

EXERCISES QUE TRABAJAREMOS EN LA CUE

- EXERCISE 1: HOLA MUNDO CON JAVASCRIPT
- EXERCISE 2: ESCRIBIENDO CÓDIGO CON JAVASCRIPT
- EXERCISE 3: REALIZANDO OPERACIONES MATEMÁTICAS
- EXERCISE 4: FUNCIONES Y OPERACIONES ARITMETICAS
- EXERCISE 5: TRABAJANDO CON CONDICIONALES EN JAVASCRIPT

EXERCISE 1: HOLA MUNDO CON JAVASCRIPT

El lenguaje de **FrontEnd** por excelencia, altamente utilizado y muy demandado, es el que nos permitirá comenzar a dar dinamismo a nuestras páginas web.

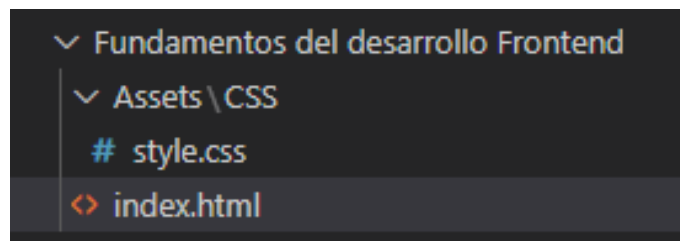
Cuando comenzamos a trabajar con **JavaScript** tenemos tres formas de incorporarlo a nuestro proyecto, dos de ellas incluyen el código directamente en nuestro archivo **HTML**, haciendo que se vuelva más compleja su mantención y que nuestro código sea más difícil de leer.

La tercera forma incluye un archivo externo enlazado, tal como lo hemos realizado antes con **CSS**.

1.- PRIMERA FORMA DE INCORPORAR JAVASCRIPT.

Para comenzar, debemos tener un proyecto con el archivo **index.html**, al cual le incorporaremos la estructura básica, enlazaremos **Bootstrap** y enlazaremos un archivo **CSS**, por lo que crearemos una carpeta **Assets** que contendrá la carpeta **CSS** y en ella, el archivo **style.css**

La estructura de carpetas es la siguiente:



Ahora, dentro del `<head>` incorporaremos las etiquetas `<script>` y dentro de ellas podremos escribir nuestro código **JavaScript**.

Comenzaremos con un típico “Hola Mundo” que imprimiremos por consola.

Dentro de las etiquetas `<script>` escribiremos la indicación utilizando `console.log` y dentro de un paréntesis, utilizando comillas, escribiremos “Hola Mundo”.

El código completo de nuestro archivo **index.html** es el siguiente:

```
<!doctype html>
<html lang="es">

<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <!-- Bootstrap CSS -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"
  integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOMLAsjC"
crossorigin="anonymous">

  <link rel="stylesheet" href="Assets/CSS/style.css">
  <title>Bootstrap 2</title>
  <script>

    console.log("Hola Mundo");

  </script>
</head>

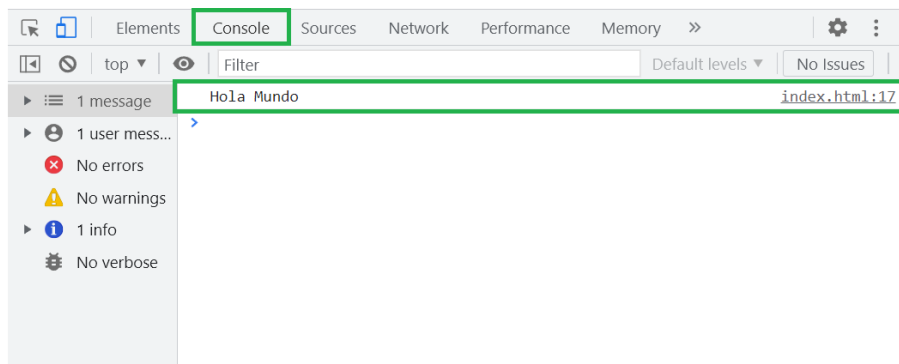
<body>

  <!-- Option 1: Bootstrap Bundle with Popper -->
```

```
<script  
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"  
  integrity="sha384-MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"  
  crossorigin="anonymous"></script>  
  
</body>  
  
</html>
```

Para poder ver el resultado de nuestro primer código **JavaScript**, abriremos el proyecto en nuestro navegador y no encontraremos nada.

Utilizando el botón secundario, ingresaremos al inspector de elementos y ahí nos dirigiremos a la consola.



De esta forma hemos obtenido la impresión en pantalla (es decir, podemos visualizar) el código que hemos escrito con **JavaScript** y hemos conocido la primera forma de utilizar **JavaScript** en nuestros proyectos.

Podemos observar que, al tener un código tan pequeño como un “Hola Mundo”, nuestro archivo **index.html** no es complejo, pero cuando programamos, en múltiples ocasiones, tendremos códigos muy largos. Es en esos casos que se hará complejo esta forma de uso de **JavaScript**.

