

Curso HTML5, CSS y JavaScript

Módulo 5: Introducción a JavaScript

- emilianoagustingallo@gmail.com
- <https://www.linkedin.com/in/emiliano-gallo/>



¿Que es JavaScript?

Es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones más complejas en una páginas web.

Es quien le da comportamiento a nuestro sitios.

Lenguajes compilados e interpretados

Ambos convierten el código que escribimos en **lenguaje de máquina**.

El **lenguaje de máquina** es lo que sabe leer el procesador de los ordenados, dispositivos móviles, etc.

```
package main

import "fmt"

func main() {
    fmt.Printf("hello, world")
}
```

Lenguaje de alto nivel que
entiende el programador



```
0101010111101110001101
0100010100010101001010
0101010010101010000101
0011010001010100011110
0110010100101010101001
1110001101010010010001
```

Lenguaje de máquina que
entiende el procesador

Lenguajes compilados e interpretados

La principal diferencia es que el lenguaje **compilado** requiere un paso adicional antes de ser ejecutado, la compilación, que convierte el código que escribes a **lenguaje de máquina**.

Un lenguaje **interpretado**, es convertido a lenguaje de máquina a medida que es ejecutado.

- Algunos lenguajes compilados: C, C++, C#, JAVA
- Algunos lenguajes interpretados: **JavaScript**, Python, Ruby

Ventajas en lenguajes **interpretados**

- El ciclo de desarrollo es más rápido ya que no tenemos que compilar el código
- No es necesario generar ejecutables para **sistema operativo** que se vaya a usar
- Está optimizado para simplificar la tarea del programador (aunque esto resulte una carga adicional para el ordenador)

Ventajas en lenguajes compilados

- Más rápido en **runtime**. Ya que el código fue compilado a lenguaje de máquina y también fue optimizado
- Para ser usado no es necesario tener un **interpretador** instalado en nuestro ordenador
- Está optimizado para el momento de la ejecución (aunque esto resulte una carga adicional para el desarrollador)

Las 3 capas de una web

- **HTML** es el lenguaje de marcado que usamos para estructurar y dar significado a nuestro contenido web, por ejemplo, definiendo párrafos, encabezados y tablas de datos, etc.
- **CSS** es un lenguaje de reglas de estilo que usamos para darle una imagen a nuestro contenido HTML, por ejemplo, establecer colores de fondo y tipos de letra, etc.
- **JavaScript** es un lenguaje de secuencias que nos permite crear contenido de actualización dinámica, controlar multimedia, animar imágenes, persistir información, etc.

JavaScript en el <head>

```
<!DOCTYPE html>
<html>

<head>
<script>
function myFunction() {
    document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>

<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```


JavaScript en el <body>

```
<!DOCTYPE html>
<html>
<body>

<h1>A Web Page</h1>
<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

JavaScript en archivo externo

```
<script src="myScript1.js"></script>  
<script src="myScript2.js"></script>
```

```
function myFunction() {  
    document.getElementById("demo").innerHTML = "Paragraph changed.";  
}
```

Ventajas de usar archivo externo

- Separa **HTML** y **código**
- Hace que **HTML** y **JavaScript** sean más fáciles de leer y mantener
- Los archivos **JavaScript** en caché pueden acelerar la carga de la página
- Podemos incluir más de un archivo JS a nuestra página

```
<script src="myScript1.js"></script>
```

```
<script src="myScript2.js"></script>
```

Mostrando información en JS

- Escribir en un elemento HTML, usando **innerHTML**
- Escribir en la salida HTML usando **document.write ()**
- Escribir en un cuadro de alerta, usando **window.alert ()**
- Escribir en la consola del navegador, usando **console.log ()**

Método **innerHTML**

Podemos acceder a un elemento **HTML**, usando el método **document.getElementById (id)**.

El atributo **id** define el elemento **HTML**.

La propiedad **innerHTML** define el contenido **HTML**

Método innerHTML

```
<!DOCTYPE html>
<html>
<body>

<h2>Título</h2>
<p>Párrafo.</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>

</body>
</html>
```

Título

Párrafo.

11

Método `document.write()`

```
<!DOCTYPE html>
<html>
<body>

<h2>Texto</h2>
<p>Párrafo.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

Texto

Párrafo.

11

Método `windows.alert()`

```
<!DOCTYPE html>
<html>
<body>

<h2>My First Web Page</h2>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

My First Web Page

My first paragraph.

