramienta para desarrolladores pulsando F12:

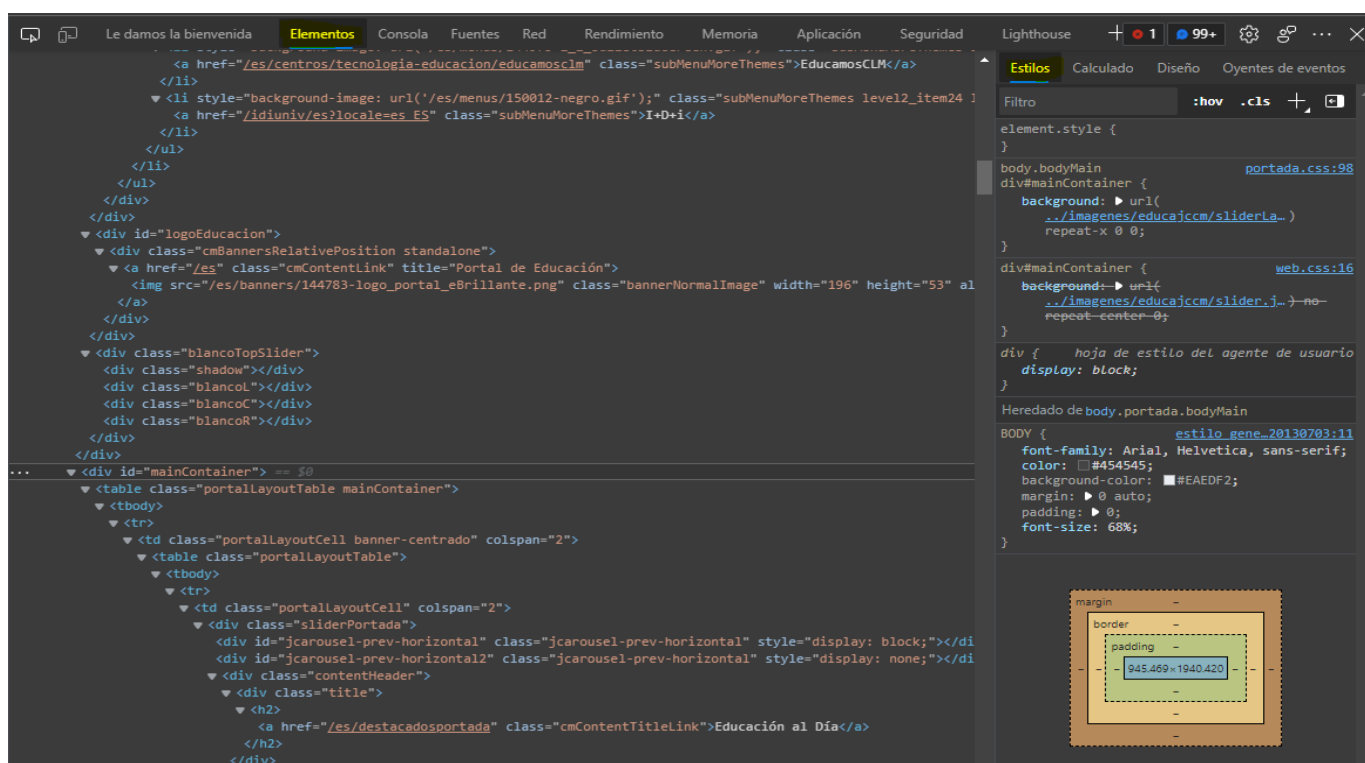


Se nos abrirá una ventana en la parte inferior de la página abierta en la pestaña **Inspector** (que vendría a ser lo mismo que la pestaña **Elementos** en Edge y Chrome). Desde ahí podemos ver todo el código **HTML** de la página (se puede editar el código **HTML** desde aquí y ver los cambios al instante). En la pestaña Consola podemos probar scripts de **Javascript**.

A la derecha tenemos una pestaña que hace referencia a los estilos. Al seleccionar un elemento en esta pestaña nos mostrará los estilos que tiene asociados. Destacamos la pestaña **Disposición** de la derecha, donde podemos observar la disposición del elemento.