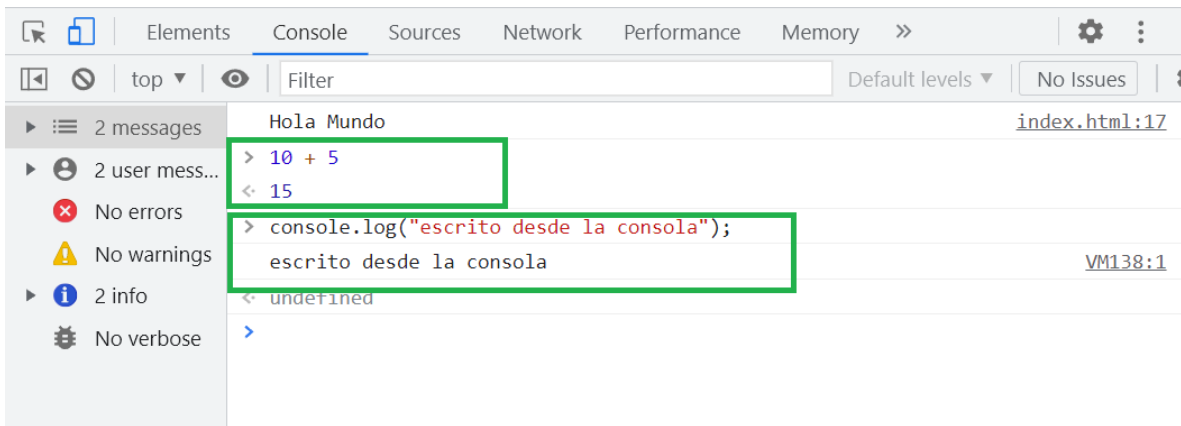
Antes de continuar con nuestros ejemplos nos preguntamos exactamente por la consola.

QUÉ ES LA CONSOLA DE JAVASCRIPT

La consola de **JavaScript** o, más bien dicho, la consola del **Navegador** es una consola web que podemos encontrar en todos los navegadores (ingresando al inspector de elementos).

En ella podemos encontrar solicitudes de red, código **JavaScript** (como el del ejemplo que hemos realizado), errores y advertencias de seguridad.

Además de eso, al utilizar la consola, podemos ejecutar instrucciones **JavaScript** escritas directamente en ella o, por ejemplo, realizar operaciones matemáticas.



Continuando con nuestro ejemplo de **JavaScript**, borraremos el código que se encuentra dentro de las etiquetas `<script>` y borraremos las etiquetas `<script>` que se encuentran en el `<head>`.

2.- SEGUNDA FORMA DE INCORPORAR CÓDIGO JAVASCRIPT

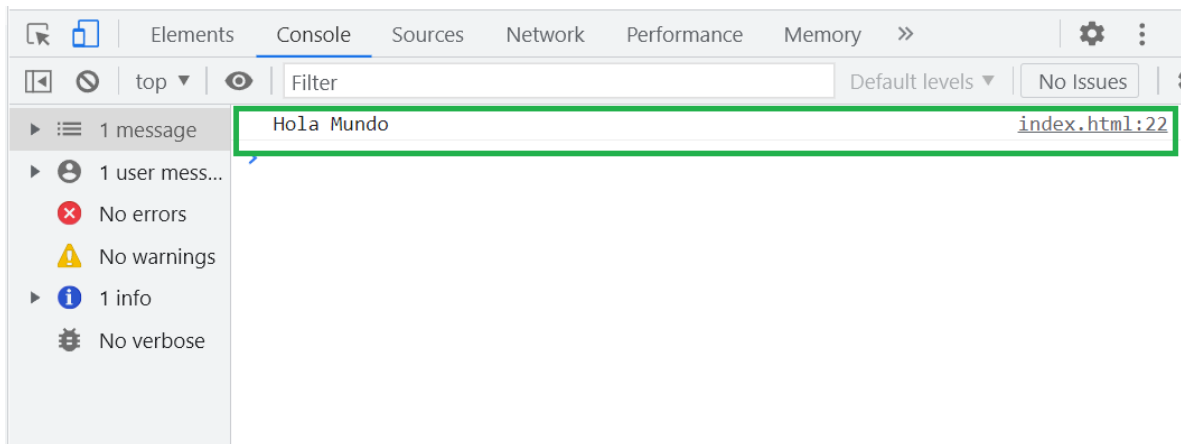
Ahora, dentro de las etiquetas `<body>`, antes de la etiqueta `<script>` con el enlace copiado de **Bootstrap**, incorporaremos otras etiquetas `<script>` y entre ellas, nuevamente un `console.log` con el mensaje "hola mundo".

El código es el siguiente:

```
1 <!doctype html>
2 <html lang="es">
3
4 <head>
5   <!-- Required meta tags -->
6   <meta charset="utf-8">
7   <meta name="viewport" content="width=device-width, initial-
8 scale=1">
9
10  <!-- Bootstrap CSS -->
11  <link
12 href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap
13 .min.css" rel="stylesheet"
14   integrity="sha384-
15 EVSTQN3/azprG1Anm3QDgpgJLIIm9Nao0Yz1ztcQTWfspd3yD65VohhpuuCOMLASjC"
16 crossorigin="anonymous">
17
18  <link rel="stylesheet" href="Assets/CSS/style.css">
19  <title>Bootstrap 2</title>
20
21 </head>
22
23 <body>
24
25  <script>
26
27    console.log("Hola Mundo");
28
29  </script>
30
31  <!-- Option 1: Bootstrap Bundle with Popper -->
32  <script
33 src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.b
34 undle.min.js"
35   integrity="sha384-
36 MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
37   crossorigin="anonymous"></script>
```

```
38  
39 </body>  
40  
41 </html>
```

Y nuestra consola se verá así:



Podemos observar que la consola siempre nos muestra el número de línea y el archivo en el cual está escrito el código que nos está imprimiendo, para este ejemplo en el archivo index.html en la línea 22.

