
Unidad 1 – Primeros pasos Javascript

DWEC - 2º DAW

Rafael Torres Ferrer

Unidad 1 – Primeros pasos Javascript

1. Integración de Javascript con código HTML
2. Ficheros externos
3. Herramientas y entornos de desarrollo web
4. La sintaxis del lenguaje

1. Integración de Javascript con código HTML

- El código JavaScript siempre irá entre las etiquetas <script> i </script>
- Estas etiquetas pueden situarse:
 - En la sección <head> </head>
 - En la sección <body> </body> del document HTML

El código JavaScript al final de la etiqueta <body> </body> mejora la velocidad de carga de la página.

- Los scripts de JavaScript también pueden almacenarse en un archivo externo (extensión.js) y hacer una llamada desde el código principal → aprovechamiento de código

1. Integración de Javascript con código HTML

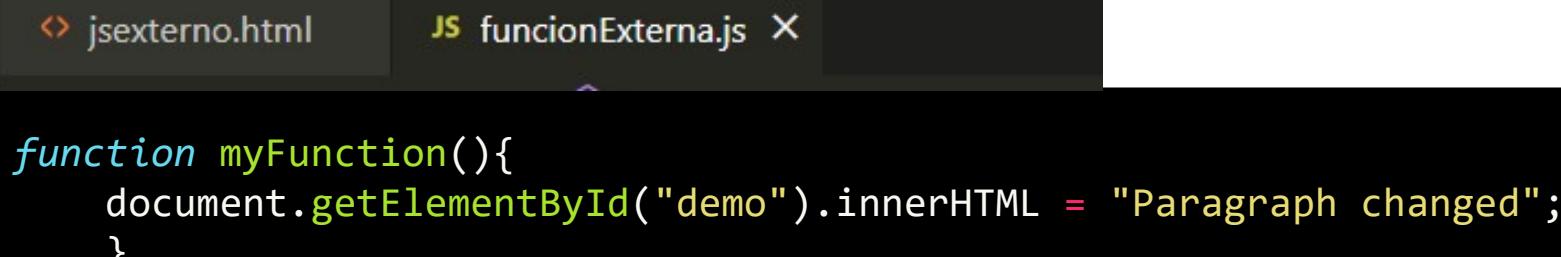
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Mi primer ejemplo</title>
  </head>

  <body>
    <h1>Mi primer ejemplo</h1>
    <p id="parrafo">Texto original del párrafo</p>
    <button type="button" onclick="Saludamos()">Clícame</button>
    <script>
      function Saludamos() {
        document.getElementById("parrafo").innerHTML = ("Hola, bienvenidos");
      }
    </script>
  </body>

</html>
```

2. Ficheros externos

```
<!DOCTYPE html>
<html>
  <body>
    <h1>A Web Page</h1>
    <p id="demo">A Paragraph</p>
    <button type="button" onclick="myFunction()">Try it</button>
    <script src="funcionExterna.js"></script>
  </body>
</html>
```



The screenshot shows a code editor interface with two tabs. The left tab is labeled "jsexterno.html" and contains the HTML code shown above. The right tab is labeled "JS funcionExterna.js" and contains the following JavaScript code:

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

3. Herramientas y entornos de desarrollo web

- Para programar con JavaScript, necesitamos un editor de texto + un entorno de desarrollo (IDE).
- Los navegadores web más actuales ya incorporan un IDE (Integrated Development Environment)
- Del editor esperamos:
 - Selección del lenguaje JavaScript
 - Coloración sintáctica
 - Autocompletado
 - Herramientas avanzadas de búsqueda y reemplazo
 - Instalación de plugins con funcionalidades extra
 - Sublime, Notepad++(Windows), Notepadqq(Linux), Brackets

3. Herramientas y entornos de desarrollo web

- Otras opciones:
 - Instalar un IDE, como Eclipse, NetBeans ó Visual Studio Code (hay que conocer a fondo estas herramientas)
 - Un entorno de JavaScript online (codepen.io) →
<https://codepen.io/>

4. La sintaxis del lenguaje

1. Especificaciones oficiales: ECMAScript

- EcmaScript es el estándar que, desde 2015, rige cómo debe interpretarse y cómo debe funcionar el lenguaje JavaScript → compatibilidad entre navegadores, sistemas operativos, etc.
- Ha ido introduciendo nuevas versiones mejoradas:
 - Primera versión → EcmaScript 1 (junio 2019)
 - La más actual (versión 15) → Ecma-262 (2024)
<https://ecma-international.org/publications-and-standards/standards/ecma-262/>

4. La sintaxis del lenguaje

2. Sentencias, comentarios y variables

- Un programa es un conjunto de sentencias que serán interpretadas por un intérprete de JavaScript.
- No es necesario acabar con punto y coma (;), pero sí recomendable.
- JavaScript es CaseSensitive (distingue entre mayúsculas y minúsculas)

4. La sintaxis del lenguaje

2. Sentencias, comentarios y variables

- Normas sobre las variables:
 - Es recomendable declararlas, pero no obligatorio.
 - Las declaramos con la palabra reservada var, let o cons.
 - Sólo pueden contener letras, dígitos, signo de subrayado (_) y el signo de dólar (\$).
 - Deben empezar con una letra (o bien con el signo de subrayado _) o el signo de dólar).
 - No podemos utilizar palabras reservadas del lenguaje como nombre de variable.

4. La sintaxis del lenguaje

2. Sentencias, comentarios y variables