- **Microsoft Edge**

Podemos abrir la herramienta para desarrolladores en pulsando F12:



Se nos abrirá una ventana en la parte derecha de la página abierta en la pestaña **Elementos** para ver todo el código **HTML** de la página (se puede editar el código **HTML** desde aquí y ver los cambios al instante). En la pestaña Consola podemos probar scripts de **Javascript**.

A la derecha tenemos la pestaña **Estilos**. Al seleccionar un elemento de la pestaña **Elementos**, nos mostrará los estilos que tiene asociados, así como la disposición del mismo.

### 3. Validación página web y corrección de errores

Usando la dirección de validación del código HTML del W3C ([validator.w3.org](http://validator.w3.org)) realizaremos la validación de la página de la Universidad de Castilla-La Mancha ([www.uclm.es](http://www.uclm.es)) e indicaremos los tipos de errores encontrados, nos ha dado **10** errores y **11** warnings.

Errors (10) · [Hide all errors](#) · [Show all errors](#)

1

☒ Element `meta` is missing one or more of the following attributes: `content`, `property`. (2)

2

☒ Bad value `https://fonts.googleapis.com/css?family=Archivo+Narrow|Open+Sans` for attribute `href` on element `link`: Illegal character in query: `|` is not allowed.

3

☒ Element `script` must not have attribute `charset` unless attribute `src` is also specified.

4

☒ Attribute `cookie-consent` not allowed on element `script` at this point. (3)

5

☒ Text not allowed in element `ul` in this context. (3)

Warnings (11) · [Hide all warnings](#) · [Show all warnings](#)

1

☐ The `type` attribute is unnecessary for JavaScript resources. (7)

2

☐ The `charset` attribute on the `script` element is obsolete. (2)

3

☐ The `navigation` role is unnecessary for element `nav`. (2)

A continuación se propone una solución para 3 de estos errores:

- **Error 1:**

**Error** Bad value `https://fonts.googleapis.com/css?family=Archivo+Narrow|Open+Sans` for attribute `href` on element `link`: Illegal character in query: `|` is not allowed.

From line 8, column 1562; to line 8, column 1658

```
80.png" /><link href="https://fonts.googleapis.com/css?family=Archivo+Narrow|Open+Sans" rel="stylesheet" /><link
```

En este error vemos que existe un carácter no permitido (|) en la consulta del atributo **href**. Para solucionarlo debemos sustituir dicho carácter (|) por su codificación "%7C".

- **Error 2:**

**Error** Element `script` must not have attribute `charset` unless attribute `src` is also specified.

From line 19, column 5; to line 19, column 51

```
ript><script type="text/javascript" charset="UTF-8"><script type="text/javascript" src="d
```

Este error nos indica que el atributo **charset** no debería estar si no lo indica el atributo **src**. La solución pasa por eliminar el atributo **charset** ya que no es necesario usarlo en esta ocasión.

- Error 3:

**Error** Attribute `cookie-consent` not allowed on element `script` at this point.

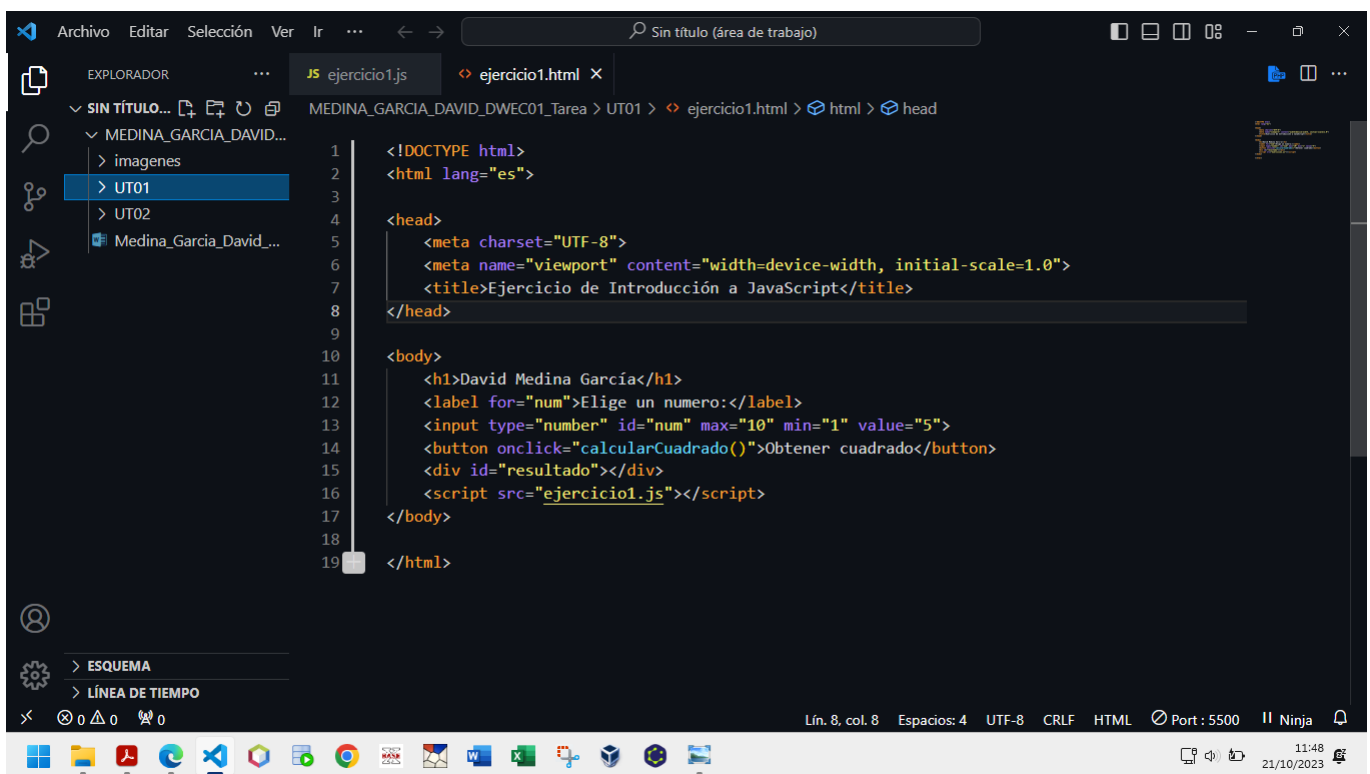
From line 42, column 5; to line 42, column 56

```
ript><script type="text/plain" cookie-consent="tracking"></script>
```

Este error nos indica que el elemento `script` no permite que este el atributo `cookie-consent`. Una solución sería usar un atributo personalizado (`data-cookie-consent="tracking"`).