3.- TERCERA FORMA DE TRABAJAR CON JAVASCRIPT

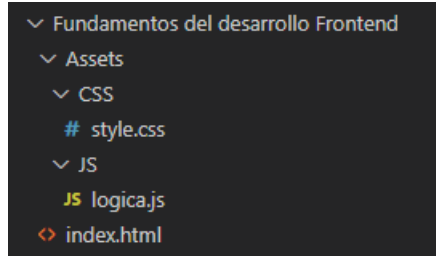
Nuevamente, eliminaremos las etiquetas `<script>` y el código escrito dentro de ellas.

La mejor forma de utilizar **JavaScript** es a través de un archivo externo al **HTML**, tal como lo hacemos con **CSS**.

De esta forma mantendremos separados los códigos y será más fácil realizar la mantención de nuestro proyecto y también, realizar la lectura del código.

Dentro de la carpeta **Assets** vamos a crear una carpeta llamada **JS** (a la misma altura de la carpeta **CSS**) y en ella, incorporaremos un archivo de nombre **lógica.js**. Es de suma importancia que la extensión sea **JS** para indicar que es un archivo de **JavaScript**.

La estructura de carpetas será la siguiente:



Luego, al final del **<body>**, justo antes del enlace a **Bootstrap**, incorporaremos una etiqueta **<script>** con el atributo **src** y en él enlazaremos nuestro archivo.

El código es el siguiente:

```
1 <script src="Assets/JS/logica.js"></script>
```

El código completo de index.html es el siguiente:

```

1 <!doctype html>
2 <html lang="es">
3
4 <head>
5   <!-- Required meta tags -->
6   <meta charset="utf-8">
7   <meta name="viewport" content="width=device-width, initial-
8 scale=1">

```

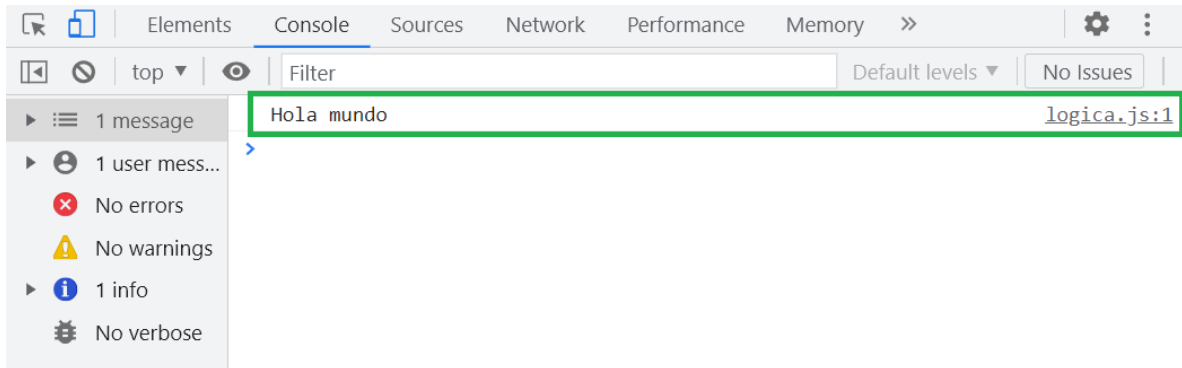
```
9
10 <!-- Bootstrap CSS -->
11 <link
12 href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap
13 .min.css" rel="stylesheet"
14 integrity="sha384-
15 EVSTQN3/azprG1Anm3QDgppJLIIm9Nao0Yz1ztcQTWfSpd3yD65VohhpUuCOmLASjC"
16 crossorigin="anonymous">
17
18 <link rel="stylesheet" href="Assets/CSS/style.css">
19 <title>Bootstrap 2</title>
20
21 </head>
22
23 <body>
24
25 <script src="Assets/JS/logica.js"></script>
26 <!-- Option 1: Bootstrap Bundle with Popper -->
27 <script
28 src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.b
29 undle.min.js"
30 integrity="sha384-
32 MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
33 crossorigin="anonymous"></script>
34
35 </body>
36
37 </html>
```

De esta forma hemos enlazado el nuevo archivo, al final del documento. Esto lo hacemos porque, al momento de abrir nuestro proyecto y que se procesa a renderizar el código (es decir, a convertir en el diseño visual que el usuario podrá ver), cargará primero todo el **HTML** y finalmente cargará el **JS**, que puede pesar más.

En el archivo **lógica.js** escribimos nuestra línea de código:

```
1 console.log("Hola mundo");
```


y en la consola, nuevamente revisamos nuestra impresión del código.



Podemos observar que ahora nos indica que el código se encuentra escrito en el archivo **lógica.js** en su línea 1.

EXERCISES 2: ESCRIBIENDO CÓDIGO CON JAVASCRIPT

Ahora que tenemos nuestro archivo **lógica.js** enlazado a nuestro archivo **index.html**, llegó el momento de comenzar a utilizar este interesante lenguaje de programación para ir complejizando nuestra escritura y comenzar a programar.

Comenzaremos creando variables. En **JavaScript**, como previamente dijimos, existen las variables y las constantes.

Como su nombre lo indica, una variable es un espacio en memoria que guardará un dato que puede variar con el tiempo y, por su parte, una constante es un espacio en memoria que guardará un dato que se mantendrá constante en el tiempo.

