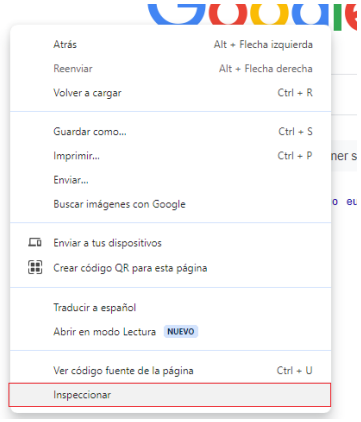
¿Qué diferencia a Javascript de cualquier otro lenguaje de programación?

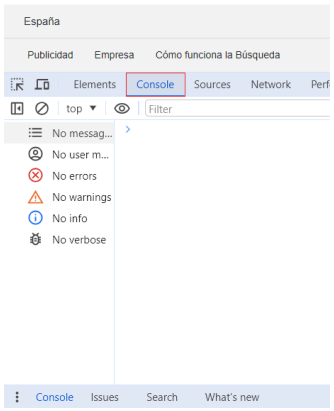
La principal razón es que es el único lenguaje de programación que entiende un navegador. Otros lenguajes como Python o Java, tienen que estar disponibles en el servidor, y es aquí donde realizan su tarea que después será traducida para que sea entendida por el navegador del cliente.

JavaScript no necesita esa configuración. Se puede ejecutar directamente en el navegador del cliente sin la necesidad de instalar ningún software adicional. Lo que lo convierte en una opción muy dinámica y flexible para generar interacción entre el usuario y el contenido del navegador.

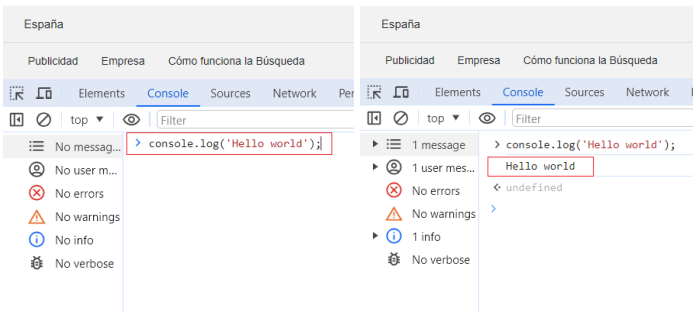
La opción más simple de desarrollo es la de “inspeccionar” la página del navegador desde el menú contextual generado al pulsar el botón derecho del ratón en la propia página.



De esta forma entramos en las opciones de desarrollador y podemos acceder a la consola de comandos donde se pueden ejecutar directamente nuestras líneas de código JavaScript.



En este ejemplo, imprimiremos “Hello world”:



¿Cuáles son algunos tipos de datos JS?

Los tipos más comunes en JavaScript son muy similares a los de otros lenguajes. Tendremos tipos para utilizar números, cadenas de texto, lógica booleana, entidades de clave y valor... veamos:

El tipo *number*: válido para números enteros o de punto flotante.

```
var numero = 12;            //Variable de tipo number
```

El tipo *string*: cadena de caracteres en formato texto.

```
var cadena = 'Mi cadena';   //Variable de tipo string
```

El tipo *boolean*: para uso en lógica en la que representará si es verdadero o falso.

```
var booleano = True;        //Variable con valor verdadero. False, para indicar falso.
```

El tipo *object*: colección de clave y valor. Se obtiene el valor asociado a la clave indicada.

```
var objeto = {nombre : 'Peter'}; //Al llamar a objeto.nombre, devuelve 'Peter'.
```

El tipo *undefined*: objeto no definido en ese momento. Podemos crear una variable pero no necesitamos asignarle un tipo hasta su uso gracias a su tipado débil.

```
var indifeinida;            //Variable sin tipo asociado.
```

¿Cuáles son las tres funciones de String en JS?

Como podemos ver en JavaScript String Reference (w3schools.com) el tipo de datos String cuenta con bastantes más funciones que sólo tres. Un dato a tener en cuenta es que devuelven un valor nuevo y no modifican la variable original.