```
var price1 = 5;  
var price2 = 6;  
var total = price1 + price2;  
  
var person = "John Doe", carName = "Volvo", price = 200;  
  
var person = "John Doe",  
carName = "Volvo",  
price = 200;
```

4. La sintaxis del lenguaje

2. Sentencias, comentarios y variables

Si queremos declarar variables que sean válidas solo en el ámbito en el que se declaran usaremos **let**:

```
if (2> 1)

{
    let name = "Nacho";

    console.log ("Name inside:", name);

}

console.log ("Name out:", name);
```

Este código daría error.

4. La sintaxis del lenguaje

2. Sentencias, comentarios y variables

También podemos usar la palabra **const** para definir constantes en el código. Esto será especialmente útil tanto para definir constantes convencionales (como un texto o un número fijo, por ejemplo) como para cargar bibliotecas.

```
const pi = 3.1416;
```

4. La sintaxis del lenguaje

2. Sentencias, comentarios y variables

Comentarios → para explicar el código y hacerlo más legible

Comentarios de línea simple → // This is a comment...

Comentarios multilínea → /* This is a comment

with multiple lines */

```
// Change heading:  
document.getElementById("myH").innerHTML = "My First Page";  
  
/*  
The code below will change  
the heading with id = "myH"  
and the paragraph with id = "myP"  
in my web page:  
*/  
document.getElementById("myH").innerHTML = "My First Page";  
document.getElementById("myP").innerHTML = "My first paragraph.";
```

4. La sintaxis del lenguaje

3. Tipos de datos soportados

- Undefined
- Null
- Boolean
- String (cadena de texto o literal)
- Numérico
- Objetos

4. La sintaxis del lenguaje

3. Tipos de datos soportados

- **Undefined** → cualquier variable a la que no se ha asignado ningún valor, es de una variable de tipo “Undefined” → no hay que confundirla con las cadenas vacías.
- **Null** → equivale a lo mismo que Undefined, pero las variables de tipo Null son objetos y las de tipo Undefined, no lo son.

```
var z;
console.log(typeof(z)); //undefined
z="";
console.log(typeof(z)); //string
z=null;
console.log(typeof(z)); //object
```

Si queremos ejecutar código javascript directamente podemos usar node en la terminal de visual Studio code:
-> node nombre.js

```
PS C:\Users\bea_v\Documents\SERPIS_24_25\dwecl\ejercicios\1. introJS> node 3.datos.js
undefined
string
object
```

4. La sintaxis del lenguaje

3. Tipos de datos soportados

- . **Boolean:** las variables pueden contener los valores lógicos ‘true’ y ‘false’