Escribiremos nuestra primera variable usando la palabra reservada **var** y llamaremos "**numero1**". Usando un signo igual (=) vamos a asignarle el valor 5.

```
1 var numero1 = 5;
```

luego de eso, crearemos una segunda variable de nombre numero 2 y le asignaremos el valor 10.

```
1 var numero2 = 10;
```

Finalmente, crearemos una variable de nombre resultado y el valor asignado será el correspondiente a las dos variables anteriores (es decir, sumará el valor que se encuentre contenido en las variables, independiente de cuál sea ese valor).

Para hacer eso, indicamos la suma de numero 1 y numero 2.

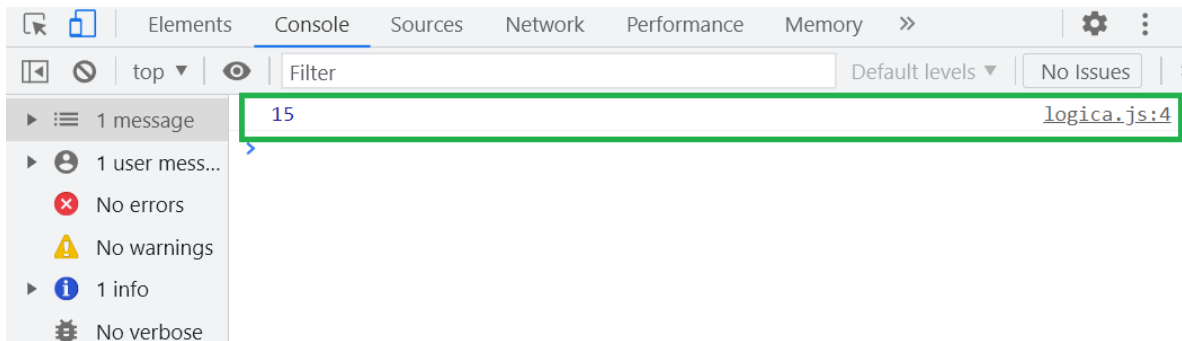
```
1 var resultado = numero1 + numero2;
```

imprimiremos este valor en la consola escribiendo, dentro del paréntesis del `console.log` el nombre de la variable resultado.

El código completo quedará así:

```
1 var numero1 = 5;  
2 var numero2 = 10;  
3 var resultado = numero1 + numero2;  
4 console.log(resultado);
```

Nos dirigiremos a nuestro navegador y, en el inspector de elementos, ingresaremos a la consola para ver que nos muestra:



Podemos agregar distintos tipos de datos a una variable, como explicamos previamente, como, por ejemplo, un dato numérico (como acabamos de ejemplificar), un dato *booleano*, que consta únicamente de un verdadero (*true*) o un falso (*false*) o una cadena de texto, que consta de texto.

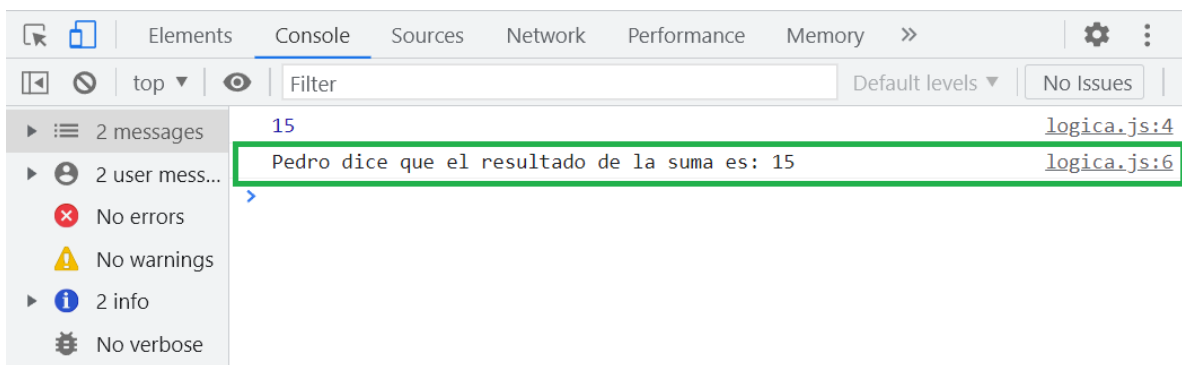
Escribamos una variable de tipo `String` (cadena de texto) e imprimámosla “concatenada” con la variable resultado.

Concatenar es enlazar o vincular hechos o ideas, en este caso, variables.

```
1 console.log(nombre + " dice que el resultado de la suma es: " +  
   resultado);
```

Acá podemos ver que, al mismo tiempo, estamos imprimiendo la variable nombre, un texto y la variable resultado. Hemos realizado dos concatenaciones.