An embedded page on this page says

11

OK

Método `console.log()`

```
<!DOCTYPE html>
<html>
<body>

<h2>Activate Debugging</h2>

<p>F12 on your keyboard will activate debugging.</p>
<p>Then select "Console" in the debugger menu.</p>
<p>Then click Run again.</p>

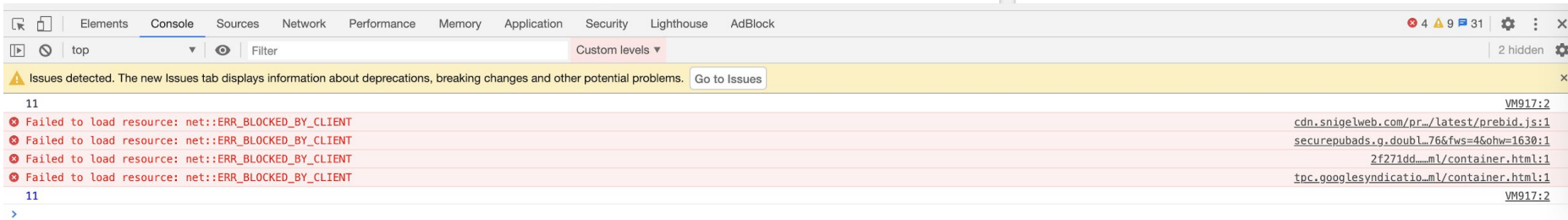
<script>
console.log(5 + 6);
</script>
```

Activate Debugging

F12 on your keyboard will activate debugging.

Then select "Console" in the debugger menu.

Then click Run again.



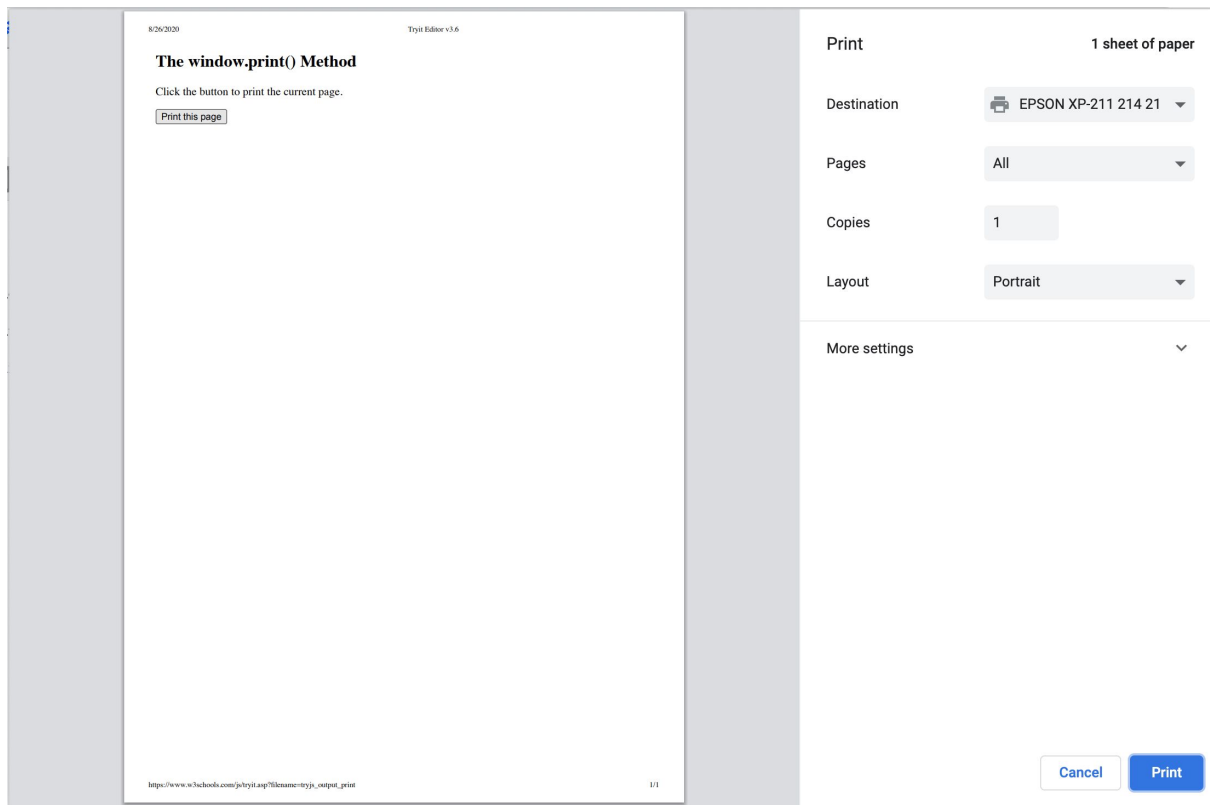
Método `print()`

JavaScript no tiene ningún objeto o método de impresión.

No puede acceder a los dispositivos de salida desde JavaScript.

La única forma de hacerlo es llamando al método **`window.print ()`** en el navegador para imprimir el contenido de la ventana actual.

