

Módulo 9 - Desarrollo Web

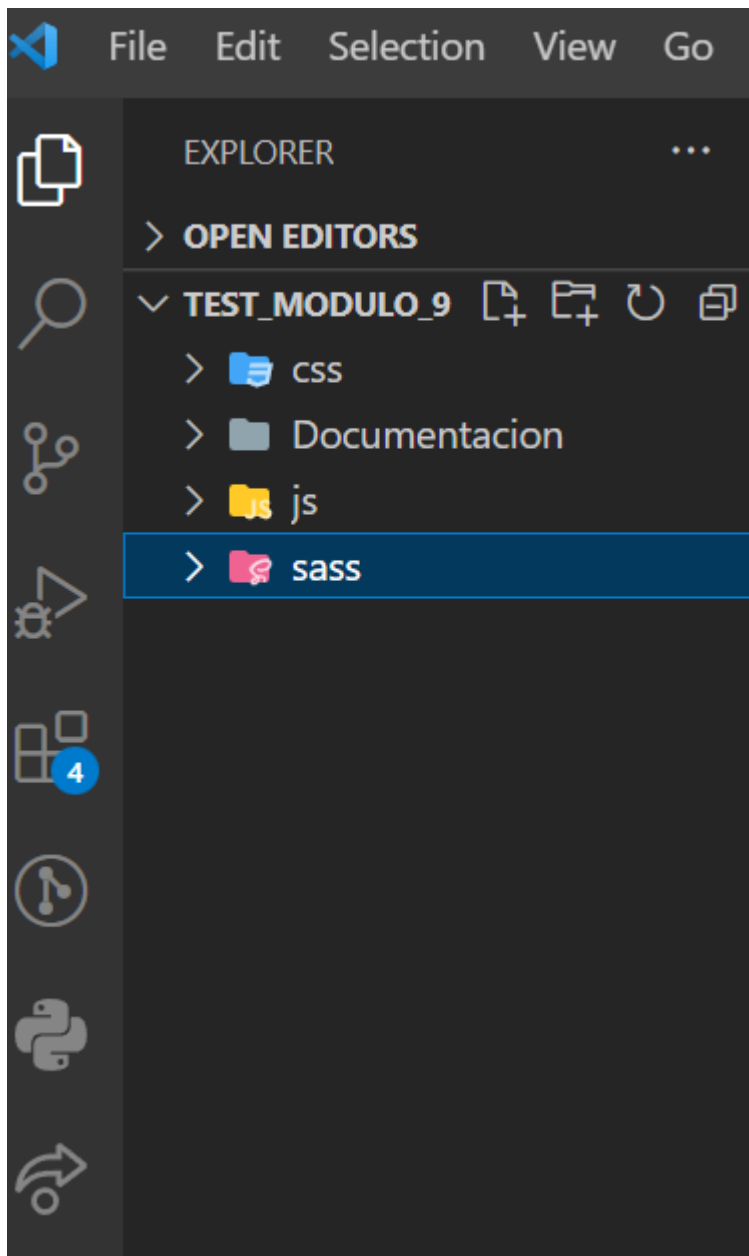
Desarrolla los pasos que habrá que seguir para definir todo el código de la siguiente manera:

1. Instalar Sass .
2. Crea un directorio vacío llamado evaluación, ábrelo con Visual Studio Code y crea a su vez tres directorios en la raíz denominados css, sass y js.
3. En el directorio sass, implementa el patrón 7-1 con sus diferentes directorios y un archivo main.scss.
4. Desarrolla los estilos del formulario usando el patrón 7-1 y Flexbox.
5. Compila el archivo main.scss a un archivo styles.css en el directorio css.
6. Crea el archivo index.html con los elementos para el formulario.
7. Implementa la lógica JavaScript para que, al pulsar en un botón de "Enviar", los datos del formulario se almacenen en un objeto.
8. Enlaza el archivo main.js con index.html y ejecuta el proyecto en el navegador.
9. Utiliza las herramientas de desarrollador para obtener diferentes valores por depuración.

1. Instalamos Sass poniendo en la terminal de vs code:

```
sass sass/main.scss css/style.css
```

2. Crea un directorio vacío llamado [test_modulo_9](#), ábrelo con Visual Studio Code y crea a su vez tres directorios en la raíz denominados css, sass y js.



3. En el directorio sass, implementa el patrón 7-1 con sus diferentes directorios y un archivo main.scss.

File Edit Selection View Go

EXPLORER

> OPEN EDITORS

TEST_MODULO_9

Documentation

js

sass

abstract

functions.scss

mixins.scss

variables.scss

base

reset.scss

typography.scss

components

buttons.scss

forms.scss

layout

footer.scss

header.scss

pages

about.scss

home.scss

themes

theme-dark.scss

theme-default.scss

vendors

bootstrap.scss

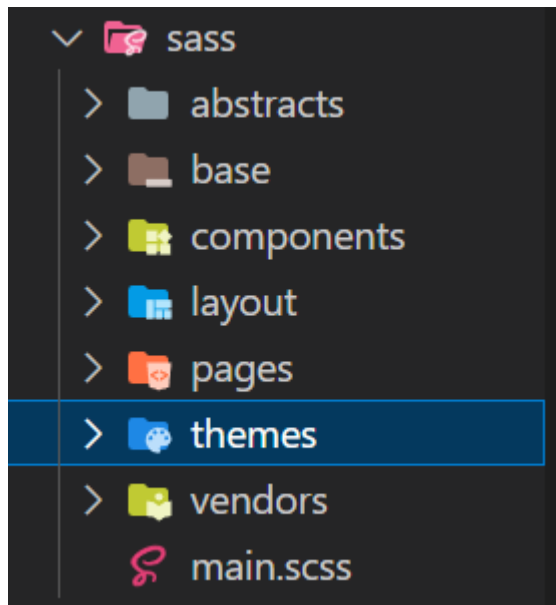
main.scss

> OUTLINE

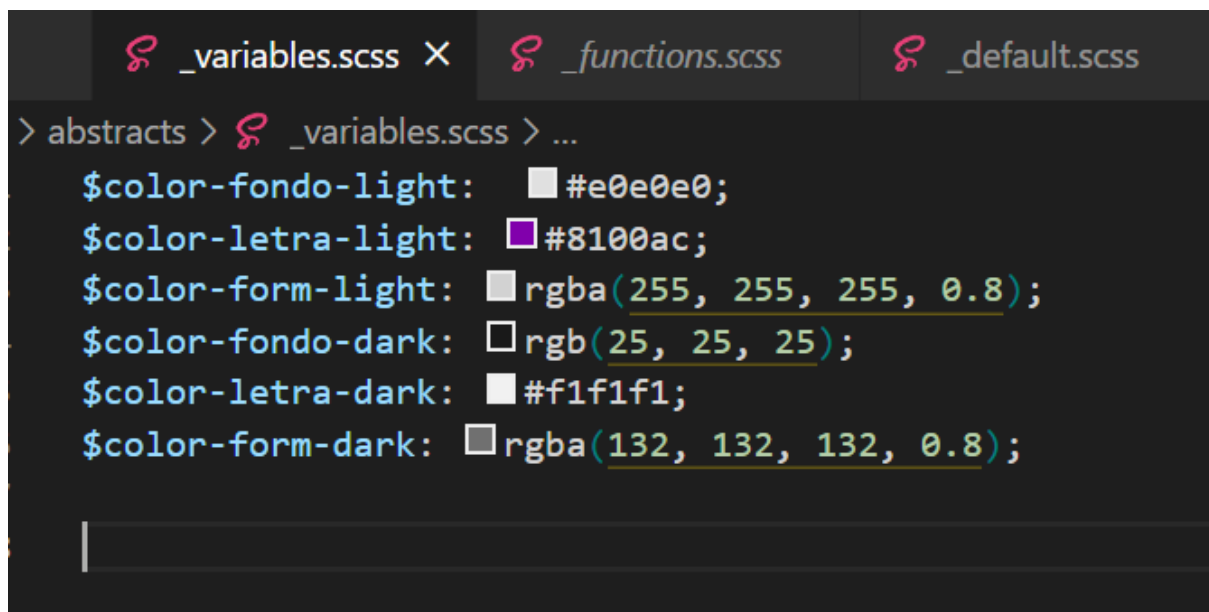
4

4. Desarrolla los estilos del formulario usando el patrón 7-1 y Flexbox.

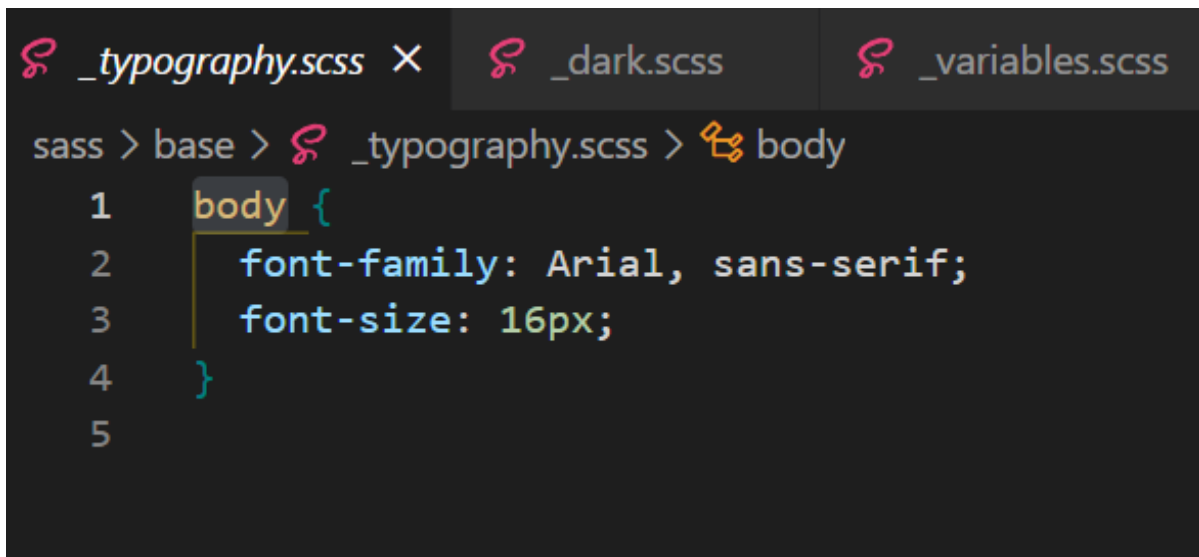
Aquí tenemos la estructura de carpetas del patrón 7-1:



Ahora iremos una por una para añadir los estilos. Vamos a **abstracts/_variables.scss** y creamos las variables que utilizaremos en los demás ficheros scss:



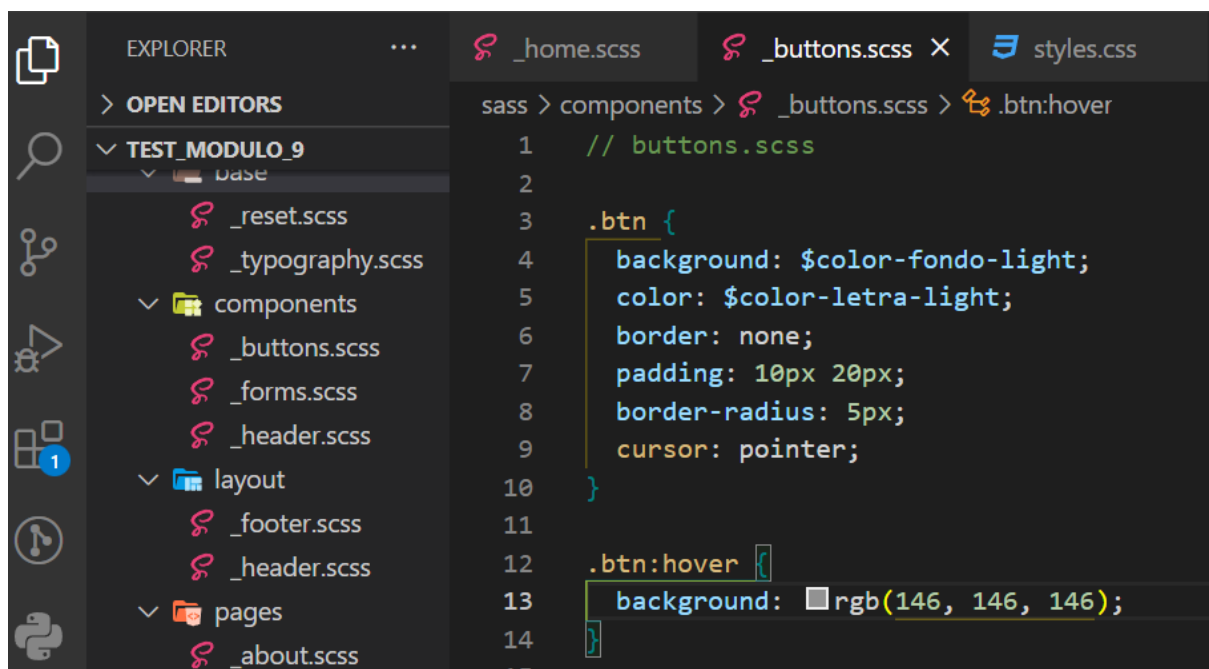
Siguiente: en **base/_typography.scss** ponemos el tipo de letra, tamaño de letra y los demás estilos que queramos para el texto:



```
sass > base > _typography.scss > body

1  body {
2      font-family: Arial, sans-serif;
3      font-size: 16px;
4  }
5
```

Ahora vamos a **components/_buttons.scss** y aquí ponemos todos los estilos de los botones(si tenemos botones):

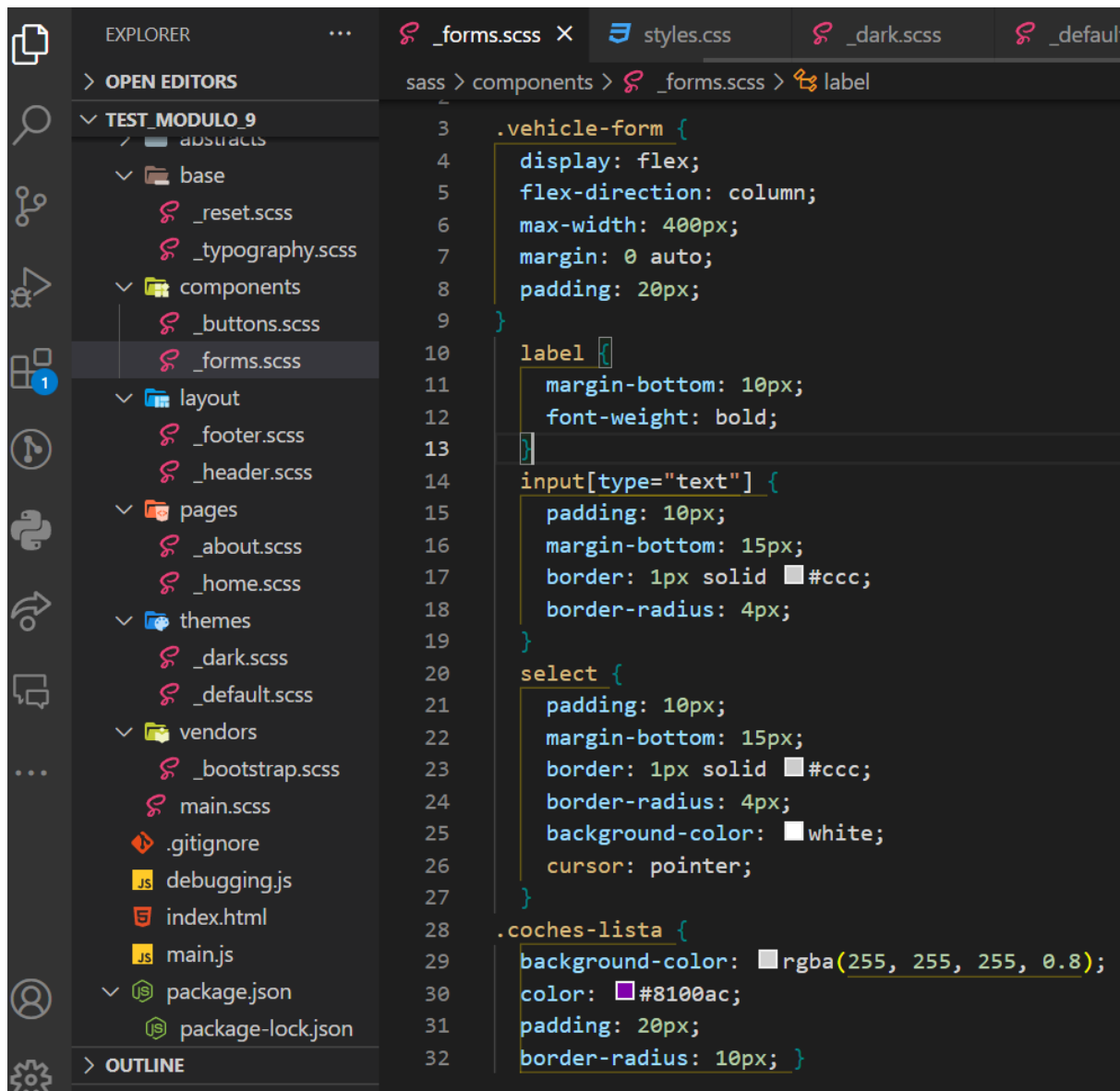


```
EXPLOLER
> OPEN EDITORS
TEST_MODULO_9
  Base
    _reset.scss
    _typography.scss
  components
    _buttons.scss
    _forms.scss
    _header.scss
  layout
    _footer.scss
    _header.scss
  pages
    _about.scss

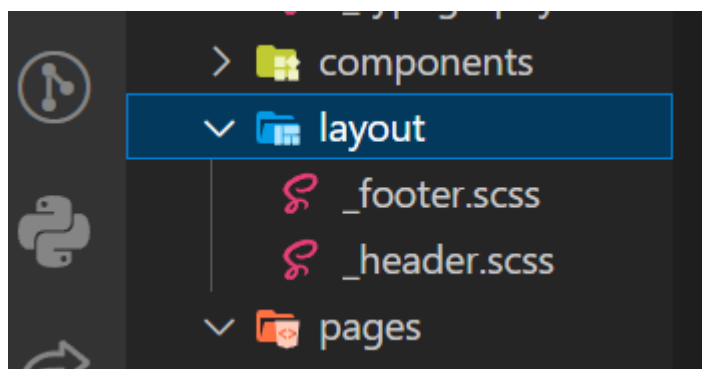
sass > components > _buttons.scss > .btn:hover

1  // buttons.scss
2
3  .btn {
4      background: $color-fondo-light;
5      color: $color-letra-light;
6      border: none;
7      padding: 10px 20px;
8      border-radius: 5px;
9      cursor: pointer;
10 }
11
12 .btn:hover {
13     background: rgb(146, 146, 146);
14 }
15
```