El resultado que esto nos entrega es:



COMENTARIOS

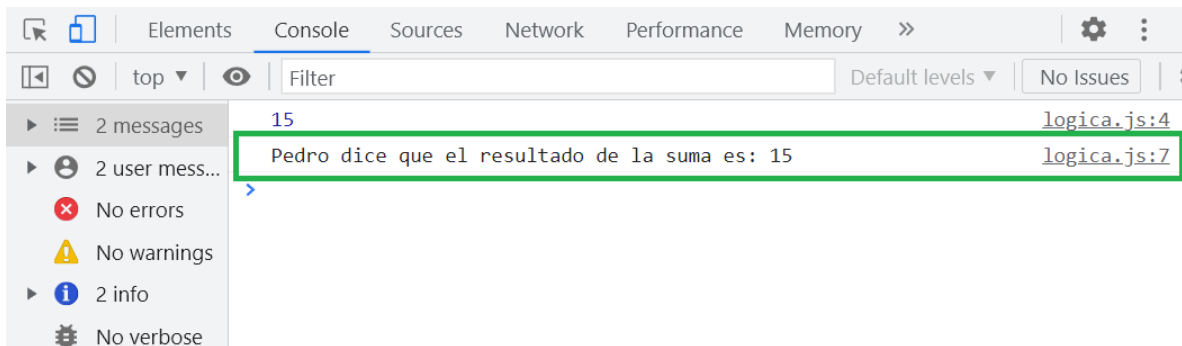
Podemos escribir comentarios, los cuales serán líneas de código ignoradas por el compilador, por lo tanto, no se verán en la consola (incluso si la indicación es una impresión en consola).

Esto nos sirve, por ejemplo, para dejar información que guíe la lectura y comprensión del código.

Se realiza anteponiendo dos "slash" (//)

```
1 var numero1 = 5;  
2 var numero2 = 10;  
3 var resultado = numero1 + numero2;  
4 console.log(resultado);
```

```
5 var nombre = "Pedro";  
6 //IMPRESION CONCATENADA console.log("esto no se imprime");  
7 console.log(nombre + " dice que el resultado de la suma es: " +  
8 resultado);
```



Podemos ver que la única diferencia entre una y otra impresión en consola es que en la segunda ocasión nos indica que se encuentra en la línea 7 y la vez anterior en la línea 6.

EXERCISE 3: REALIZANDO OPERACIONES MATEMÁTICAS

Continuando con nuestro proceso de introducción a **JavaScript**, vamos a lograr solicitar que el usuario ingrese un dato y lo imprimiremos a través de una alerta.

Para eso, vamos a declarar 3 variables (cuando hablamos de declarar variables nos referimos al momento en el cual definimos que existirá ese espacio en memoria que utilizaremos para guardar algo e indicamos un nombre para él, pero no necesariamente el valor que va a guardar) de nombre "numero1, numero2 y resultado".

```
1 var numero1, numero2, resultado;
```

Posterior a eso vamos a solicitar al usuario que ingrese un número y se lo asignaremos a la variable "numero1". Para poder realizar dicha acción escribiremos el nombre de la variable en la que deseamos asignar el valor y, utilizando un signo igual (el signo igual, siempre será para asignar algo) escribiremos la instrucción `prompt()` y en el paréntesis una frase que el usuario verá.

Recordamos que estas frases siempre deben ir entre comillas.

```
1 numero1 = prompt("Ingrese el primer número");
```

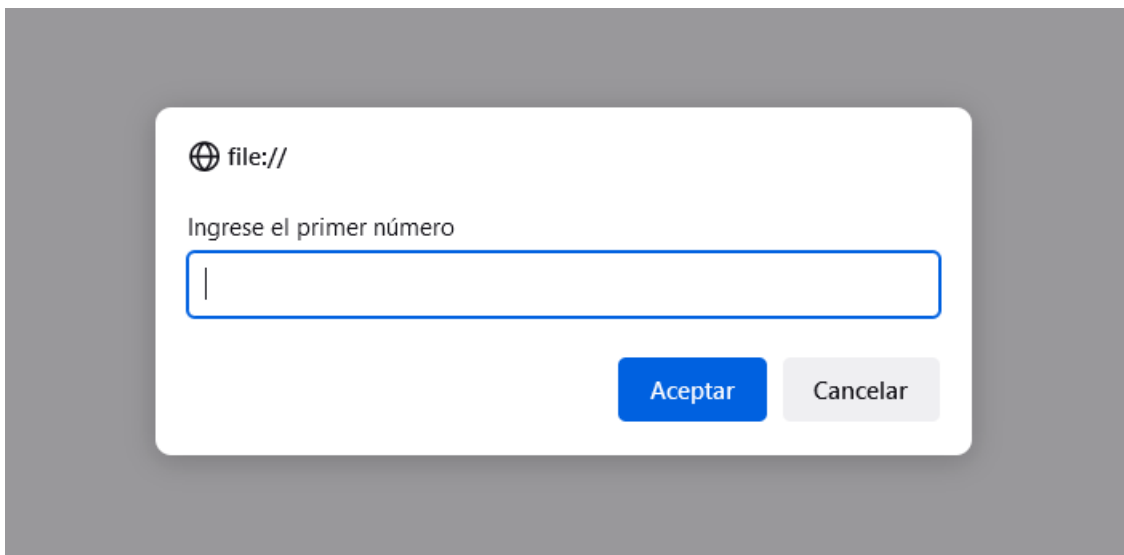
Con esta instrucción hemos asignado a la variable "numero1" el valor que el usuario ingrese, independiente de cuál sea ese valor.

En JavaScript las variables no definen previamente el valor que van a almacenar (a diferencia de algunos lenguajes en los cuales debes indicar el tipo de dato que será permitido) es por esto por lo que una variable puede guardar valores como números enteros, decimales, cadenas de texto, etc.