```
var1=undefined;
console.log(typeof(var1)); //muestra el resultado 'Undefined'
var2=null;
console.log(typeof(var2)); ///muestra el resultado 'Objecte'
console.log(var1==var2); // true
console.log(var1==>var2); // false --> Compara el valor y el tipo
```

4. La sintaxis del lenguaje

Tipos de datos soportados

.String: conjunto entre 0 y $2^{53}-1$ caracteres, para representar información textual (codificación UTF-16). Se engloban entre comillas (simples o dobles)

```
var answer1 = "It's alright";           // Single quote inside double quotes
var answer2 = "He is called 'Johnny"';   // Single quotes inside double quotes
var answer3 = 'He is called "Johnny"';    // Double quotes inside single quotes
```

4. La sintaxis del lenguaje

3. Tipos de datos soportados

- **Numérico:** podemos escribirlos con o sin decimales.

```
var x1 = 34.00;      // Written with decimals  
var x2 = 34;        // Written without decimals
```

- Podemos utilizar la notación científica por números muy grandes o muy pequeños.

```
var y = 123e5;       // 12300000  
var z = 123e-5;     // 0.00123
```

4. La sintaxis del lenguaje

3. Tipos de datos soportados

- **Objetos:** tipo de variable con propiedades y métodos

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

4. La sintaxis del lenguaje

3. Tipos de datos soportados

- Los objetos se definen entre llaves. Podemos definir sus propiedades (con pares nombre:valor y separadas por comas)

```
var coche = {id: 1235, marca : "Audi", color : "azul", matricula : "2356ABC"};
console.log(coche.matricula);
```

y los métodos:

```
var coche = {
    id: 1235,
    marca: "Audi",
    color: "azul",
    matricula : "2356ABC",
    obtener_matricula: function(){
        return this.matricula;
    }
};
console.log(coche.obtener_matricula());
//2356ABC
```

4. La sintaxis del lenguaje

4. Asignaciones

Los operadores de asignación asignan valor a las variables. Aparte del operador igual ('='), podemos utilizar los siguientes operadores:

$a += b$ equivale a $a = a + b$

$a -= b$ equivale a $a = a - b$

$a *= b$ equivale a $a = a * b$

$a /= b$ equivale a $a = a / b$

$a %= b$ equivale a $a = a \% b$

4. La sintaxis del lenguaje

4. Asignaciones: operadores

Los operadores sirven para realizar operaciones entre dos o más variables:

- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)
- Resto (%)
- Incremento (++)
- Decremento (--)

4. La sintaxis del lenguaje

4. Asignaciones: operadores

Como operador de cadena tenemos la concatenación. Se representa con el símbolo + en el sentido de que estamos sumando cadenas:

```
var cadena1 = "Hola, bona vesprada"  
var cadena2 = ". Com esteu?"  
var cadena3 = cadena1 + cadena2;  
console.log(cadena3); //o directament console.log(cadena1 + cadena2)
```

4. La sintaxis del lenguaje

4. Asignaciones: operadores

Los operadores lógicos y de comparación disponibles:

igual en valor que (==)

mayor o igual que (>=)

igual en valor y en tipos que (====)

más pequeño o igual que (<=)

no igual en valor que (!=)

conjunction lógica (&&)

no igual en valor o en tipos que (!==)

disyunction lógica (||)

mayor que (>)

negación lógica (!)

más pequeño que (<)

operador ternario (?)

4. La sintaxis del lenguaje

4. Asignaciones, operadores

No debemos confundir la asignación (=) con la comparación lógica (==). Es un error frecuente de los principiantes.

```
var a = 10; var b = 10; var c = 20;
var d = "10";
console.log (a == c); // false
console.log (a === b); // true
console.log (a == d); // true
console.log (a === d); // false
console.log (a != c); // true
console.log (a !== d); // true
console.log (a > c); // false
console.log (a < c); // true
console.log (a >= b); // true
console.log (a <= b); // true
```

4. La sintaxis del lenguaje

5. La comparación entre cadenas se realiza de forma alfabética:

```
var cadena1 = "llibre";
var cadena2 = "revista";
console.log(cadena1 > cadena2); //false
```

Si hacemos una comparación entre números y cadenas, JavaScript convertirá la cadena a número. Si la cadena es no numérica, la convertirá a NaN (Not a Number); si la cadena está vacía, la convertirá a 0.