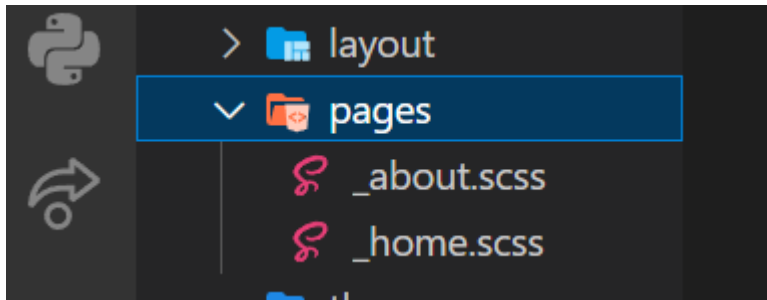
Ahora le damos estilo al formulario y para eso vamos a **components/_forms.scss**:



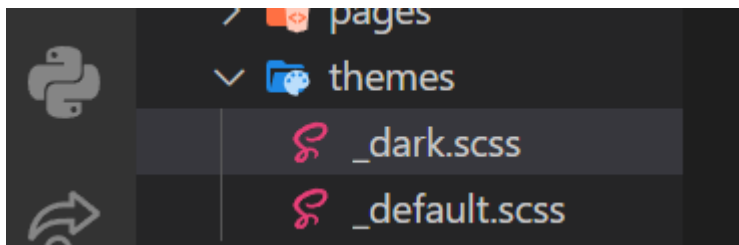
En layout/_footer.scss o header.scss (si tenemos), creamos los estilos de estos.



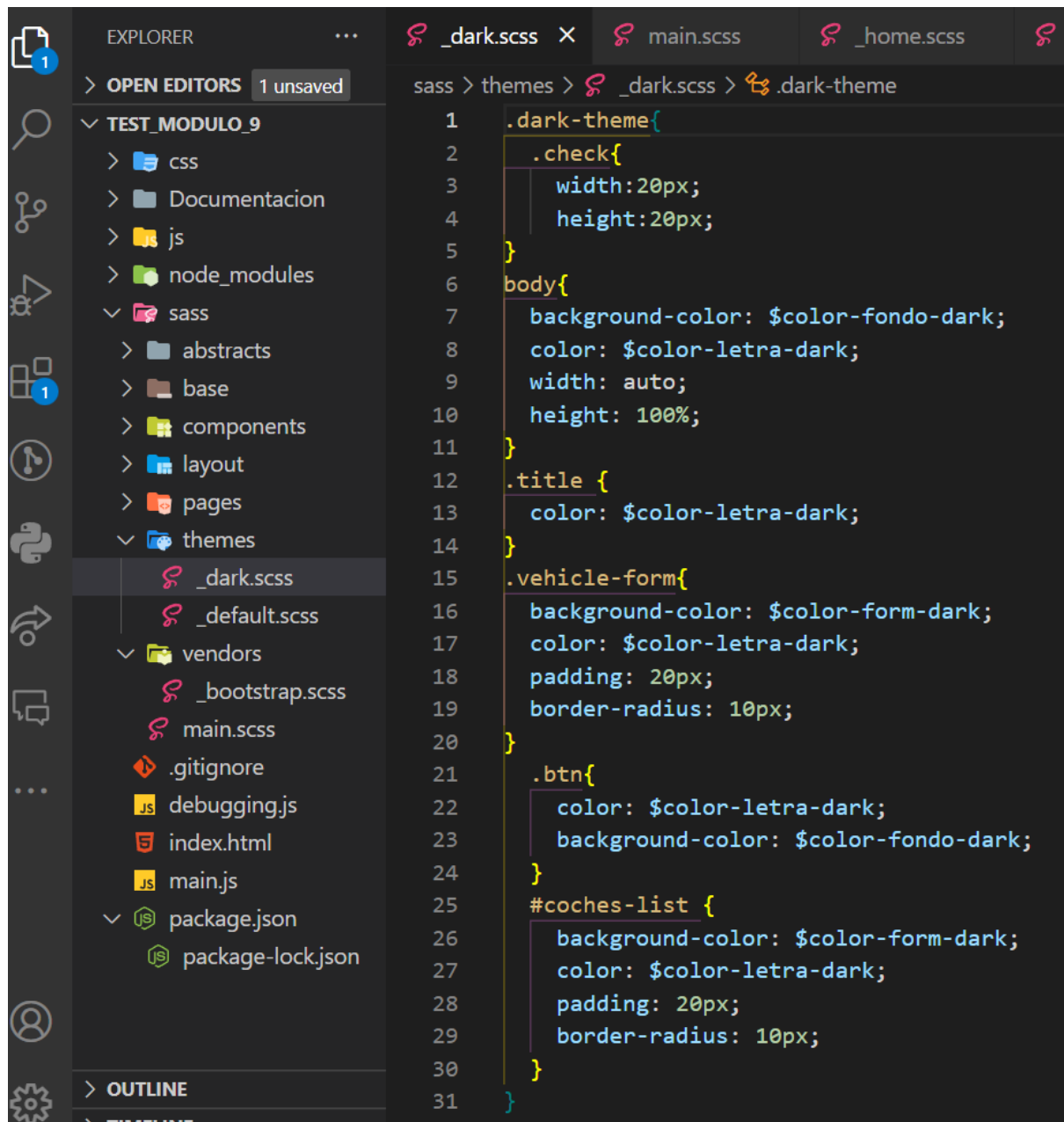
En pages/_about.scss podemos crear los estilos de nuestra página about o home.



En themes podemos hacer los estilos que utilizaremos para cambiar de modo claro/oscuro, lo que vamos a utilizar en el botón check que cambia el modo.



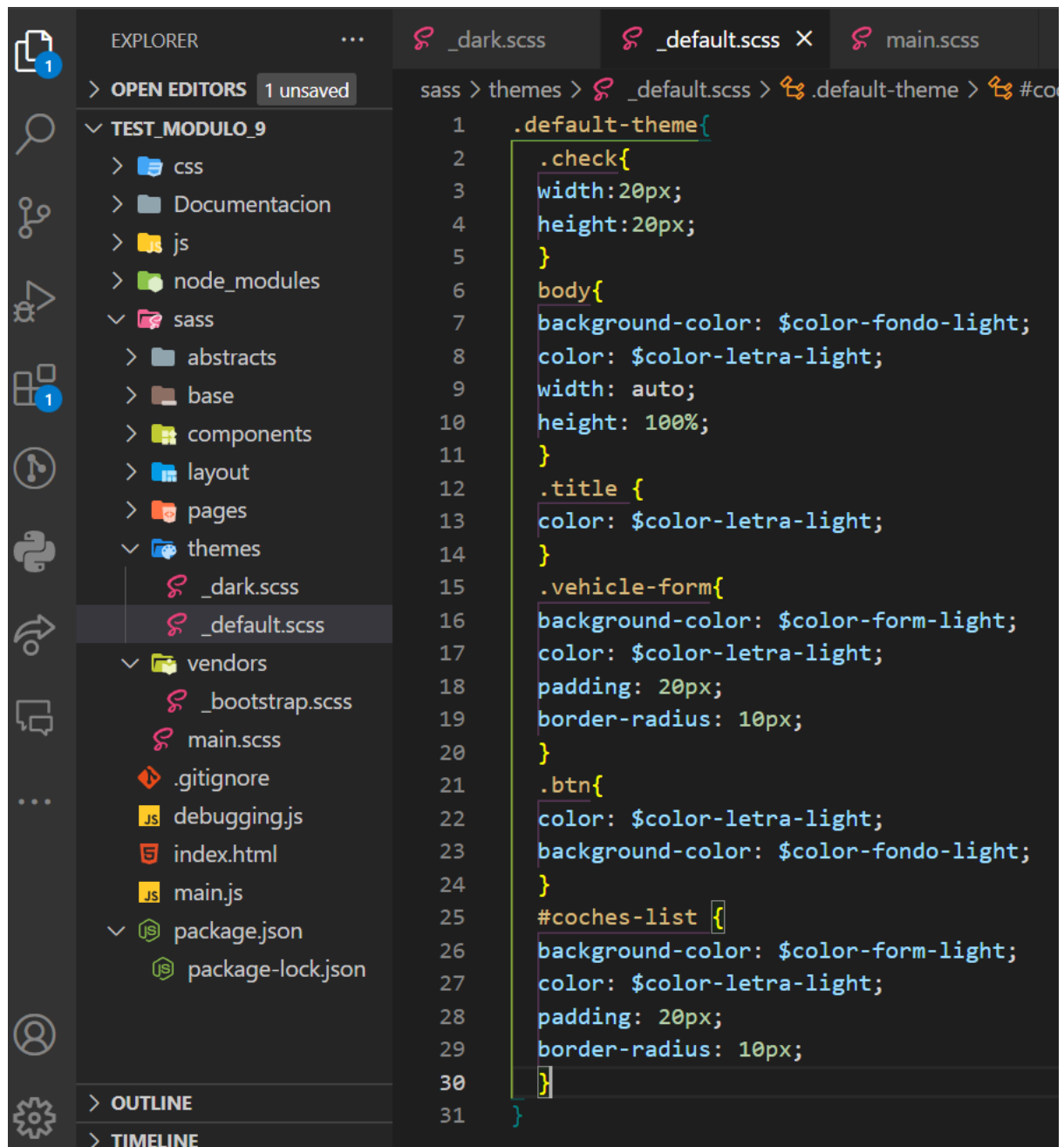
Aquí creo el modo oscuro:



The image shows a screenshot of the Visual Studio Code (VS Code) editor interface. The Explorer sidebar on the left displays a project structure for 'TEST_MODULO_9'. The 'sass' directory is expanded, showing files like '_dark.scss', '_default.scss', and 'main.scss'. The 'themes' subdirectory is also visible. The main editor area shows the content of '_dark.scss', which defines a dark theme. The code includes a '.dark-theme' block, a '.check' class, a 'body' style, a '.title' class, a '.vehicle-form' class, a '.btn' class, and a '#coches-list' class. The code uses Sass variables for colors and dimensions. The breadcrumb at the top of the editor indicates the path: 'sass > themes > _dark.scss > .dark-theme'.

```
1 .dark-theme{
2   .check{
3     width:20px;
4     height:20px;
5   }
6   body{
7     background-color: $color-fondo-dark;
8     color: $color-letra-dark;
9     width: auto;
10    height: 100%;
11  }
12  .title {
13    color: $color-letra-dark;
14  }
15  .vehicle-form{
16    background-color: $color-form-dark;
17    color: $color-letra-dark;
18    padding: 20px;
19    border-radius: 10px;
20  }
21  .btn{
22    color: $color-letra-dark;
23    background-color: $color-fondo-dark;
24  }
25  #coches-list {
26    background-color: $color-form-dark;
27    color: $color-letra-dark;
28    padding: 20px;
29    border-radius: 10px;
30  }
31 }
```

Y el modo default, luego con el .js cambiamos entre ellos:



En main.js creamos la lógica para cambiar el color:

```

46 //cambiamos el color de fondo
47 const themeSwitch = document.getElementById("theme-switch");
48 const body = document.body;
49
50 themeSwitch.addEventListener("change", function () {
51   if (this.checked) {
52     body.classList.add("dark-theme");
53     body.classList.remove("default-theme");
54   } else {
55     body.classList.remove("dark-theme");
56     body.classList.add("default-theme");
57   }
58 });
59
60
61
62

```

En el archivo principal main.scss, importamos todos los estilos:

```

@import 'abstracts/variables';
@import 'base/typography';
@import 'base/reset';
@import 'components/buttons';
@import 'components/forms';
@import 'pages/home';
@import 'themes/dark';
@import 'themes/default';

```

5. Compila el archivo main.scss a un archivo styles.css en el directorio css.

Instalamos node-sass en el proyecto con el siguiente comando:

```
npm install node-sass --save-dev
```

Ahora en el archivo [package.json](#) agregamos un script para compilar Sass:

```
{
```

```
"scripts": {
  "compile-sass": "node-sass sass/main.scss css/styles.css"
}
```

El siguiente comando compila a un archivo CSS, crea un archivo de mapeo que no será utilizado y lanza la terminal para comprobar errores de sintaxis en el archivo .scss y recompilar cuando se produzcan errores.

```
50 padding: 20px; }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

Sass is watching for changes. Press Ctrl-C to stop.
PS C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9> sass --watch sass/main.scss css/style.css
[2023-08-28 20:22] Compiled sass/main.scss to css/style.css.
Sass is watching for changes. Press Ctrl-C to stop.
```

Ejecutamos el script utilizando el siguiente comando en la terminal:

`npm run compile-sass`

Esto genera lo siguiente en la terminal:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL powershell +v [ ] [ ] ... ^ X

PS C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9> npm run compile-sass

> test_modulo_9@1.0.0 compile-sass
> node-sass sass/main.scss css/styles.css

Rendering Complete, saving .css file...
Wrote CSS to C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9\css\styles.css
PS C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9>
```

6. Crea el archivo index.html con los elementos para el formulario. Creamos el index.html y dentro añadimos el CDN de bootstrap.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width,
initial-scale=1.0">

<!-- cdn css -->

<link rel="stylesheet" href="
https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css
">
  <link rel="stylesheet" href="css/styles.css">
  <title>Formulario de Datos de Vehículo</title>
</head>
<body>
  <h1 class="text-center mt-5">Formulario de Datos de Vehículo</h1>

  <form class="vehicle-form">
    <label for="brand">Marca:</label>
    <input type="text" id="brand" name="brand" required>

    <label for="model">Modelo:</label>
    <input type="text" id="model" name="model" required>

    <label for="year">Año:</label>
    <input type="date" class="mb-3" id="year" name="year" required>

    <label for="color">Color:</label>
    <input type="text" id="color" name="color" required>

    <label for="price">Precio:</label>
    <input type="number" class="mb-3" id="price" name="price"
step="0.01" required>

    <button type="submit">Enviar</button>
  </form>

  <!-- cdn js -->

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bun
dle.min.js"></script>
</body>
</html>
```

7. Implementa la lógica JavaScript para que, al pulsar en un botón de "Enviar", los datos del formulario se almacenen en un objeto.

Implementamos la lógica js y almacenamos los datos en un objeto llamado coches:

```
document.addEventListener("DOMContentLoaded", function() {
  const form = document.getElementById("vehicle-form");
  const coches = []; // Aquí almacenaremos los datos de los coches
  const cochesListElement = document.getElementById("coches-list"); // Elemento donde se mostrarán los datos

  form.addEventListener("submit", function(event) {
    event.preventDefault();

    const brand = document.getElementById("brand").value;
    const model = document.getElementById("model").value;
    const year = parseInt(document.getElementById("year").value);
    const color = document.getElementById("color").value;
    const price = parseFloat(document.getElementById("price").value);

    const vehicleData = {
      brand: brand,
      model: model,
      year: year,
      color: color,
      price: price
    };

    coches.push(vehicleData); // Agregamos el objeto al array de coches
    console.log(coches); // el array de coches en la consola

    // Mostramos los datos almacenados en el div coches-list
    cochesListElement.innerHTML = "";
    coches.forEach(coche => {
      cochesListElement.innerHTML += `
        <div>
          <p>Marca: ${coche.brand}</p>
          <p>Modelo: ${coche.model}</p>
          <p>Año: ${coche.year}</p>
          <p>Color: ${coche.color}</p>
          <p>Precio: ${coche.price}</p>
          <hr>
        </div>
      `;
    });
  });
});
```

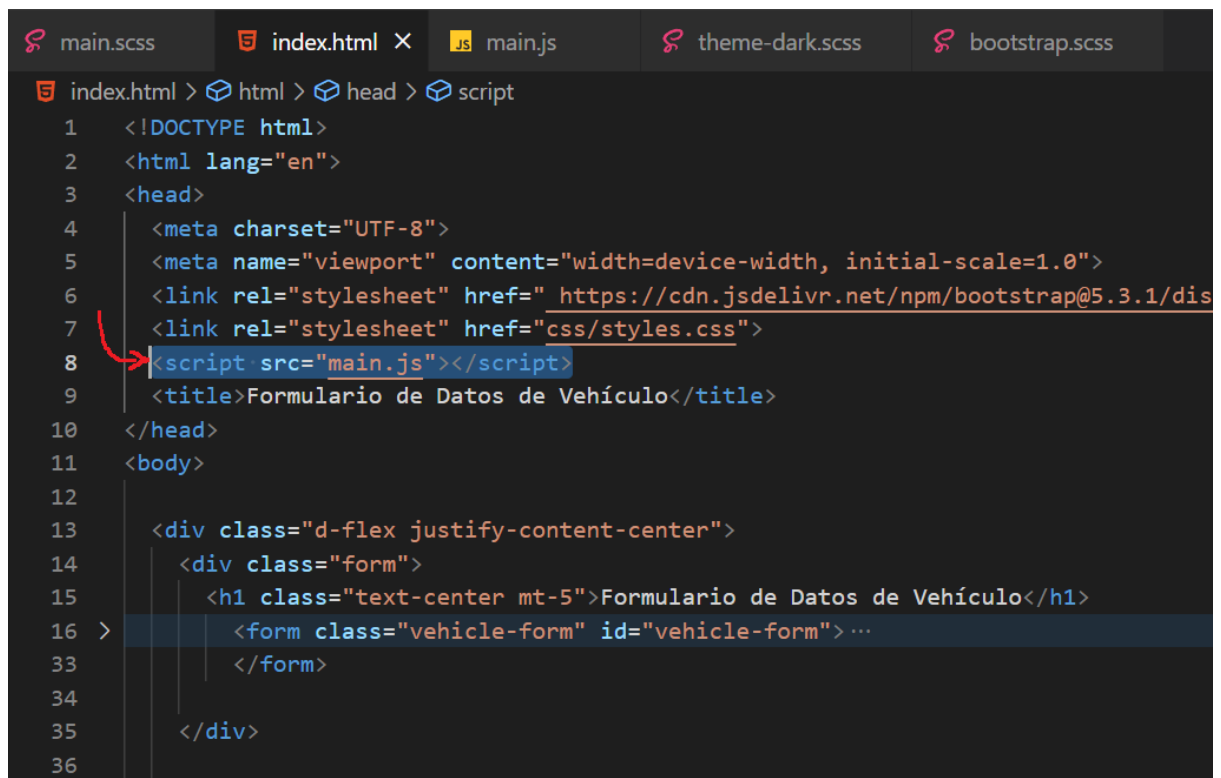
```

    });

    // Limpiamos el formulario después de enviar los datos
    form.reset();
  });
});