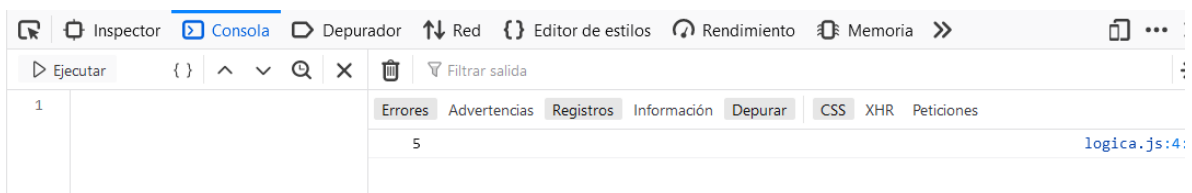
Para corroborar que hemos obtenido el valor ingresado por el usuario, vamos a imprimir en consola la variable.

```
1 console.log(numero1);
```

Ingresamos al navegador y obtenemos, al inicio, la solicitud del número:



Ingresamos un valor y apretamos aceptar. Posterior a eso, en la página no aparece nada. Para verlo ingresamos al inspector de elementos, específicamente a la consola.



Las imágenes de los ejercicios anteriores muestran el inspector de elementos de Google Chrome y las actuales muestran el inspector de elementos de Mozilla Firefox. De esta forma podemos ver que se ve levemente distinto en cada uno de los navegadores, pero que en todos se puede inspeccionar.

Volviendo a nuestro **Visual Studio Code** para continuar codificando, eliminaremos la línea de impresión del `console.log()` ahora que sabemos que está funcionando y solicitaremos el segundo número de la misma forma que la vez anterior.

```
1 numero2 = prompt("Ingrese el segundo número");
```

Una vez que hemos capturado ambos números debemos “parsearlos”, es decir, transformar el valor de un tipo de dato a otro.

Esto se debe a que cada vez que capturamos un dato de la forma que lo estamos haciendo, el dato se captura como un **String**, es decir, como una cadena de texto y, al no ser numérico, no puede realizar la suma de dos números, para eso, pasamos el dato a número, guardándolo en la misma variable:

```
1 numero1 = parseInt(numero1);  
2 numero2 = parseInt(numero2);
```

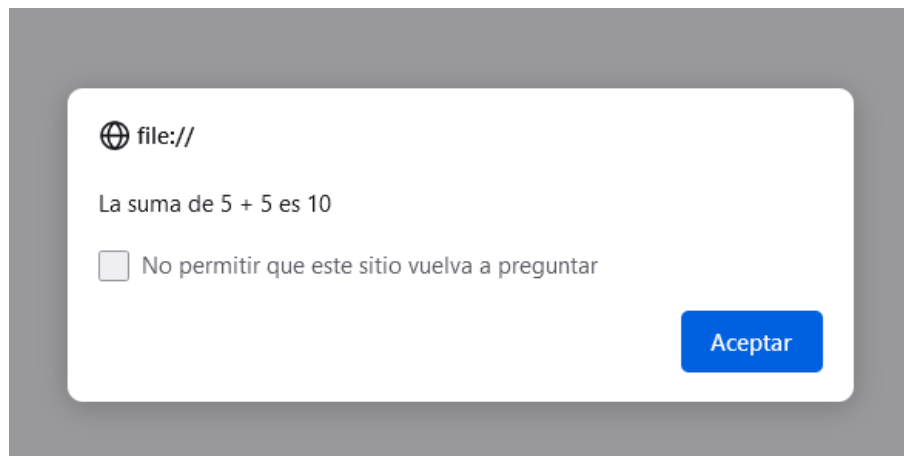

Posteriormente calculamos la suma de estos números:

```
1 resultado = numero1 + numero2;
```

Y finalmente la mostraremos al usuario a través de una alerta que se verá en la página web. En la alerta tratamos de colocar un mensaje que sea claro para el usuario concatenando lo que sea necesario.

```
1 alert("La suma de " + numero1 + " + " + numero2 + " es " + resultado);
```

Vamos al navegador, actualicemos la página web e ingresemos valores.



De esta forma hemos capturado, parseado e impreso datos utilizando variables.

EXERCISE 4: FUNCIONES Y OPERACIONES ARITMETICAS

Las funciones son uno de los bloques de construcción fundamentales de **JavaScript**. Son un bloque de instrucciones que realizan una tarea que comúnmente será repetitiva, lo que nos permite escribir la instrucción una sola vez y utilizarla en varios lugares distintos de nuestro código.

Para definir una función usaremos la palabra reservada **function** seguida del nombre de la función, la lista de parámetros que puede recibir (los valores que requiere de entrada para poder llevar a cabo la operación) y, entre llaves, las instrucciones que componen la función:

```
function nombreFuncion (valorentrada){  
  
    Instrucciones  
  
}
```

Además, las funciones pueden o no retornar un valor, es decir, devolver un valor que puede ser utilizado en otra parte de nuestro programa. Para eso se utiliza la instrucción **return** que especifica el valor devuelto por la función.

Continuando con nuestro ejercicio, realizamos una suma, pero, imaginemos que necesitamos utilizar las cuatro operaciones matemáticas básicas.

Esto lo podemos hacer con una función para cada una:

```
1 function suma(numero1, numero2) {  
2     resultado = numero1 + numero2;  
3     return resultado;  
4 }
```

Si observamos con detenimiento el código, tenemos la palabra reservada **function**, seguida del nombre de la función que se lo asignamos nosotros. Este nombre debe ser representativo del objetivo que cumple la función.

En el paréntesis incorporamos los parámetros o valores de entrada de la función, es decir, aquellos valores que debemos ingresar para que la función pueda llevar a cabo su acción. En este caso hemos ingresado los valores que el usuario ingresó y que almacenamos en esas variables.