Veamos una selección de ellas para el siguiente String:

```
var miCadena = "the quick brown fox jumps over the lazy dog" ;
```

Esta cadena es típica para testear tipografías por contener todas las letras del abecedario.

Podemos conocer su longitud, muy útil para recorrerla en bucles:

```
miCadena.length;            //43
```

Conocer si incluye alguna palabra concreta y poder confirmar accesos:

```
miCadena.includes('fox');   //true
```

Cambiar su formato a mayúsculas y simplificar su uso si se han introducido datos sin un formato de mayúsculas y minúsculas controlado:

```
miCadena.toUpperCase()      // "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"
```

¿Qué es un condicional?

Un condicional es una estructura básica para controlar la ejecución de un bloque de código y otra. Se crea así un comportamiento.

En seudocódigo sería algo así:

Si es cierto:

Ejecuta este bloque

Si no lo era:

Ejecuta este otro bloque

Sus palabras clave serían: if, else, else if.

```
//Si es cierto

if(condición){    }

//Si no lo era, pero es cierta esta otra opción

else if(condición 2){   }

//Si no es ninguna de las dos

else{}
```

Para realizar condiciones podemos usar comparaciones entre dos o más datos o variables, o simplemente confirmar si una variable es de un tipo concreto o incluso cierta, ejemplos:

```
var edad = 15;

if(edad > 18){ }          //Compara si la edad sería mayor a 18 años, al no serlo, no ejecuta el código que seguiría entre corchetes.

var acceso = true;

if(acceso) { }          //Confirma si es cierto.
```

¿Qué es un operador ternario?

Es un operador condicional muy útil cuando necesitas devolver un resultado de una condición creada en una sola línea, como por ejemplo dentro de un código HTML o en una sentencia return. La sintaxis sería la siguiente:

```
Condición ? Resultado para cierto : Resultado para falso;

console.log(edad > 18 ? “Puedes pasar” : “No puedes pasar”);

//Imprime puedes o no puedes pasar, según el valor de edad.
```

En una página HTML podría formar parte de un texto y cambiarlo según la respuesta y en una función podría devolver el resultado si la usamos después del return.

¿Cuál es la diferencia entre una declaración de función y una expresión de función?

Una declaración de función es un bloque de código que utiliza la palabra clave *function* y que puede ser llamado desde cualquier parte del código, por lo que es necesario que tenga un nombre. Ejemplo:

```
function sumaDosValores(a, b){
    return a+b;
}
SumaDosValores(2,7);                    //Imprime 9
```

Por otro lado, una expresión de función suele ser denominadas también anónimas y se asigna a una variable. De esta forma no pueden ser llamadas desde otras partes del código.

```
var a =5;
var b =7;
var resultado = function(){
    return a+b;
};
Console.log(resultado());               //Imprime 12
```

La ventaja de una expresión es que puede ser utilizada en bloques de código como dentro de estructuras condicionales, por ejemplo:

```
if(typeof a== 'number' && typeof b == 'number'){
    var resultado = function(){
        return a+b;
    };
}
```

¿Qué es la palabra clave "this" en JS?

Se refiere al objeto instanciado en esa ejecución. Si hablamos de las variables de una función, su uso se limitaría a las instancias de esas variables en esa llamada, como si fuese una flecha que apunta su propio objeto durante su tiempo de ejecución, veamos un simple ejemplo:

```
function saludar(nombre){
    this.nombre = nombre;
    console.log('Mi nombre es ' + this.nombre + '.');
}
saludar('Peter');                       //Mi nombre es Peter
```

Hace referencia a la variable creada en esa llamada, a la que está en uso durante esa ejecución y no a la que se pueda crear con otra llamada y otro nombre.

Pero su uso no se limita sólo a funciones, si estamos en un ámbito global, this sería el objeto window para el caso de un navegador. Y si estamos intentando capturar una pulsación en un icono, this se usaría para capturar esa pulsación concreta.

Pueden consultar el siguiente enlace para ver estos ejemplos:
https://www.w3schools.com/js/js_this.asp