```

8. Enlaza el archivo app.js con index.html y ejecuta el proyecto en el navegador.



The screenshot shows a code editor with several tabs: main.scss, index.html (active), main.js, theme-dark.scss, and bootstrap.scss. The index.html file is open, showing the following code:

```

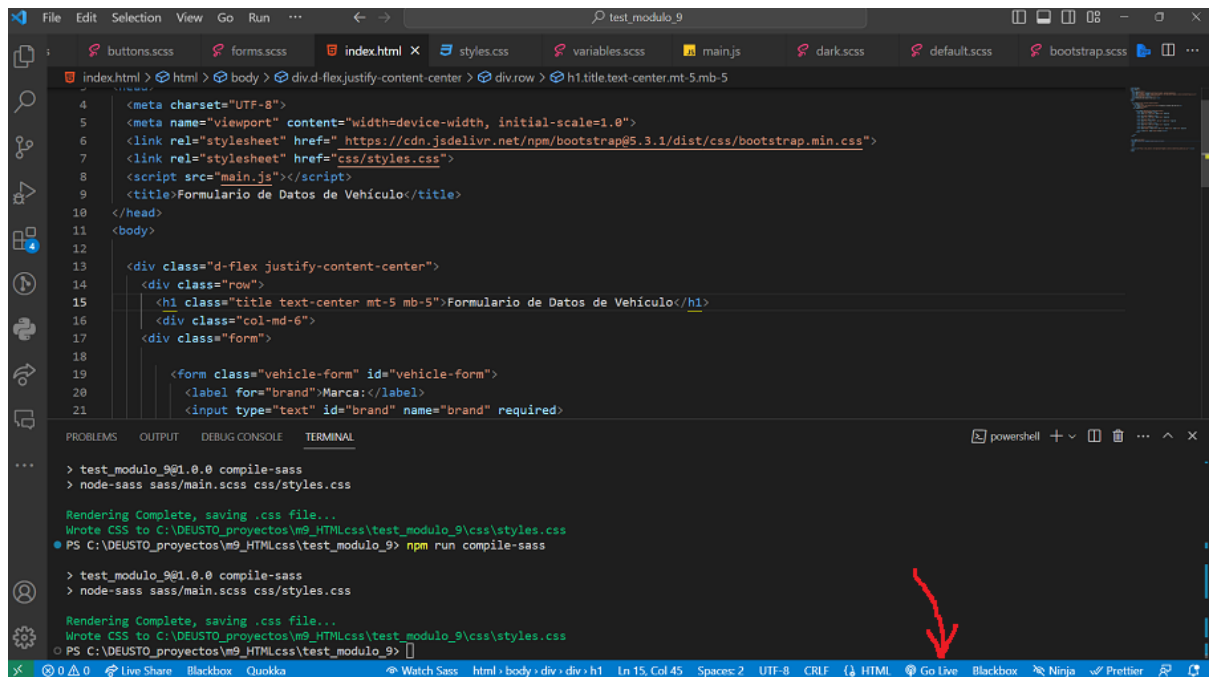
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css">
7    <link rel="stylesheet" href="css/styles.css">
8    <script src="main.js"></script>
9    <title>Formulario de Datos de Vehículo</title>
10 </head>
11 <body>
12
13   <div class="d-flex justify-content-center">
14     <div class="form">
15       <h1 class="text-center mt-5">Formulario de Datos de Vehículo</h1>
16       <form class="vehicle-form" id="vehicle-form"> ...
33     </form>
34   </div>
35
36

```

A red arrow points to the script tag on line 8, which is highlighted in blue. The breadcrumb navigation at the top of the editor shows the path: index.html > html > head > script.

9. Utiliza las herramientas de desarrollador para obtener diferentes valores por depuración.

Utilizamos Live Server para abrir la app en la web:



The image shows a VS Code editor window with a file explorer on the left. The editor is open to `index.html`, which contains the following code:

```
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css">
7 <link rel="stylesheet" href="css/styles.css">
8 <script src="main.js"></script>
9 <title>Formulario de Datos de Vehículo</title>
10 </head>
11 <body>
12
13 <div class="d-flex justify-content-center">
14 <div class="row">
15 <h1 class="title text-center mt-5 mb-5">Formulario de Datos de Vehículo</h1>
16 <div class="col-md-6">
17 <div class="form">
18
19 <form class="vehicle-form" id="vehicle-form">
20 <label for="brand">Marca:</label>
21 <input type="text" id="brand" name="brand" required>
```

The terminal at the bottom shows the following commands and output:

```
> test_modulo_9@1.0.0 compile-sass
> node-sass sass/main.scss css/styles.css

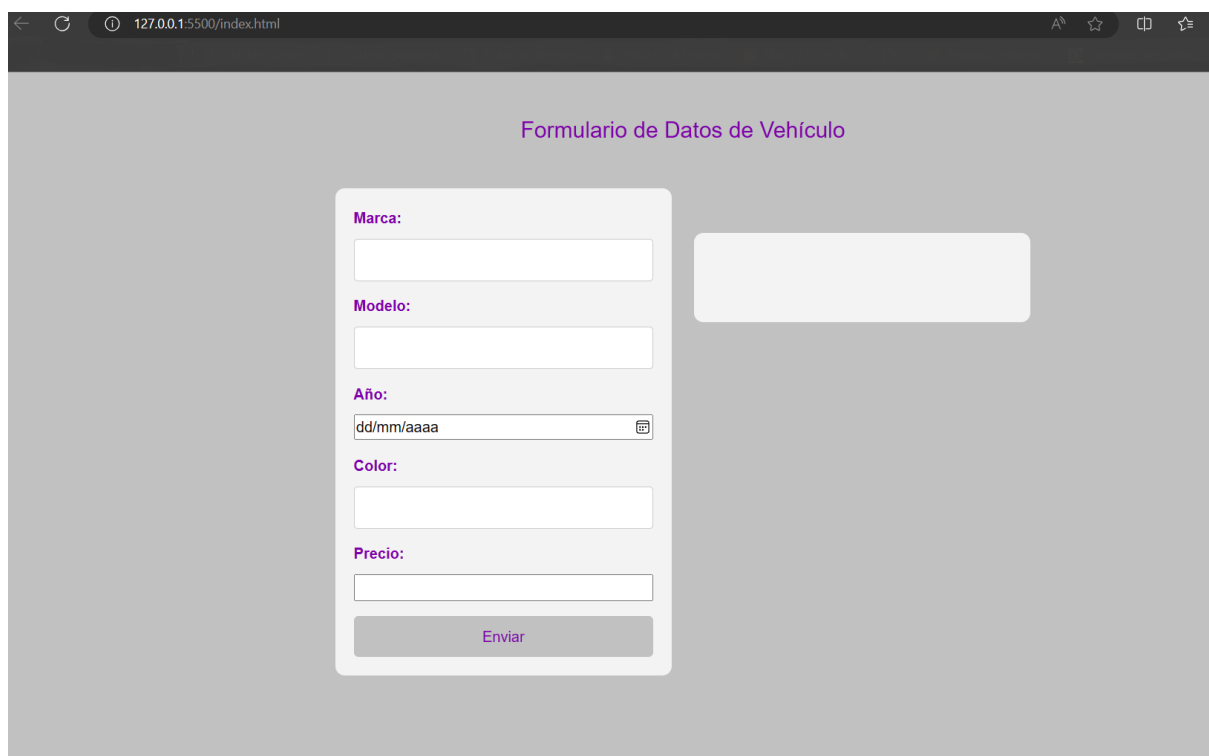
Rendering Complete, saving .css file...
Wrote CSS to C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9\css\styles.css
PS C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9> npm run compile-sass

> test_modulo_9@1.0.0 compile-sass
> node-sass sass/main.scss css/styles.css

Rendering Complete, saving .css file...
Wrote CSS to C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9\css\styles.css
PS C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9>
```

A red arrow points to the terminal output.

Aqui esta la pagina:



Ejecutamos en la terminal:

`npm run compile-sass`

Y comprobamos que no hay errores, luego miramos en la pestaña problems (como se ve en la imagen) para comprobar que no hay errores

```
6 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap"
7 <link rel="stylesheet" href="css/styles.css">
8 <script src="main.js"></script>
9 <title>Formulario de Datos de Vehículo</title>
10 </head>
11 <body>
12
13 <div class="d-flex justify-content-center">
14 <div class="row">
15 <h1 class="title text-center mt-5 mb-5">Formulario de Datos de Vehículo</h1>
16 <div class="col-md-6">
17 <div class="form">
18
19 <form class="vehicle-form" id="vehicle-form">
20 <label for="brand">Marca:</label>
21 <input type="text" id="brand" name="brand" required>

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL**

```
> test_modulo_9@1.0.0 compile-sass
> node-sass sass/main.scss css/styles.css

Rendering Complete, saving .css file...
Wrote CSS to C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9\css\styles.css
● PS C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9> npm run compile-sass

> test_modulo_9@1.0.0 compile-sass
> node-sass sass/main.scss css/styles.css

Rendering Complete, saving .css file...
Wrote CSS to C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9\css\styles.css
○ PS C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9>

```

También podemos ejecutar en la terminal el comando:

`sass --watch sass/main.scss css/style.css`

que inicia el compilador Sass en modo de observación, lo que significa que estará atento a los cambios en el archivo main.scss y generará automáticamente un archivo style.css actualizado en la carpeta css cada vez que realicemos cambios en los estilos Sass:

```
15 }
16 .vehicle-form{
17   background-color: $color-form-dark;
18   color: $color-letra-dark;
19   padding: 20px;
20   border-radius: 10px;
21 }
22

```

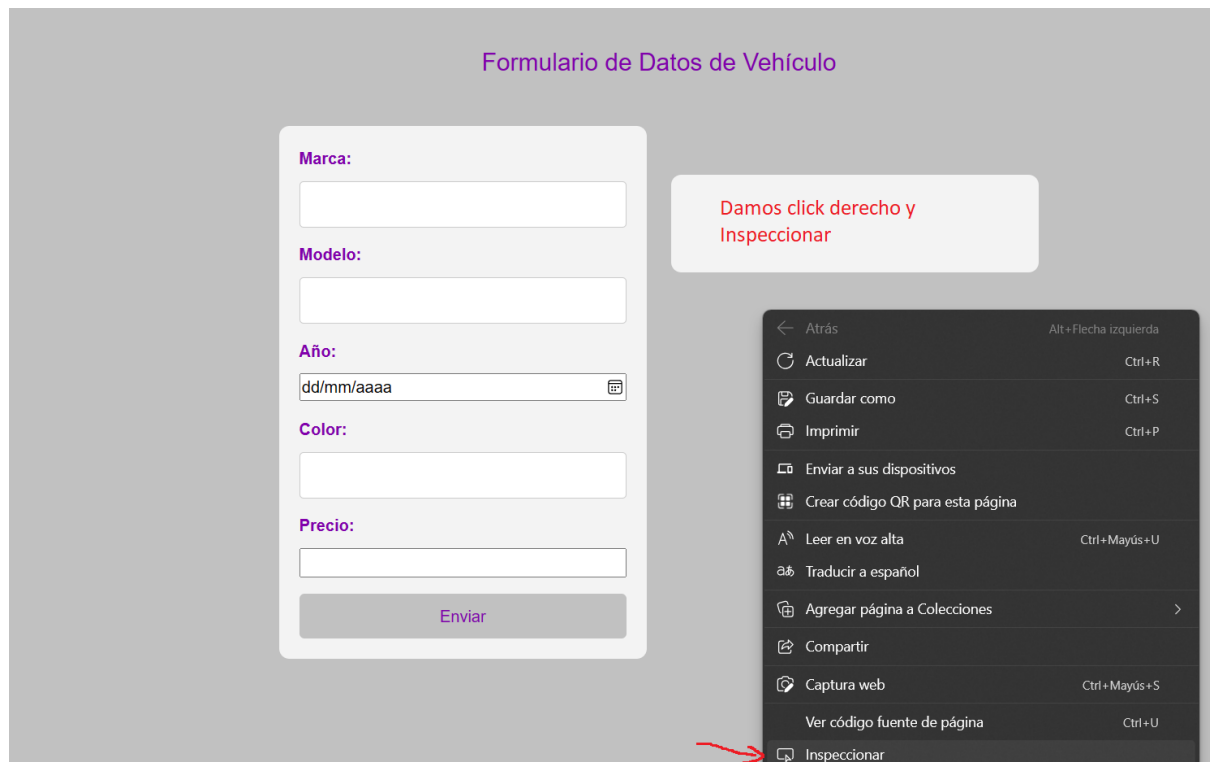
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL**

```
PS C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9> sass --watch sass/main.scss css/style.css
Sass is watching for changes. Press Ctrl-C to stop.