Utilizando la variable resultado hemos sumado ambos valores y retornado su valor. Retornar este valor nos permite capturarlo fuera de la función y hacer con él lo que estimemos conveniente.

Hagamos lo mismo para resta, multiplicación y división.

```
1 function suma(numero1, numero2) {  
2     resultado = numero1 + numero2;  
3     return resultado;  
4 }  
5  
6 function resta(numero1, numero2) {  
7     resultado = numero1 - numero2;  
8     return resultado;  
9 }  
10  
11 function multiplicacion(numero1, numero2) {  
12     resultado = numero1 * numero2;  
13     return resultado;  
14 }  
15  
16 function division(numero1, numero2) {  
17     resultado = numero1 / numero2;  
18     return resultado;  
19 }
```

Hasta acá tenemos realizadas las funciones, pero aún no hacen ninguna acción porque no las hemos llamado.

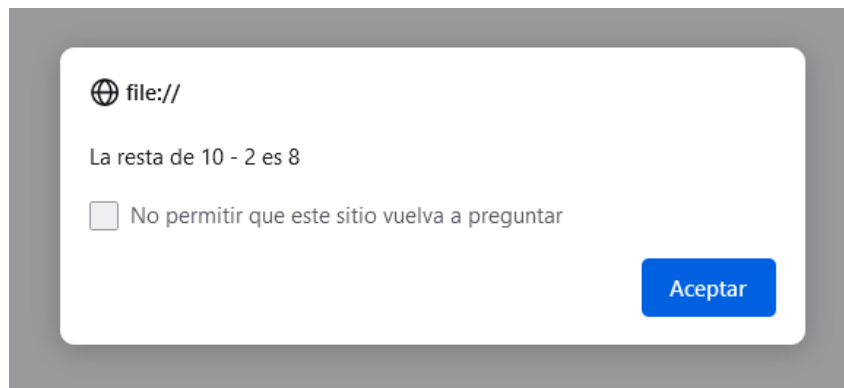
Llamar una función es el acto de utilizar esa función en beneficio del desarrollo del algoritmo realizado.

Vamos a utilizar la variable "resultado" y asignar el valor que retorne la función "resta". Posteriormente con un **alert** lo imprimimos.

```
1 resultado = resta(numero1, numero2);  
2 alert("La resta de " + numero1 + " - " + numero2 + " es " +  
3 resultado);
```

Al momento de escribir el nombre de la función y agregar los parámetros obligatorios, estamos "llamando la función".

En nuestro navegador actualizaremos nuestra página web y nos solicitará los números nuevamente, dándonos como resultado:



El código completo de nuestro archivo **lógica.js** es el siguiente:

```
1 var numero1, numero2, resultado;  
2  
3 numero1 = prompt("Ingrese el primer número");  
4 numero2 = prompt("Ingrese el segundo número");  
5  
6 numero1 = parseInt(numero1);  
7 numero2 = parseInt(numero2);  
8  
9 //LLAMADA A LA FUNCION
```

```
10 resultado = resta(numero1, numero2);
11 alert("La resta de " + numero1 + " - " + numero2 + " es " +
12 resultado);
13
14 //INICIO FUNCIONES
15 function suma(numero1, numero2) {
16     resultado = numero1 + numero2;
17     return resultado;
18 }
19
20 function resta(numero1, numero2) {
21     resultado = numero1 - numero2;
22     return resultado;
23 }
24
25 function multiplicacion(numero1, numero2) {
26     resultado = numero1 * numero2;
27     return resultado;
28 }
29
30 function division(numero1, numero2) {
31     resultado = numero1 / numero2;
32     return resultado;
33 }
```

EXERCISE 5: TRABAJANDO CON CONDICIONALES EN JAVASCRIPT

Cuando programamos en **JavaScript** la ejecución de las instrucciones se realiza de manera lineal, es decir, desde la línea 1 hacia abajo. Esto puede entorpecer nuestro trabajo cuando requerimos enviar la ejecución a otro lado del programa y no realizarla lineal.

Para solucionar esta problemática existen las sentencias condicionales de control de flujo. Estas son sentencias (que comienzan siempre con una palabra reservada de **JavaScript**) que van a realizar la evaluación de una indicación y, en consecuencia de esa evaluación, administraran el flujo de ejecución del programa.

CONDICIONAL IF

La condicional **if** (o si en español) se utiliza para ejecutar una instrucción si una condición lógica es verdadera (*true*). Utiliza opcionalmente la clausula **else** que permite indicar instrucciones a realizarse en caso de que la condición sea falsa (*false*).

La sintaxis es la siguiente:

```
if (condición) {  
    Instrucciones que se ejecutan  
    si la condición es true  
}else{  
    Instrucciones que se ejecutan  
    si la condición es false  
}
```

Vamos a comenzar a ejemplificarlo en nuestro archivo **lógica.js**.

Vamos a solicitar a un usuario que ingrese un número. Si el número ingresado es 1 ocurrirá una instrucción específica. Si ingresa cualquier otro número, ocurrirá otra instrucción.