#### 4. Ejercicio UT01.1: Integrar el código JavaScript en nuestro sitio web

- CODIGO HTML



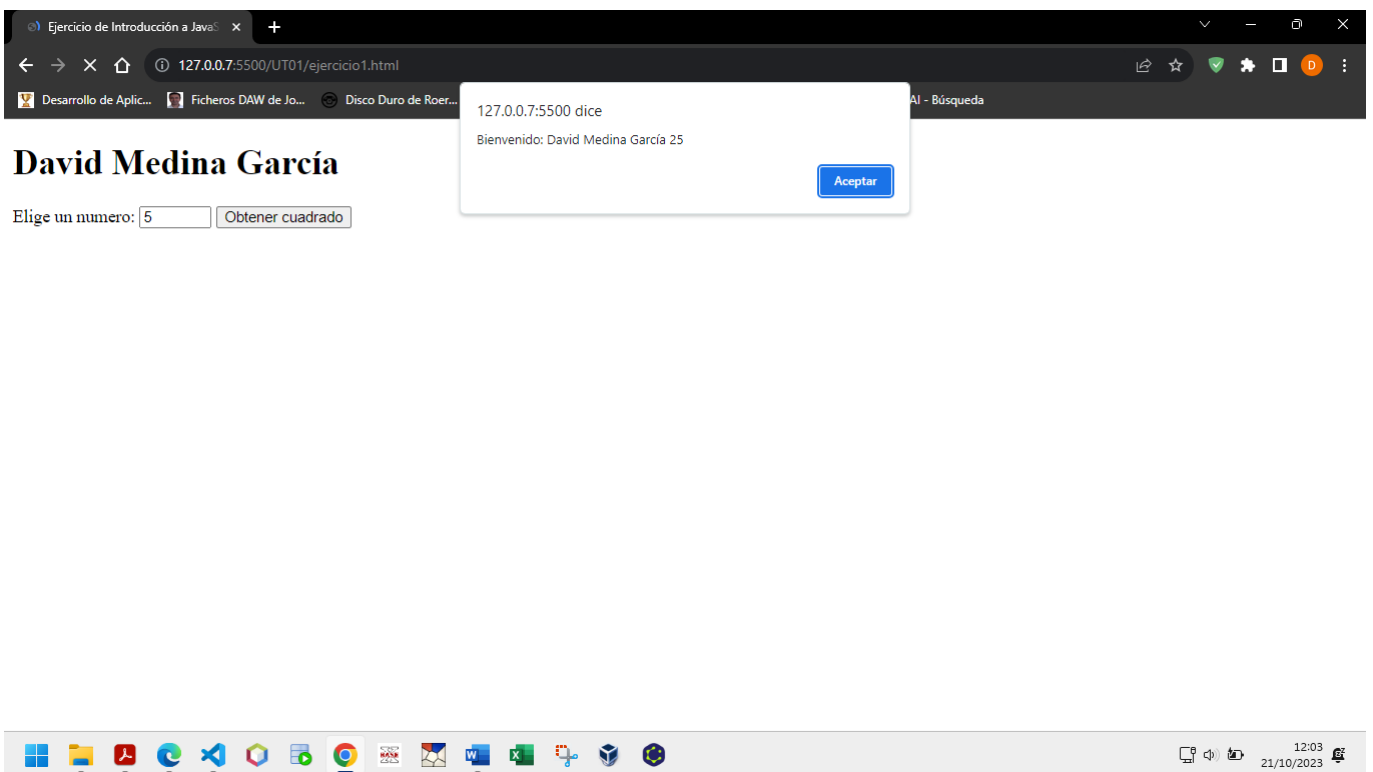
```
1 <!DOCTYPE html>
2 <html lang="es">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Ejercicio de Introducción a JavaScript</title>
8 </head>
9
10 <body>
11   <h1>David Medina García</h1>
12   <label for="num">Elige un numero:</label>
13   <input type="number" id="num" max="10" min="1" value="5">
14   <button onclick="calcularCuadrado()">Obtener cuadrado</button>
15   <div id="resultado"></div>
16   <script src="ejercicio1.js"></script>
17 </body>
18
19 </html>
```

- **CODIGO JAVASCRIPT**

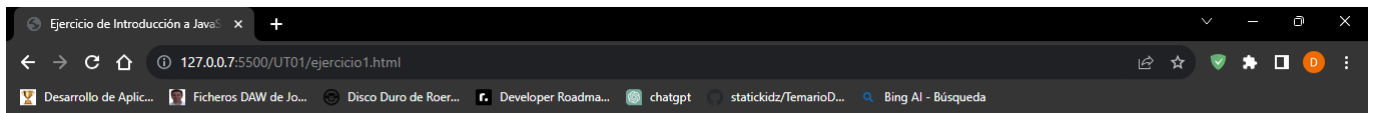
```
1 // Variable que almacena el nombre
2 let nombre = "David Medina García";
3 // Muestra un mensaje de bienvenida utilizando la variable 'nombre' y la función 'cuadrado'
4 alert("Bienvenido: " + nombre + " " + cuadrado(5));
5 // Define la función 'cuadrado' que toma un número como argumento
6 // Comment Code
7 function cuadrado(num) {
8   // Calcula el cuadrado del número
9   let resultado = num * num;
10  // Muestra un mensaje en la consola con el resultado
11  console.log("El cuadrado de " + num + " es: " + resultado);
12  // Devuelve el resultado
13  return resultado;
14 }
15 // Define la función 'calcularCuadrado' para procesar un número ingresado por el usuario
16 // Comment Code
17 function calcularCuadrado() {
18   // Obtiene el elemento de entrada con el ID "num"
19   let input = document.getElementById("num");
20   // Obtiene el elemento donde se mostrará el resultado
21   let resultado = document.getElementById("resultado");
22   // Muestra un mensaje en la consola con el valor obtenido del input
23   console.log("El valor obtenido del input es " + input.value);
24   // Muestra el resultado en el elemento HTML
25   resultado.innerHTML = "El cuadrado es: " + cuadrado(input.value);
26 }
27 // Llama a la función 'calcularCuadrado' para mostrar el cuadrado de un número al cargar la página
28 calcularCuadrado();
29 // Configura una función para ejecutarse cuando se carga la ventana. Como la función no está definida la dejo comentada.
30 // window.onload = ejemploConsola;
```