Método `print()`



¿Qué es un programa JS?


Un programa de computadora es una lista de **instrucciones** que una computadora debe **ejecutar**.

En un lenguaje de programación, estas instrucciones de programación se denominan **declaraciones**.


Un programa **JavaScript** es una lista de declaraciones de programación.

En **HTML**, el código **JS** es ejecutado en el navegador.

Características de un algoritmo

Somos programadores


¿Qué es un algoritmo?



Es la secuencia de pasos que hay que seguir para resolver un problema.


Partes de un algoritmo

Entrada




Datos que recibe el algoritmo

Proceso







Operaciones realizadas con los datos de entrada

Salida



Resultado obtenido de la operación con los datos de entrada

Un algoritmo debe ser claro, tener un inicio y un fin y ser capaz de resolver el problema.

 Somos programadores  @somosprograma2  @somos_programadores_  Somos programadores

Declaraciones o Statements

Se componen de:

- Valores
- Operadores
- Expresiones
- Palabras clave
- Comentarios

```
var x, y, z;           // Statement 1
x = 5;                 // Statement 2
y = 6;                 // Statement 3
z = x + y;             // Statement 4
```

Diferencia entre **expression** y **statement**

La sintaxis del lenguaje JavaScript distingue entre **expresiones** y **declaraciones**. Estos dos conceptos son sutilmente diferentes.

En general, una **expresión** es un fragmento de código que se evalúa como un valor.

Una **declaración** es un fragmento de código que realiza una acción.

```
0 // 0
1 + 1 // 2
'Hello' + ' ' + 'World' // 'Hello World'
{ answer: 42 } // { answer: 42 }
Object.assign({}, { answer: 42 }) // { answer: 42 }
answer !== 42 ? 42 : answer // 42
answer = 42 // 42
```

```
// 'if' statement
if (answer !== 42) { answer = 42 }
// 'for' is a statement
for (;;) { console.log('Hello, World'); }
// Declaring a variable is a statement
let answer = 42
```

Punto y coma ;

El **punto y coma** separa las declaraciones de JavaScript. Debemos agregarlo al final de cada instrucción ejecutable.

```
var a, b, c;      // Declare 3 variables
a = 5;            // Assign the value 5 to a
b = 6;            // Assign the value 6 to b
c = a + b;        // Assign the sum of a and b to c
```

Cuando están separados por **punto y coma**, se permiten varias declaraciones en una misma línea:

```
a = 5; b = 6; c = a + b;
```


Espacios en blanco

JavaScript ignora varios espacios. Se puede agregar espacios en blanco a su secuencia de comandos para que sea más legible.

Una buena práctica es poner espacios alrededor de los operadores (= + - * /):

Las siguientes líneas son equivalentes:

```
var person = "Hege";
```

```
var person="Hege";
```

Palabras claves **Keywords**

- **break:** termina un interruptor o un bucle
- **continue:** salta un bucle y comienza en la parte superior
- **debugger:** detiene la ejecución de JS y llama a la función de depuración
- **do ... while:** ejecuta un bloque de código y lo repite mientras la condición sea verdadera
- **for:** marca un bloque de código para ser ejecutado mientras la condición sea verdadera
- **function:** declara una función
- **if ... else:** marca un bloque de código a ser ejecutado si se cumple una determinada condición
- **return:** sale de una función
- **switch:** marca un bloque de código para ser ejecutado dependiendo de diferentes casos
- **try ... catch:** implementa manejo de errores a un bloque de código
- **var:** declara una variable

Identificadores en JS

Los identificadores son nombres.

En JavaScript, los identificadores se utilizan para nombrar variables, funciones y etiquetas.

Las reglas para los nombres legales son muy parecidas en la mayoría de los lenguajes de programación.

En JavaScript, el primer carácter debe ser una letra, un guión bajo (_) o un signo de dólar (\$). No se permiten números como primer carácter.

Los caracteres siguientes pueden ser letras, dígitos, guiones bajos o signos de dólar.

Minúsculas de mayúsculas case sensitive

Todos los identificados en JS son **case sensitive**. Esto significa que no es lo mismo la letra **'a'** que **'A'**

En el siguiente ejemplo 'lastname' y 'lastName' son variables distintas:

```
var lastname, lastName;  
lastName = "Doe";  
lastname = "Peterson";
```

Caja de camello camel case

Históricamente los programadores han utilizado diferentes maneras de escribir nombres de variables con palabras compuestas:

- Guión medio: first-name, last-name, master-card, inter-city
- Guión bajo: first_name, last_name, master_card, inter_city
- Camel case mayúscula: FirstName, LastName, MasterCard, InterCity
- Camel case minúscula: firstName, lastName, masterCard, interCity.

Los programadores en JS tienden a usar lower camel case

Algunos ejemplos