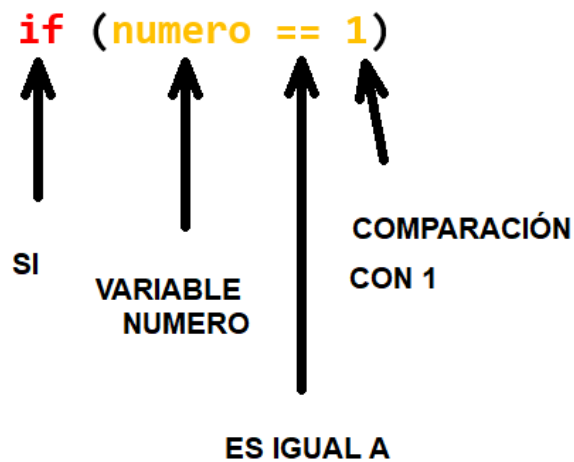
Comenzamos solicitando y parseando el número:

```
1 var numero = parseInt(prompt("Ingrese un número"));
```

Posterior a eso, comenzamos con la evaluación de la condición.

```
1 if(numero == 1)
```

En esta línea de código hemos indicado que si el número es 1 realizará algo:



Dentro de llaves vamos a indicar que ocurrirá si el número almacenado en la variable número es 1.

```
1 if(numero == 1){  
2     alert("seleccionaste el número 1. Ganaste!")  
3 }
```

Luego, utilizaremos la instrucción **else** para indicar que ocurrirá en caso de que el número ingresado sea distinto a 1.

El código completo es el siguiente:

```
1 var numero = parseInt(prompt("Ingrese un número"));
2
3 if(numero == 1){
4     alert("seleccionaste el número 1. Ganaste!")
5 }else{
6     alert("Sigue participando");
7 }
```

En nuestro navegador vamos a abrir o actualizar nuestra página web y obtendremos la pregunta (ingresamos el número que solicitamos con la instrucción **prompt**).

Luego de eso, recibimos la respuesta. En caso de ingresar 1 recibimos:

Esta página dice

seleccionaste el número 1. Ganaste!

Aceptar

En el caso de ingresar otro número:

Esta página dice

Sigue participando

Aceptar

De esta forma podemos utilizar la instrucción **if else** para evaluar condiciones.

Podemos observar que para hacer la comparación utilizamos dos signos iguales, esto se llama operadores de comparación y nos detendremos un momento en ellos.

OPERADORES DE COMPARACION

Los operadores de comparación se utilizan para comparar dos o más valores. El resultado de esta comparación siempre dará como resultado verdadero o falso (*true* o *false*).

Los operadores de comparación son los siguientes:

SÍMBOLO	DESCRIPCIÓN	EJEMPLO	BOOLEANO
==	IGUAL QUE	5 == 7	FALSE
!=	DISTINTO QUE	ROJO != VERDE	TRUE
<	MENOR QUE	8 < 12	TRUE
>	MAYOR QUE	12 > 8	TRUE
<=	MENOR O IGUAL QUE	16 <= 17	TRUE
>=	MAYOR O IGUAL QUE	67 >= 72	FALSE

Se utilizan siempre que queramos comparar algo para evaluar una condición.

Es importante tener en cuenta que cuando trabajamos con un solo símbolo igual estamos realizando una asignación de un valor y cuando trabajamos con dos signos iguales estamos comparándolos.

En **JavaScript** podemos trabajar con tres signos iguales juntos (===) que indican una comparación exacta.

Podemos aprender más en detalle sobre este tema leyendo la documentación de [Comparadores de igualdad e identidad](#).

SWITCH CASE

Continuando con las sentencias condicionales, vamos a conocer **switch case**, una instrucción que permite que un programa evalúe una expresión y envíe el flujo del programa al **case** que coincida con esa expresión.

La sintaxis es la siguiente:

```
switch (expresión){  
    case 1:  
        instrucciones  
        break;  
    case 2:  
        instrucciones  
        break;  
    default:  
        instrucciones  
        break;  
}
```

Comienza con la palabra reservada **switch** y continua con el valor, expresión o llave que será el valor que se evalúa. Cada caso (o **case**) tendrá instrucciones distintas según el valor y, el caso por defecto (**default**), se utiliza para enviar el flujo de programa a ese punto cuando no coincide con ninguno de los otros casos esperados.

En nuestro archivo **lógica.js** vamos a mantener la variable **número** solicitando un número al usuario y eliminaremos el **if else** que habíamos trabajado previamente.

Escribiremos la palabra reservada **switch** y entre paréntesis ingresaremos la variable "número". Entre llaves el **case 1** que enviará una alerta y colocamos la instrucción **break**.

La declaración **break** no es obligatoria, pero, en caso de no colocarla, el programa continuará el flujo normal de ejecución, por lo tanto, se ejecutarán los siguientes **case** sin que hayan sido los seleccionados por el usuario.

Vamos a colocar un **case 2** y **case 3** y uno por defecto, dejando el código de la siguiente manera:

```
1 var numero = parseInt(prompt("Ingrese un número"));
2
3 switch (numero) {
4     case 1:
5         alert("la opción ingresada es 1");
6         break;
7     case 2:
8         alert("la opción ingresada es dos");
9         break;
10    case 3:
11        alert("la opción ingresada es tres");
12        break;
13
14    default:
15        alert("La opción ingresada es distinto a 1, 2 o 3");
16        break;
17 }
```

Ahora nos dirigimos a nuestro navegador, actualizamos nuestra página web e ingresamos una opción, por ejemplo 2:

Esta página dice
la opción ingresada es dos

Aceptar

Ingreseemos ahora una opción distinta a 1, 2 o 3.

Esta página dice

La opción ingresada es distinto a 1, 2 o 3

Aceptar

De esta forma, hemos comenzado a introducirnos en el desarrollo de programas utilizando JavaScript y sus elementos básicos.