- **RESULTADO**

-Primero se carga la página y salta el método **alert()**.



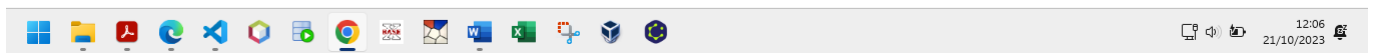
-A continuación, se muestra la página cargada, habiendo llamado a la función **calcularCuadrado()**.



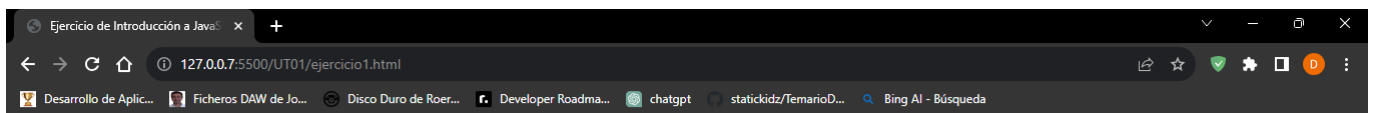
## David Medina García

Elige un numero:

El cuadrado es: 25



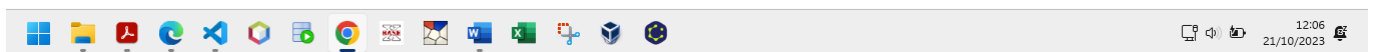
-Y pulsamos el botón para que haga la raíz cuadrada del número insertado



## David Medina García

Elige un numero:

El cuadrado es: 64

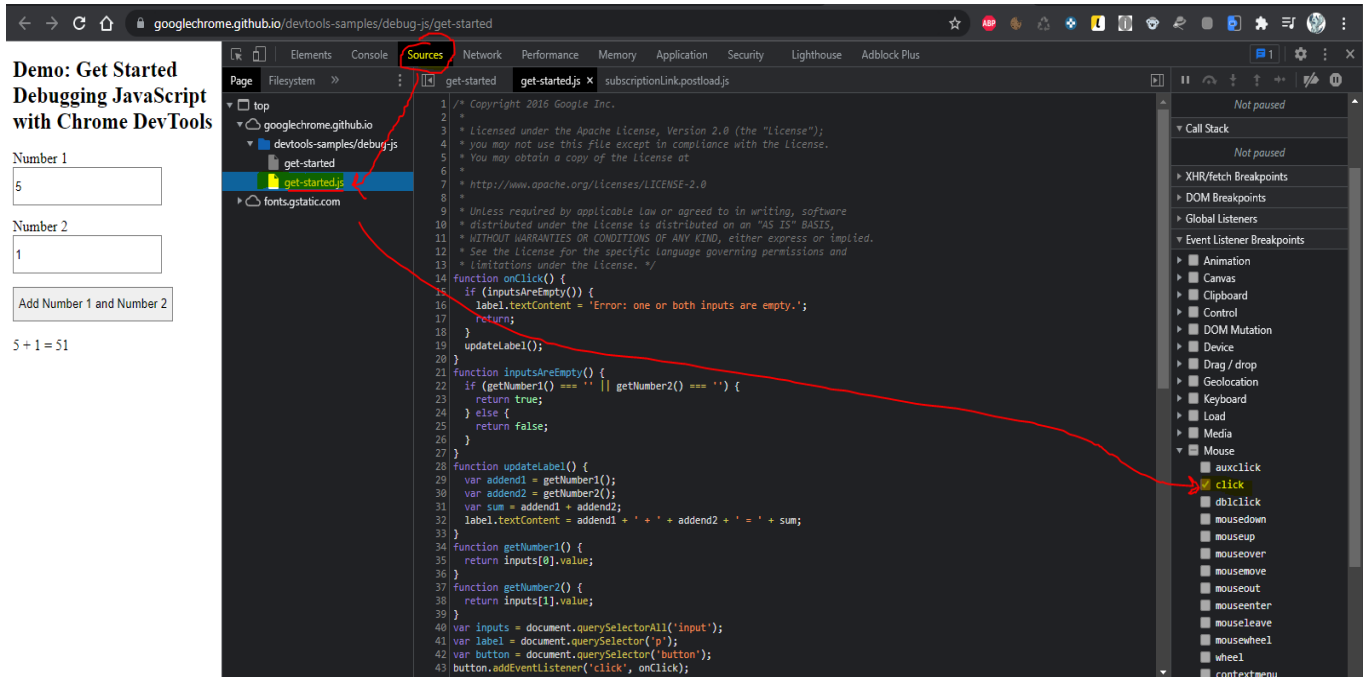




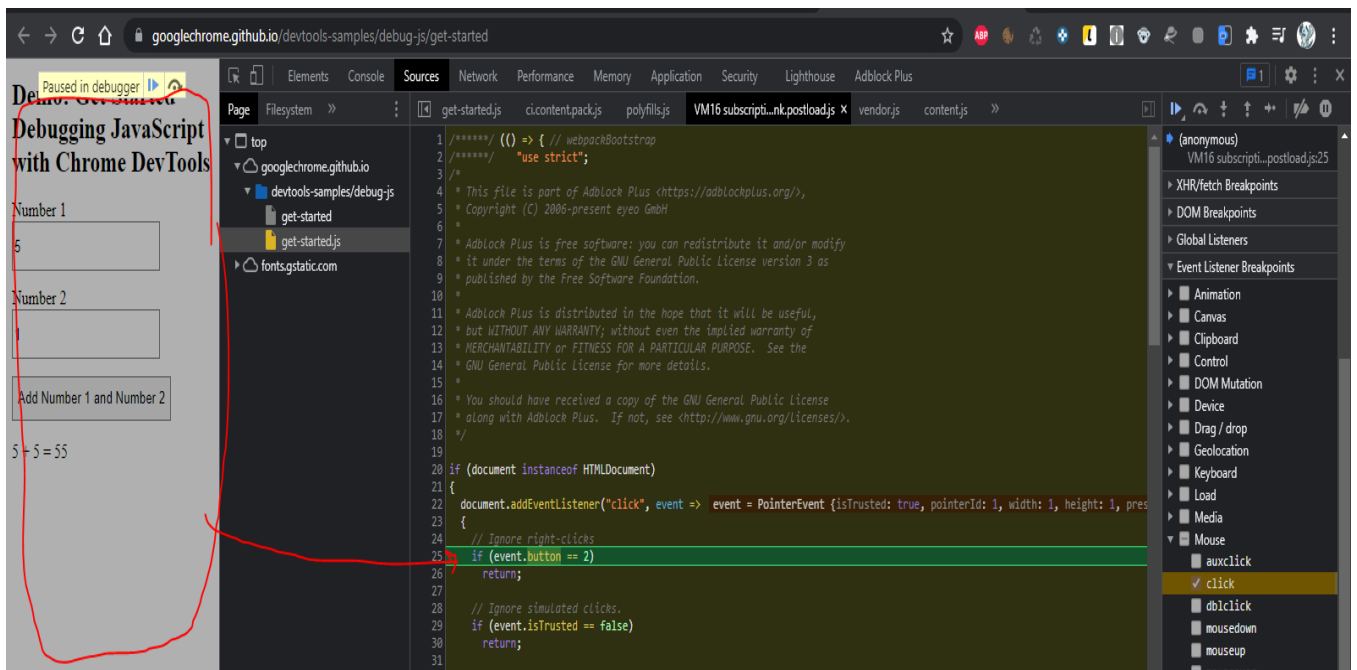
## 5. Ejercicio UT01.2: Utilización del depurador

Para este ejercicio he usado mi navegador habitual, Chrome.