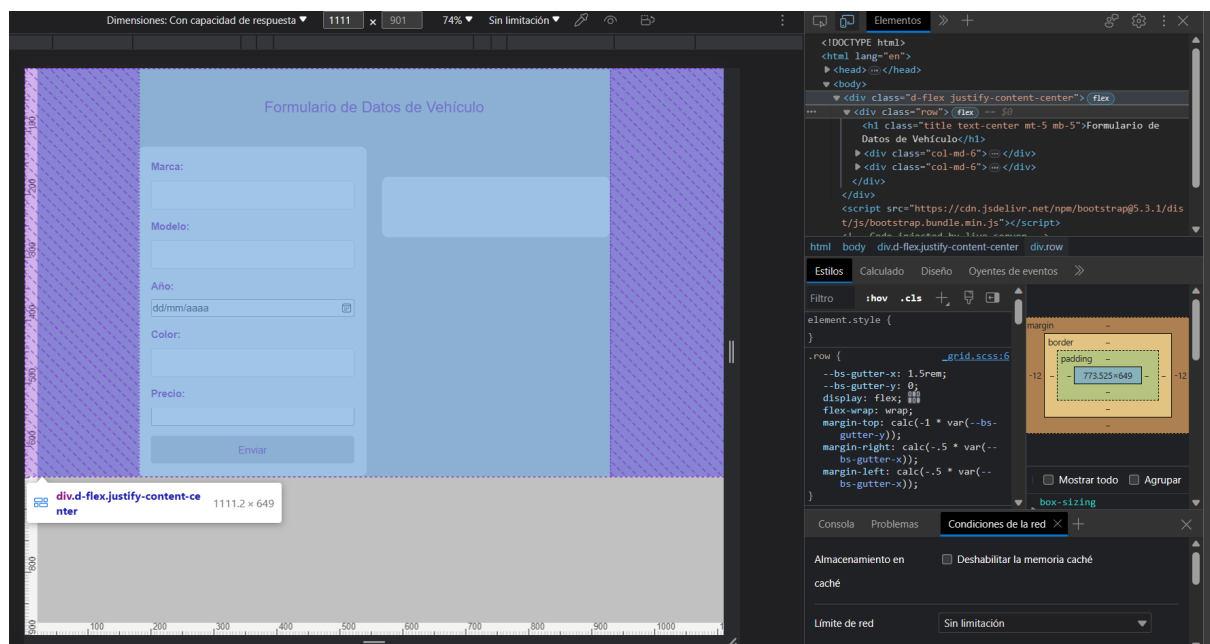
[2023-08-29 16:07] Compiled sass/main.scss to css/style.css.
[2023-08-29 16:07] Compiled sass/main.scss to css/style.css.

```

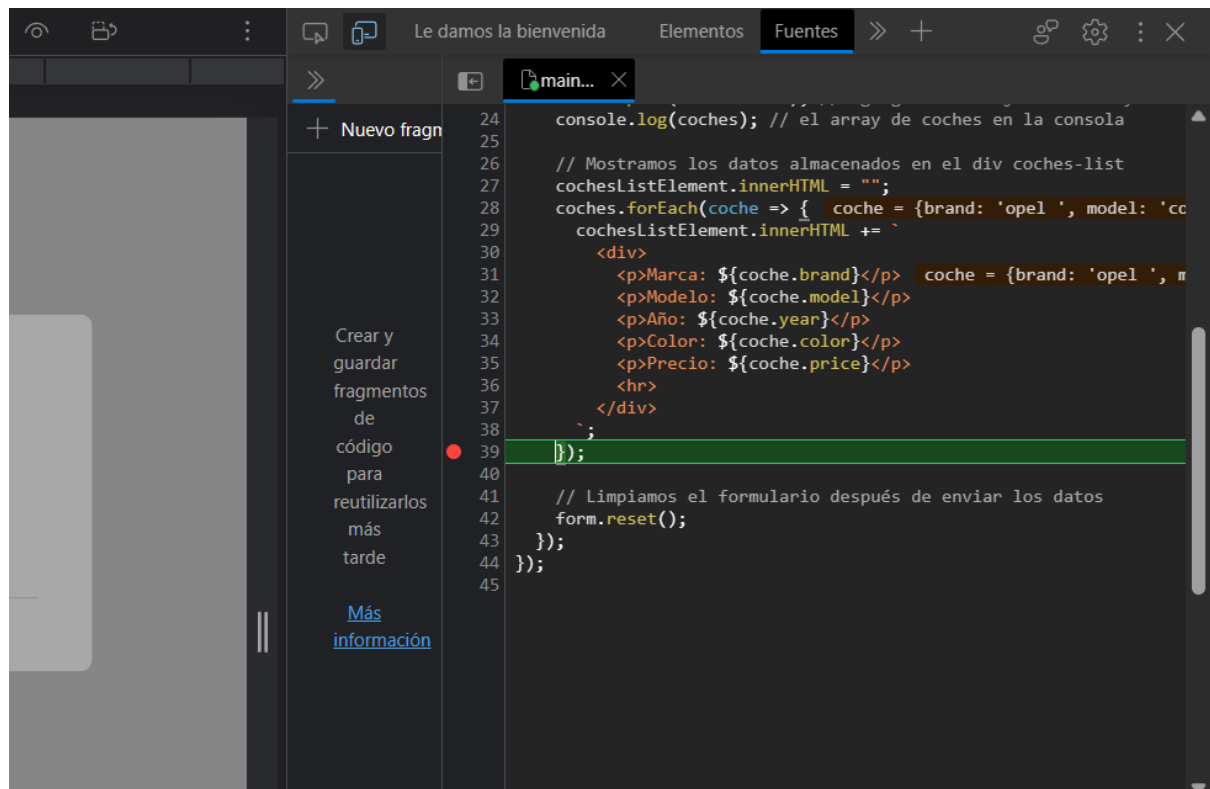
Vamos a la página web y inspeccionamos:



Aquí podemos ver los estilos y el html:



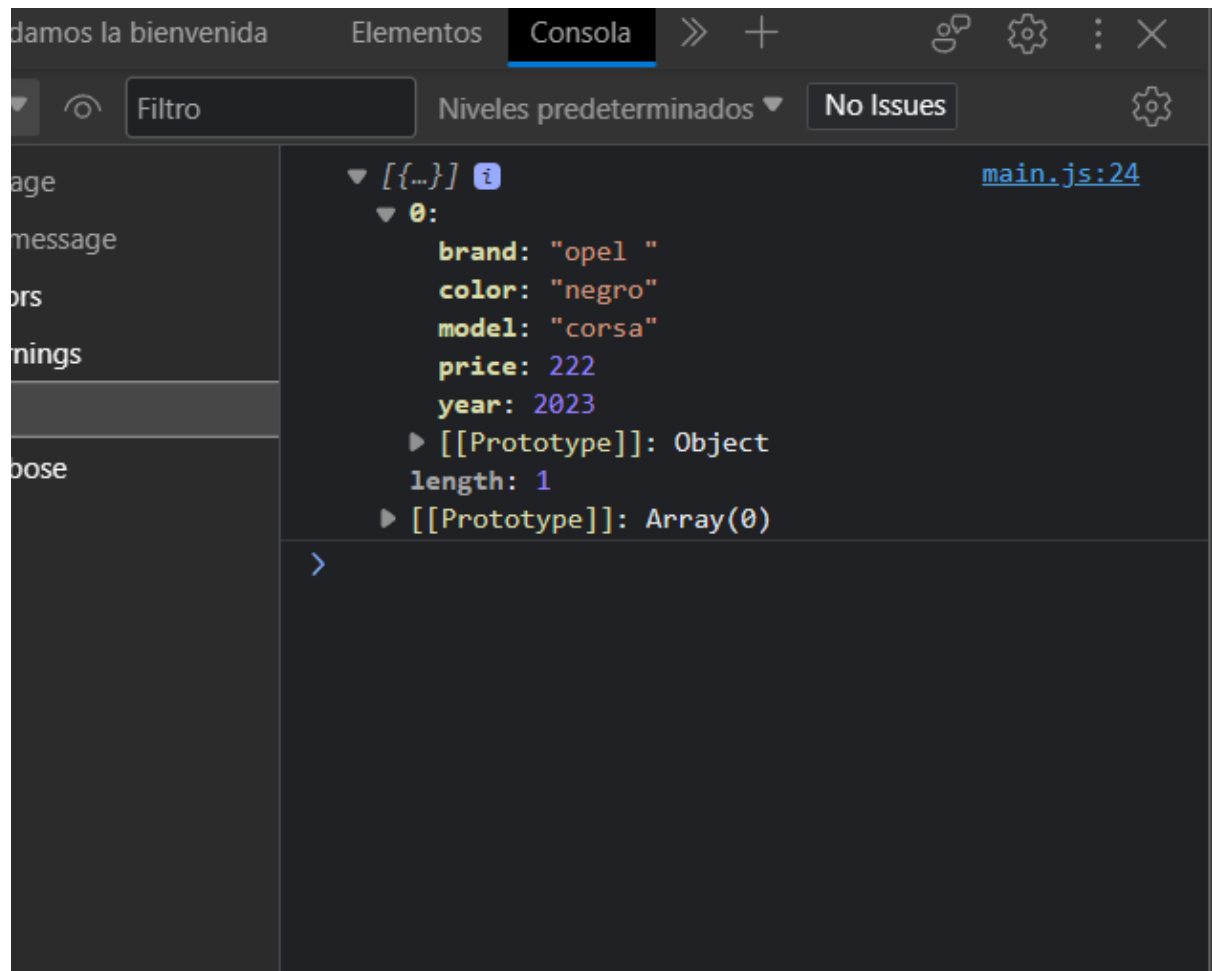
Aquí podemos ver la depuración:



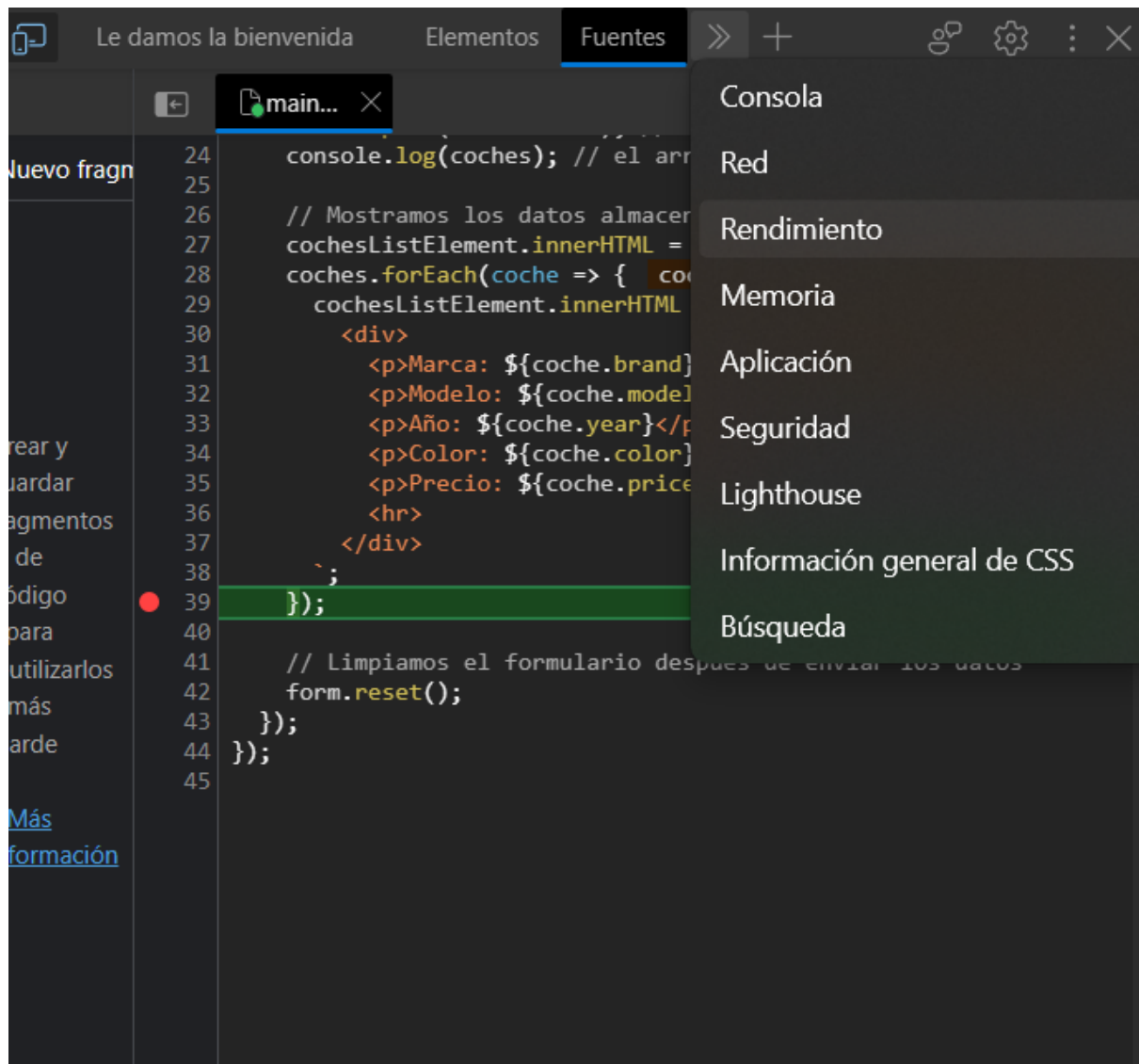
The image shows a web browser window with the developer tools open to the 'Fuentes' (Sources) tab. The file 'main...' is open, displaying JavaScript code. The code logs an array of cars to the console and then iterates over it to populate a list. The list items are HTML fragments containing car details like brand, model, year, color, and price. The code is as follows:

```
24 console.log(coches); // el array de coches en la consola
25
26 // Mostramos los datos almacenados en el div coches-list
27 cochesListElement.innerHTML = "";
28 coches.forEach(coche => { coche = {brand: 'opel ', model: 'cc
29 cochesListElement.innerHTML += `
30 <div>
31 <p>Marca: ${coche.brand}</p> coche = {brand: 'opel ', m
32 <p>Modelo: ${coche.model}</p>
33 <p>Año: ${coche.year}</p>
34 <p>Color: ${coche.color}</p>
35 <p>Precio: ${coche.price}</p>
36 <hr>
37 </div>
38 `;
39 });
40
41 // Limpiamos el formulario después de enviar los datos
42 form.reset();
43 });
44 });
45
```

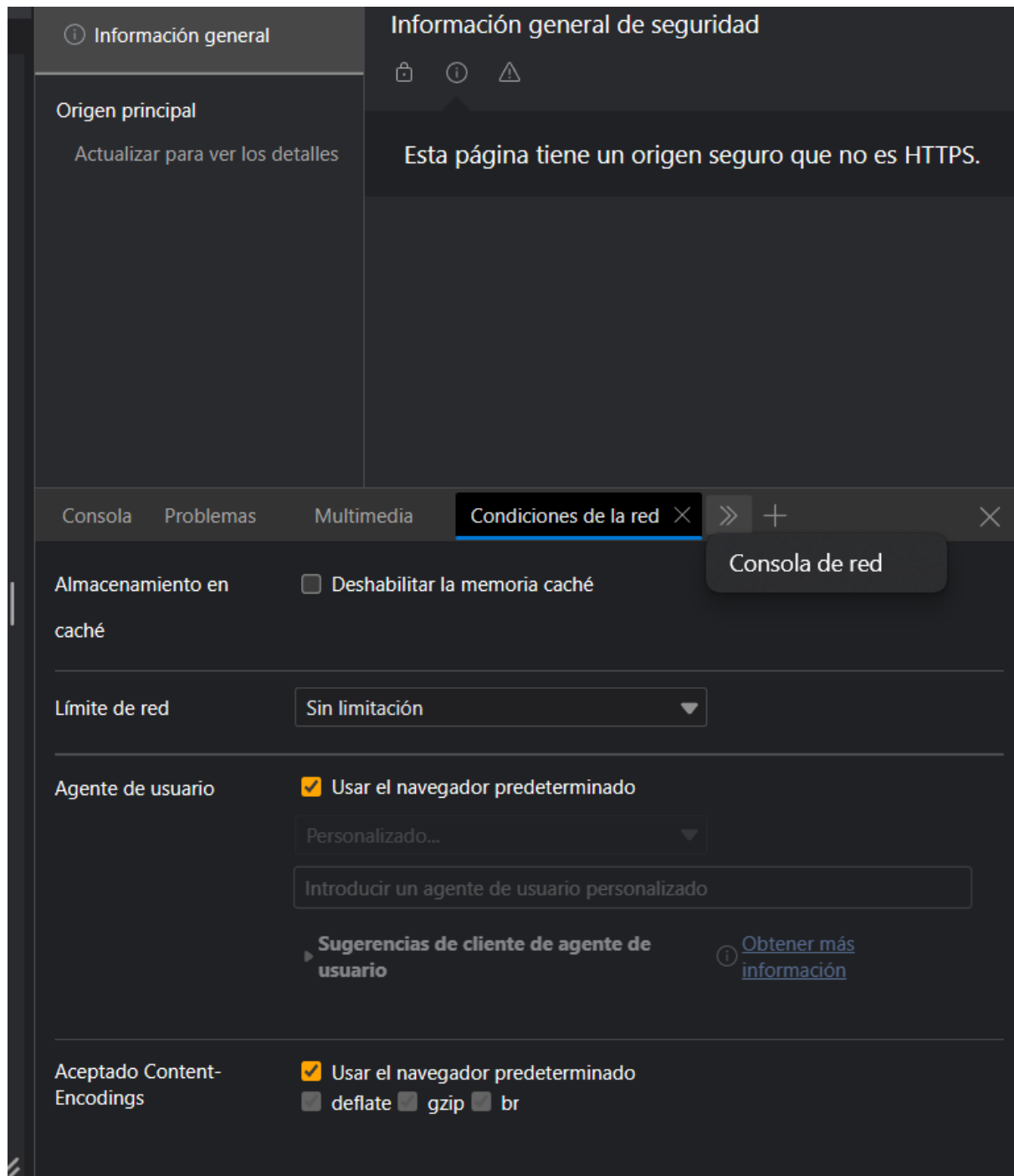
A continuación vemos en consola los valores del objeto donde almacenamos la información:



Aquí vemos varias opciones para analizar nuestra aplicación, como información de red, detalles de memoria, estilos y demás:

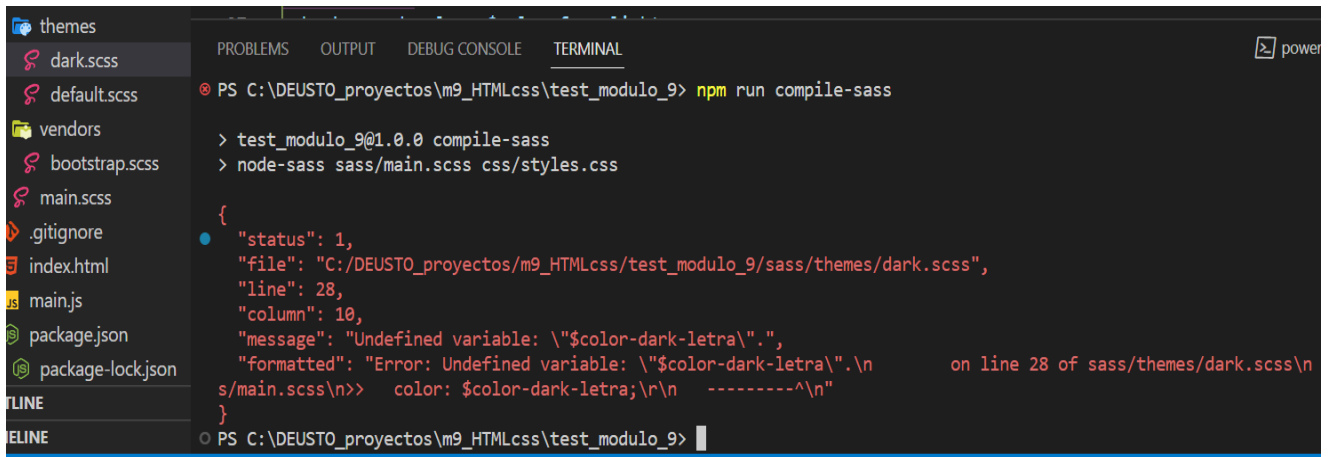


Aquí podemos ver otro tipo de valores como condiciones de red o problemas:



Si tenemos errores cuando ejecutamos:
`npm run compile-sass`

Aparece algo como lo siguiente:

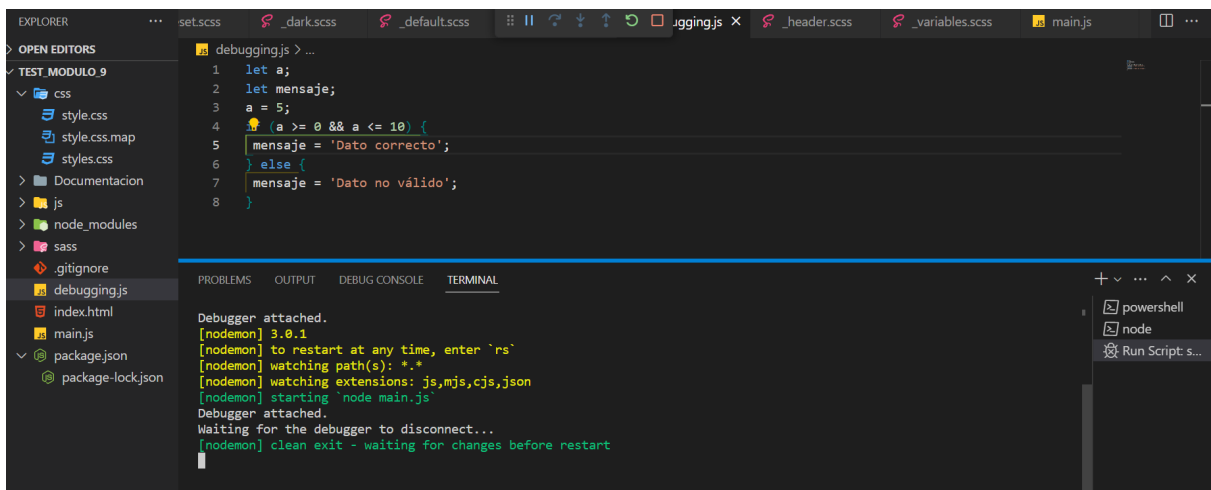


```
PS C:\DEUSTO_proyectos\m9_HTMLcss\test_modulo_9> npm run compile-sass

> test_modulo_9@1.0.0 compile-sass
> node-sass sass/main.scss css/styles.css

{
  "status": 1,
  "file": "C:/DEUSTO_proyectos/m9_HTMLcss/test_modulo_9/sass/themes/dark.scss",
  "line": 28,
  "column": 10,
  "message": "Undefined variable: \"$color-dark-letra\".",
  "formatted": "Error: Undefined variable: \"$color-dark-letra\".\n          on line 28 of sass/themes/dark.scss\n          s/main.scss\n>>   color: $color-dark-letra;\r\n          -----^\\n"
}
```

A continuación utilizamos el debugging de vs code pulsando F5:



```
1 let a;
2 let mensaje;
3 a = 5;
4 if (a >= 0 && a <= 10) {
5   mensaje = 'Dato correcto';
6 } else {
7   mensaje = 'Dato no válido';
8 }
```

```
Debugger attached.
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node main.js`
Debugger attached.
Waiting for the debugger to disconnect...
[nodemon] clean exit - waiting for changes before restart
```

Aquí es cómo queda la pagina:

El modo claro:



Formulario de Datos de Vehículo

Marca:

Modelo:

Año:

Color:

Precio:

Dark ☐

El modo oscuro:

Formulario de Datos de Vehículo

Marca:

Modelo:

Año:

Color:

Precio:

Enviar

Dark ☒

Completamos el formulario y le damos enviar:

Formulario de Datos de Vehículo

Marca:

opel

Modelo:

corsa

Año:

08/08/2023



Color:

negro

Precio:



Rellene este campo.

Dark ☐

Los datos enviados aparecen en la parte derecha y el formulario se limpia:

Formulario de Datos de Vehículo

Marca:

Modelo:

Año:

Color:

Precio:

Enviar

Dark ☐

Marca: opel

Modelo: corsa

Año: 2023

Color: negro

Precio: 2200

Ahora lo subimos a github:

```
git init
git add .
git commit -m "First commit"
```

Ahora creamos un repositorio en github llamado modulo9_sass y luego ponemos esto en la terminal:

```
git remote add origin https://github.com/geannyna/modulo9\_sass.git
git branch -M main
git push -u origin main
```