```
var a=10;
var cad1 = "10";
var cad2 = "15";
var str = "cotxe";
console.log (a < cad2); //true
console.log (a == str); //false
console.log (cad1 < cad2);//true
```

4. La sintaxis del lenguaje

6. Sentencias condicionales: nos permiten controlar el flujo del programa y bifurcar el código.

- If, else, else if
- Utilizaremos **if** para especificar un bloque de código que se ejecutará cuando la condición sea verdadera:

```
var a = 99, b=98;
if (a % 3 == 0) console.log(a + ' es múltiplo de 3');
if (b % 2 == 0) {
  console.log (b + " es par");
  console.log (b + " es múltiplo de 2");
```

4. La sintaxis del lenguaje

6. Sentencias condicionales

Utilizamos **else** para especificar el bloque de código que se ejecutará cuando la condición es falsa:

```
var b=98;
if (b % 2 == 0) {
    console.log (b + " es par");
    console.log(b + " es múltiplo de 2");
} else {
    console.log (b + ' es impar');
}
```

4. La sintaxis del lenguaje

6. Sentencias condicionales

Utilizamos **else if** cuando queremos que el programa evalúe diversas condiciones:

```
var b = 54;
if ( b % 4 == 0) {
    console.log( b +' es divisible por 4');
    console.log(b + ' es par');
} else if ( b % 3 ==0) {
    console.log(b + ' es divisible por 3');
} else if ( b % 2 == 0) {
    console.log(b + ' es divisible por 2');
    console.log(b + ' es par');
} else {
    console.log('otro caso');
}
```

4. La sintaxis del lenguaje

6. Sentencias condicionales

Switch: se utiliza cuando tenemos una sola expresión que debe compararse muchas veces, y sólo una de ellas será la correcta.

```
var mes;
var estacion;

switch (new Date().getMonth()) {
    case 0:
        mes = "Enero";
        estacion = "Invierno";
        break;
    case 1:
        mes = "Febrero";
        estacion = "Invierno";
        break;
    case 2:
        mes = "Marzo";
        estacion = "Invierno a primavera";
        break;
    case 8:
        mes = "Septiembre";
        estacion = "Otoño";
        break;
    case 11:
        mes = "Diciembre";
        estacion = "Otoño a invierno";
        break;
    default:
        console.log("Pasa por aqui si no entra en ninguno de los casos anteriores");
}
console.log ("Mes: " + mes);
console.log ("Estación: " + estacion);
```

4. La sintaxis del lenguaje

7. Bucles

La sintaxis básica del bucle for:

```
for ( expressió1; expressió2; expressió3) {  
    bloc de codi a executar;  
}
```

Ejemplo:

```
var cad = "";  
for (i = 1; i <= 10; i++){  
    cad += "6 * " + i + " = " + 6*i + "\n";  
}  
console.log(cad);
```

4. La sintaxis del lenguaje

7. Bucles

La cláusula **break** fuerza la salida del bucle

```
var i = 0;
for ( ; ; ){
    console.log(i);
    i++;
    if (i >= 15) break;
}
```

4. La sintaxis del lenguaje

7. Bucles

La sintaxis básica del bucle while:

```
while (condició) {  
    bloc de codi  
}
```

Ejemplo:

```
var str = "Bienvenidos al curso 2024-2025";  
var cad = "";  
var i = 0;  
var res = "";  
  
while (res != 'c') {  
    res = str.charAt(i);  
    cad += res;  
    i++;  
}  
console.log (cad);
```

4. La sintaxis del lenguaje

7. Bucles

La sintaxis básica del bucle do/while:

```
do {  
    bloc de codi  
}  
while (condició);
```

Ejemplo:

```
var str = "Bienvenidos al curso 2024-2025";  
var cad = "";  
var i = 0;  
var res = "";  
  
do {  
    res = str.charAt(i);  
    cad += res;  
    i++;  
} while (res != 'B')  
  
console.log(cad);
```