Abrimos la página del ejemplo, accedemos a la Herramienta para desarrolladores (F12) y vamos a la pestaña **Sources** y seleccionamos el archivo **.js**. Abrimos **Event Listener Breakpoints** > **Mouse** > **Click** para indicar que debe realizarse un punto de ruptura ante la detección de un click.



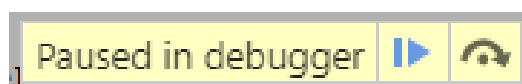
Si hacemos un click en cualquier parte de la página, detectará el evento.



Normalmente nos interesa evaluar un click válido (una función **onClick**) o un botón concreto, no cualquier click. Para ello podemos poner el punto de ruptura en la línea de código pertinente.

```
13  * Limitations under the License. */
14  function onClick() {
15      if (!inputsAreEmpty()) {
16          label.textContent = 'Error: one or both inputs are empty.';
17          return;
18      }
19      updateLabel();
20  }
21  function inputsAreEmpty() {
22      if (getNumber1() === '' || getNumber2() === '') {
23          return true;
24      } else {
25          return false;
26      }
27  }
28  function updateLabel() {
29      var addend1 = getNumber1();
30      var addend2 = getNumber2();
31      var sum = addend1 + addend2;
32      label.textContent = addend1 + ' + ' + addend2 + ' = ' + sum;
33  }
34  function getNumber1() {
35      return inputs[0].value;
36  }
37  function getNumber2() {
38      return inputs[1].value;
39  }
40  var inputs = document.querySelectorAll('input');
41  var label = document.querySelector('p');
42  var button = document.querySelector('button');
43  button.addEventListener('click', onClick);
44  }
```

Para analizar paso a paso las líneas de código que se ejecutan tras realizar la acción tenemos los siguientes botones.

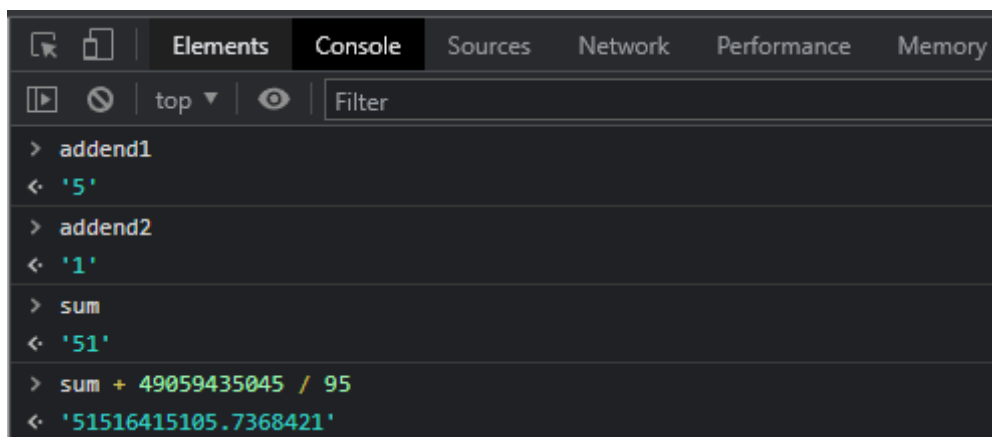


Con ellos podemos inspeccionar todos los pasos (incluido los valores que toman las variables involucradas) que se realizan hasta que la función completa su función.

```
8  function updateLabel() {
9      var addend1 = getNumber1(); addend1 = "5"
10     var addend2 = getNumber2(); addend2 = "1"
11     var sum = addend1 + addend2; sum = "51", addend1 = "5"
12     label.textContent = addend1 + ' + ' + addend2 + ' = ' + sum;
13 }
```

```
▼ Scope
▼ Local
  ► this: Window
    addend1: "5"
    addend2: "1"
    sum: "51"
```

También es útil utilizar la pestaña consola para verificar las variables y realizar operaciones.



En resumen, es indispensable saber utilizar esta herramienta (**debug**) para diagnosticar los problemas que nos aparezcan en nuestro código.